```
**********************************************************
    5850 Wed May 27 19:48:55 2015
new/./README-ASLR.md
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
```

```
  1 # Address Space Layout Randomisation
  3 Briefly, see: ‘psecflags(1)‘, ‘security-flags(5)‘
  5 ## Administration
  7 ASLR is implemented via process security-flags (which we introduce), there are
  8 two sets of flags per-process:  The effective set and the inheritable set (see
  9 ‘security-flags(5)‘).  The effective set is immutable, it can only change when
 10 the process calls ‘exec(2)‘, at which point the effective set is replaced by
 11 the inheritable set (with one exception).  Security flags are inherited upon
 12 ‘fork(2)‘ (but the inheritable set is not promoted until ‘exec(2)‘, as
 13 mentioned).
 15 This is such that a given execution of an executable has a constant set of
 16 security-flags, which simplifies things for everyone.
 18 This unfortunately means that to enable ASLR fully system-wide, requires a
 19 reboot or at least restart of a majority of services.
 21 The system-wide ASLR flag is an SMF property on the new service
 22 ‘svc:/system/process-security‘, which contains a ‘secflags‘ property group,
 23 with one boolean property per implemented flag (see, again,
 24 ‘security-flags(5)‘).  At present, this will not take effect (at all) until a
 25 reboot.
 27 Per-process setting (and inspecting) of security-flags is done via
 28 ‘psecflags(1)‘.
 30 ## Privilege
 32 A process may change the security-flags of any process to which it could send
 33 a signal with ‘kill(2)‘, as long as the process also has the
 34 ‘PRIV_PROC_SECFLAGS‘ privilege.  This privilege is granted by default, but is
 35 not a ‘basic‘ privilege.  If you have configured custom privileges for certain
 36 users or services, they will not automatically gain the new
 37 ‘PRIV_PROC_SECFLAGS‘ (unfortunately).
 39 ## Executable tagging
 41 There is a somewhat compatible property of dynamic executables, ‘DT_SUNW_ASLR‘
 42 which controls the ASLR behaviour of a given executable.  If this dynamic tag
 43 has a value of 1, ASLR is always enabled for an execution of this process.  If
 44 the tag has a value of 0, ASLR is never enabled for an execution of this
 45 process.  The default is to inherit the ASLR flag as normal.
 47 This allows a process for which ASLR is known to be problematic to explicitly
 48 forbid it, and for processes of special sensitivity to mandate it.
 50 This is controlled via the ‘-z aslr‘ flag to ‘ld(1)‘.
 52 ## Missing bits/Problems/Worries
 54 ### The stack skewing may skew too much of the stack
 56 At present, we skew the absolute base of the stack, which means the gap
 57 between a user stack frame and the process environment is constant.  I have
 58 this vague memory that that’s sub-par, and that we want to skew the stack
```



```
 59 _after_ the environment, etc.  I could be entirely wrong about that though.
 61 ### We skew each mapping separately
 63 Some systems calculate a random skew for the various parts of a process at
 64 execution time, and apply that same skew to each mapping.  That obviously
 65 minimises the performance impact of this significantly for processes that
 66 perform many mappings.
 68 Due to some unfortunate aspects of how we manage the user address space and
 69 mappings, we don’t do that right now.  We calculate a separate random skew for
 70 each mapping we attempt.
 72 ### The way we skew mappings is problematic
 74 The user address space is currently managed somewhat unfortunately (from our
 75 point of view, at least).  When attempting a mapping we first determine the
 76 highest gap into which the requested mapping can fit, and then we place the
 77 mapping at the highest address in that gap.  This is how we manage user
 78 fragmentation.
 80 The current ASLR implementation works in basically the same way.  We still
 81 find the highest gap into which the given mapping may fit, and choose the
 82 highest address at which it may fit.  The only difference is that we then slew
 83 it backward by a random (but co-aligned), amount.
 85 This is obviously not as random as it could be, but is the easiest way I’ve
 86 found in the current code base for introducing uncertainty while also
 87 preserving any attempt at preventing unbounded user fragmentation.
 89 It is not impossible to imagine a long-lived process that makes many mappings
 90 being in a position where mappings later in its life are 100% predictable
 91 given this implementation, however.  In fact, I think it is fairly likely.
 93 I think the most at risk would be a process which makes many mappings, and
 94 uses ‘dlopen(3c)‘ at unpredictable times (rather than, as is most common, during
 95 setup).  Dynamic objects tend to have strict (and high) alignment requirements
 96 which in such a process are likely to only be fulfillable at a single location
 97 in the address space gap we choose, and thus be entirely predictable.
 98
 99 ### We want a per-zone configuration item, but it’s not implemented yet
101 I plan to implement a per-zone configuration item similar to ‘limitpriv‘ which
102 describes a zones default security-flags.  This will be set during
103 ‘zone_create‘ and apply to _every_ process in a zone (rather than the GZ
104 implementation, from which ‘svc.startd(1M)‘, ‘svc.configd(1M)‘, and ‘init(1M)‘
105 are immune unless tagged).
107 This will also allow a global zone administrator to configure the
108 security-flags of a zone, and then take ‘PRIV_PROC_SECFLAGS‘ away from that
109 zone, such that the security-flag configuration of the zone is forced upon it.
111 This is somewhat ugly though, since ‘svc:/system/process-security‘ and its
112 settings in the zone will thus be inoperative.
114 ### Randomisation of executable base addresses requires PIE
116 To randomise the executable base address, we need position independent
117 executables, which appears like it would turn into a whole separate project
118 (though perhaps not too large a project).  That code isn’t here, so the
119 executable base is fixed.
121 This means that rather than return-to-libc one could return-to-executable
122 trivially and successfully, I think.  (That would include using the
123 executable’s PLT to vector yourself to libc.  Should the executable have a PLT
124 entry for something useful to you).
```

```
125 #endif /* ! codereview */
```

```
**********************************************************
    2277 Wed May 27 19:48:56 2015
new/usr/src/cmd/ptools/Makefile
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
```

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#

#
# Copyright 2008 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#


include ../Makefile.cmd

#
# Don't add new ptools here; these are legacy ptools which must be symlinked
# into /usr/proc/bin
#
LEGACY_SUBDIRS =\
        pcred   \
        pfiles  \
        pflags  \
        pldd    \
        pmap    \
        prun    \
        psig    \
        pstack  \
        pstop   \
        ptime   \
        ptree   \
        pwait   \
        pwdx

#
# 'new' ptools are not symlinked into /usr/proc/bin
#
NEW_SUBDIRS =   \
        pargs   \
        plgrp   \
        pmadvise \
        ppriv   \
        preap   \
        psecflags

SUBDIRS = $(LEGACY_SUBDIRS) $(NEW_SUBDIRS)

all     :=      TARGET = all
install :=      TARGET = install
clean   :=      TARGET = clean
clobber :=      TARGET = clobber
lint    :=      TARGET = lint
_msg    :=      TARGET = _msg


# pmadvise depends on pmap components
PMAP =                  $(SRC)/cmd/ptools/pmap
pmadvise/pmadvise.po := CPPFLAGS +=     -I$(PMAP)

#
# Commands with messages support
#
POFILES = plgrp/plgrp.po pmadvise/pmadvise.po psecflags/psecflags.po
POFILES = plgrp/plgrp.po pmadvise/pmadvise.po
POFILE = ptools.po

.KEEP_STATE:

.PARALLEL: $(SUBDIRS)

all install clean lint: $(SUBDIRS)
clobber: $(SUBDIRS) clobber_local
clobber_local:
        $(RM) $(CLOBBERFILES)

$(NEW_SUBDIRS): FRC
        @cd $@; pwd; $(MAKE) PTOOL_TYPE=NEW -f ../Makefile.ptool $(TARGET)

$(LEGACY_SUBDIRS): FRC
        @cd $@; pwd; $(MAKE) PTOOL_TYPE=LEGACY -f ../Makefile.ptool $(TARGET)

#
# Combine all messages files into a single file and copy it to
# MSGDOMAIN directory
#
_msg: ${POFILES}
        $(RM) $(POFILE)
        $(CAT) $(POFILES) > $(POFILE)
        $(RM)  $(MSGDOMAIN)/$(POFILE)
        $(CP) $(POFILE) $(MSGDOMAIN)

FRC:
```

```
**********************************************************
    3201 Wed May 27 19:48:56 2015
new/usr/src/cmd/ptools/Makefile.bld
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
```

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#

#
# Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#

PROG:sh = basename ‘cd ..; pwd‘

OBJS = $(PROG).o

SRCS = ../$(PROG).c

FILEMODE = 0555

# libproc is added individually as pwait doesn’t need it.
# These are defined this way so lint can use them
LDLIBS_pargs            = -lproc
LDLIBS_pcred            = -lproc
LDLIBS_pfiles           = -lproc -lnsl
LDLIBS_pflags           = -lproc
LDLIBS_pldd             = -lproc
LDLIBS_plgrp            = -lproc -llgrp
LDLIBS_pmap             = -lproc
LDLIBS_pmadvise         = -lproc
LDLIBS_ppriv            = -lproc
LDLIBS_preap            = -lproc
LDLIBS_prun             = -lproc
LDLIBS_psecflags        = -lproc -lproject
#endif /* ! codereview */
LDLIBS_psig             = -lproc
LDLIBS_pstack           = -lproc -lc_db
LDLIBS_pstop            = -lproc
LDLIBS_ptime            = -lproc
LDLIBS_ptree            = -lproc -lcontract
LDLIBS_pwdx             = -lproc

LDLIBS += $(LDLIBS_$(PROG))
```



```
CERRWARN_plgrp  += -_gcc=-Wno-parentheses

CERRWARN_ppriv  += -_gcc=-Wno-parentheses
CERRWARN_ppriv  += -_gcc=-Wno-uninitialized

CERRWARN_ptree  += -_gcc=-Wno-parentheses

CERRWARN_pstack += -_gcc=-Wno-uninitialized
CERRWARN_pstack += -_gcc=-Wno-clobbered

CERRWARN_pargs  += -_gcc=-Wno-clobbered
CERRWARN_pargs  += -_gcc=-Wno-type-limits

CERRWARN += $(CERRWARN_$(PROG))

# pargs depends on ../../common/elfcap components
# pmadvise depends on pmap components

ELFCAP =        $(SRC)/common/elfcap
PMAP =          $(SRC)/cmd/ptools/pmap

CPPFLAGS_pargs =        -I$(ELFCAP)
OBJS_pargs =            elfcap.o
SRCS_pargs =            $(ELFCAP)/elfcap.c

CPPFLAGS_pmap =         -I$(PMAP)
OBJS_pmap =             pmap_common.o
SRCS_pmap =             $(PMAP)/pmap_common.c

CPPFLAGS_pmadvise =     -I$(PMAP)
OBJS_pmadvise =         pmap_common.o
SRCS_pmadvise =         $(PMAP)/pmap_common.c

CPPFLAGS +=     $(CPPFLAGS_$(PROG))
OBJS +=         $(OBJS_$(PROG))
SRCS +=         $(SRCS_$(PROG))

INSTALL_NEW=
INSTALL_LEGACY=$(RM) $(ROOTPROCBINSYMLINK) ; \
        $(LN) -s ../../bin/$(PROG) $(ROOTPROCBINSYMLINK)

.KEEP_STATE:

elfcap.o:       $(ELFCAP)/elfcap.c
                $(COMPILE.c) -o $@ $(ELFCAP)/elfcap.c

pmap_common.o:  $(PMAP)/pmap_common.c
                $(COMPILE.c) -o $@ $(PMAP)/pmap_common.c

%.o:    ../%.c
        $(COMPILE.c) $<

all:    $(PROG)

ROOTBINPROG=$(ROOTBIN)/$(PROG)
ROOTPROCBINSYMLINK=$(ROOT)/usr/proc/bin/$(PROG)

$(PROG): $$(OBJS)
        $(LINK.c) $(OBJS) -o $@ $(LDLIBS)
        $(POST_PROCESS)

#
# Install the ptool, symlinking it into /usr/proc/bin if PTOOL_TYPE is set
# to LEGACY.
#
install: all $(ROOTISAPROG)
```

```
125         -$(RM) $(ROOTBINPROG)
126         -$(LN) $(ISAEXEC) $(ROOTBINPROG)
127         -$(INSTALL_$(PTOOL_TYPE))

129 clean:
130         $(RM) $(OBJS)

132 lint:
133         $(LINT.c) $(SRCS) $(LDLIBS)
```

```
**********************************************************
   38714 Wed May 27 19:48:56 2015
new/usr/src/cmd/ptools/pargs/pargs.c
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
______unchanged_portion_omitted_

775 static struct auxsecfl {
776         uint_t af_flag;
777         const char *af_name;
778 } auxsecfl[] = {
779         { PROC_SEC_ASLR,        "aslr" },
780 };

782 /*ARGSUSED*/
783 static void
784 at_secflags(long val, char *instr, size_t n, char *str)
785 {
786         int i;

788         *str = '\0';

790         for (i = 0; i < sizeof (auxsecfl)/sizeof (struct auxsecfl); i++) {
791                 if ((val & auxsecfl[i].af_flag) != 0) {
792                         if (*str != '\0')
793                                 (void) strlcat(str, ",", n);
794                         (void) strlcat(str, auxsecfl[i].af_name, n);
795                 }
796         }
797 }

799 #endif /* ! codereview */
800 #define MAX_AT_NAME_LEN 15

802 struct aux_id {
803         int aux_type;
804         const char *aux_name;
805         void (*aux_decode)(long, char *, size_t, char *);
806 };

808 static struct aux_id aux_arr[] = {
809         { AT_NULL,              "AT_NULL",              at_null },
810         { AT_IGNORE,            "AT_IGNORE",            at_null },
811         { AT_EXECFD,            "AT_EXECFD",            at_null },
812         { AT_PHDR,              "AT_PHDR",              at_null },
813         { AT_PHENT,             "AT_PHENT",             at_null },
814         { AT_PHNUM,             "AT_PHNUM",             at_null },
815         { AT_PAGESZ,            "AT_PAGESZ",            at_null },
816         { AT_BASE,              "AT_BASE",              at_null },
817         { AT_FLAGS,             "AT_FLAGS",             at_null },
818         { AT_ENTRY,             "AT_ENTRY",             at_null },
819         { AT_SUN_UID,           "AT_SUN_UID",           at_uid  },
820         { AT_SUN_RUID,          "AT_SUN_RUID",          at_uid  },
821         { AT_SUN_GID,           "AT_SUN_GID",           at_gid  },
822         { AT_SUN_RGID,          "AT_SUN_RGID",          at_gid  },
823         { AT_SUN_LDELF,         "AT_SUN_LDELF",         at_null },
824         { AT_SUN_LDSHDR,        "AT_SUN_LDSHDR",        at_null },
825         { AT_SUN_LDNAME,        "AT_SUN_LDNAME",        at_null },
826         { AT_SUN_LPAGESZ,       "AT_SUN_LPAGESZ",       at_null },
827         { AT_SUN_PLATFORM,      "AT_SUN_PLATFORM",      at_str  },
828         { AT_SUN_EXECNAME,      "AT_SUN_EXECNAME",      at_str  },
829         { AT_SUN_HWCAP,         "AT_SUN_HWCAP",         at_hwcap },
830         { AT_SUN_HWCAP2,        "AT_SUN_HWCAP2",        at_hwcap2 },
```



```
831         { AT_SUN_IFLUSH,        "AT_SUN_IFLUSH",        at_null },
832         { AT_SUN_CPU,           "AT_SUN_CPU",           at_null },
833         { AT_SUN_MMU,           "AT_SUN_MMU",           at_null },
834         { AT_SUN_LDDATA,        "AT_SUN_LDDATA",        at_null },
835         { AT_SUN_AUXFLAGS,      "AT_SUN_AUXFLAGS",      at_flags },
836         { AT_SUN_EMULATOR,      "AT_SUN_EMULATOR",      at_str  },
837         { AT_SUN_BRANDNAME,     "AT_SUN_BRANDNAME",     at_str  },
838         { AT_SUN_BRAND_AUX1,    "AT_SUN_BRAND_AUX1",    at_null },
839         { AT_SUN_BRAND_AUX2,    "AT_SUN_BRAND_AUX2",    at_null },
840         { AT_SUN_BRAND_AUX3,    "AT_SUN_BRAND_AUX3",    at_null },
841         { AT_SUN_SECFLAGS,      "AT_SUN_SECFLAGS",      at_secflags },
775         { AT_SUN_BRAND_AUX3,    "AT_SUN_BRAND_AUX3",    at_null }
842 };
______unchanged_portion_omitted_
```

```
**********************************************************
    7568 Wed May 27 19:48:57 2015
new/usr/src/cmd/ptools/psecflags/psecflags.c
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
```

```
 1 /*
 2  * This file and its contents are supplied under the terms of the
 3  * Common Development and Distribution License ("CDDL"), version 1.0.
 4  * You may only use this file in accordance with the terms of version
 5  * 1.0 of the CDDL.
 6  *
 7  * A full copy of the text of the CDDL should have accompanied this
 8  * source.  A copy of the CDDL is also available via the Internet at
 9  * http://www.illumos.org/license/CDDL.
10  */

12 #include <err.h>
13 #include <errno.h>
14 #include <grp.h>
15 #include <libintl.h>
16 #include <procfs.h>
17 #include <project.h>
18 #include <pwd.h>
19 #include <stdio.h>
20 #include <stdlib.h>
21 #include <string.h>

23 #include <libproc.h>
24 #include <libzonecfg.h>

26 extern const char *__progname;
27 extern int psecflags(idtype_t, id_t, psecflags_cmd_t, uint_t);

29 struct flagdesc {
30         uint_t value;
31         char *name;
32 } flagdescs[] = {
33         { PROC_SEC_ASLR, "aslr" },
34         { 0x0, NULL }
35 };

37 void
38 print_flags(char *set, uint_t flags)
39 {
40         char *buf;
41         size_t buflen = 13;     /* 0x + 8 digits + comma + space + NUL */
42         struct flagdesc *fd;

44         if (flags == 0) {
45                 (void) printf(gettext("\t%s:\tnone\n"), set);
46                 return;
47         }

49         for (fd = flagdescs; fd->value != 0x0; fd++)
50                 buflen += strlen(fd->name) + 2; /* + comma, space */

52         if ((buf = calloc(buflen, sizeof (char))) == NULL)
53                 err(1, gettext("allocation failed"));

55         for (fd = flagdescs; fd->value != 0; fd++) {
56                 if (flags & fd->value)
57                         (void) snprintf(buf, buflen, "%s, %s", buf, fd->name);
58                 flags &= ~fd->value;
```



```
59         }

61         if (flags != 0) /* Contained unknown flags */
62                 (void) snprintf(buf, buflen, "%s, 0x%08x", buf, flags);

64         buf += 2;               /* Skip the comma on the first entry */

66         (void) printf("\t%s:\t%s\n", set, buf);
67         free(buf);
68 }

70 typedef struct {
71         psecflags_cmd_t cmd;
72         uint_t value;
73 } act_arg_t;

75 act_arg_t *
76 parse_secflags(const char *spec)
77 {
78         char *flag = NULL;
79         char *s;
80         struct flagdesc *fd;
81         boolean_t fail = B_FALSE;
82         act_arg_t *ret;

84         if ((ret = malloc(sizeof (act_arg_t))) == NULL)
85                 err(1, gettext("allocation failed"));

87         if ((s = strdup(spec)) == NULL)
88                 err(1, gettext("allocation failed"));

90         if (s[0] == '+') {
91                 ret->cmd = PSECFLAGS_ENABLE;
92                 s++;
93         } else if (s[0] == '-') {
94                 ret->cmd = PSECFLAGS_DISABLE;
95                 s++;
96         } else {
97                 ret->cmd = PSECFLAGS_SET;
98         }

100         if (strncasecmp("none", s, strlen(s)) == 0) {
101                 if (ret->cmd != PSECFLAGS_SET)
102                         errx(1, gettext("'none' is only valid with set, not "
103                             "enable or disable"));
104                 ret->value = 0x0;
105                 free(s);
106                 return (ret);
107         }

109 next:
110         while ((flag = strsep(&s, ",")) != NULL) {
111                 for (fd = flagdescs; fd->value != 0x0; fd++) {
112                         if (strncasecmp(flag, fd->name,
113                             strlen(fd->name)) == 0) {
114                                 ret->value |= fd->value;
115                                 goto next;
116                         }
117                 }

119                 if (strncasecmp("none", flag, strlen(flag)) == 0) {
120                         (void) fprintf(stderr, gettext("%s: 'none' is not "
121                             "valid with other flags\n"), __progname);
122                 } else {
123                         (void) fprintf(stderr, gettext("%s: invalid "
124                             "security-flag: '%s'\n"),
```

```
125                             __progname, flag);
126                 }
127                 fail = B_TRUE;
128         }

130         free(s);

132         if (fail)
133                 exit(1);

135         return (ret);
136 }

138 /*
139  * Structure defining idtypes known to the priocntl command
140  * along with the corresponding names and a liberal guess
141  * of the max number of procs sharing any given ID of that type.
142  * The idtype values themselves are defined in <sys/procset.h>.
143  */
144 static struct idtypes {
145         idtype_t        type;
146         char            *name;
147 } idtypes [] = {
148         { P_ALL,        "all"           },
149         { P_CTID,       "contract"      },
150         { P_CTID,       "ctid"          },
151         { P_GID,        "gid"           },
152         { P_GID,        "group"         },
153         { P_PGID,       "pgid"          },
154         { P_PID,        "pid"           },
155         { P_PPID,       "ppid"          },
156         { P_PROJID,     "project"       },
157         { P_PROJID,     "projid"        },
158         { P_SID,        "session",      },
159         { P_SID,        "sid"           },
160         { P_SID,        "sid"           },
161         { P_TASKID,     "taskid"        },
162         { P_UID,        "uid"           },
163         { P_UID,        "user"          },
164         { P_ZONEID,     "zone"          },
165         { P_ZONEID,     "zoneid"        },
166         { 0,            NULL            }
167 };

169 int
170 str2idtype(char *idtypnm, idtype_t *idtypep)
171 {
172         struct idtypes  *curp;

174         for (curp = idtypes; curp->name != NULL; curp++) {
175                 if (strncasecmp(curp->name, idtypnm,
176                     strlen(curp->name)) == 0) {
177                         *idtypep = curp->type;
178                         return (0);
179                 }
180         }
181         return (-1);
182 }

184 id_t
185 getid(idtype_t type, char *value)
186 {
187         struct passwd *pwd;
188         struct group *grp;
189         id_t ret;
190         char *endp;
```



```
192         switch (type) {
193         case P_UID:
194                 if ((pwd = getpwnam(value)) != NULL)
195                         return (pwd->pw_uid);
196                 break;
197         case P_GID:
198                 if ((grp = getgrnam(value)) != NULL)
199                         return (grp->gr_gid);
200                 break;
201         case P_PROJID:
202                 if ((ret = getprojidbyname(value)) != (id_t)-1)
203                         return (ret);
204                 break;
205         case P_ZONEID:
206                 if (zone_get_id(value, &ret) == 0)
207                         return (ret);
208                 break;
209         default:
210                 break;
211         }

213         ret = (id_t)strtoul(value, &endp, 10);
214         if (errno || *endp != '\0')
215                 return ((id_t)-1);

217         return (ret);
218 }

220 int
221 main(int argc, char **argv)
222 {
223         act_arg_t *act = NULL;
224         int ret = 0;
225         int pgrab_flags = PGRAB_RDONLY;
226         int opt;
227         char *idtypename = NULL;
228         idtype_t idtype = P_PID;
229         boolean_t usage = B_FALSE;
230         boolean_t e_flag = B_FALSE;
231         boolean_t l_flag = B_FALSE;

233         while ((opt = getopt(argc, argv, "eFi:ls:")) != -1) {
234                 switch (opt) {
235                 case 'e':
236                         e_flag = B_TRUE;
237                         break;
238                 case 'F':
239                         pgrab_flags |= PGRAB_FORCE;
240                         break;
241                 case 'i':
242                         idtypename = optarg;
243                         break;
244                 case 's':
245                         act = parse_secflags(optarg);
246                         break;
247                 case 'l':
248                         l_flag = B_TRUE;
249                         break;
250                 default:
251                         usage = B_TRUE;
252                         break;
253                 }
254         }

256         argc -= optind;
```

```
257         argv += optind;

259         if (l_flag && ((idtypename != NULL) || (act != NULL) || (argc != 0)))
260                 usage = B_TRUE;
261         if ((idtypename != NULL) && (act == NULL))
262                 usage = B_TRUE;
263         if (e_flag && (act == NULL))
264                 usage = B_TRUE;
265         if (!l_flag && argc <= 0)
266                 usage = B_TRUE;

268         if (usage) {
269                 (void) fprintf(stderr,
270                     gettext("usage:\t%s [-F] { pid | core } ...\n"),
271                     __progname);
272                 (void) fprintf(stderr,
273                     gettext("\t%s -s [-+]flags [-i idtype] id ...\n"),
274                     __progname);
275                 (void) fprintf(stderr,
276                     gettext("\t%s -s [-+]flags -e command [arg]...\n"),
277                     __progname);
278                 (void) fprintf(stderr, gettext("\t%s -l\n"), __progname);
279                 return (2);
280         }

282         if (l_flag) {
283                 struct flagdesc *fd;

285                 for (fd = flagdescs; fd->value != 0x0; fd++) {
286                         (void) printf("%s\n", fd->name);
287                 }
288                 return (0);
289         } else if ((act != NULL) && e_flag) {
290                 if (psecflags(P_PID, P_MYID, act->cmd, act->value) != 0)
291                         err(1, gettext("failed setting security-flags"));

293                 (void) execvp(argv[0], &argv[0]);
294                 err(1, "%s", argv[0]);
295         } else if (act != NULL) {
296                 int i;
297                 id_t id;

299                 if (idtypename != NULL)
300                         if (str2idtype(idtypename, &idtype) == -1)
301                                 errx(1, gettext("No such id type: '%s'"),
302                                     idtypename);

304                 for (i = 0; i < argc; i++) {
305                         if ((id = getid(idtype, argv[i])) == (id_t)-1) {
306                                 errx(1, gettext("invalid or non-existent "
307                                     "identifier: '%s'"), argv[i]);
308                         }

310                         if (psecflags(idtype, id, act->cmd, act->value) != 0)
311                                 err(1, gettext("failed setting "
312                                     "security-flags"));
313                 }

315                 return (0);
316         }

318         /* Display the flags for the given pids */
319         while (argc-- > 0) {
320                 struct ps_prochandle *Pr;
321                 const char *arg;
322                 psinfo_t psinfo;
```



```
323                 int gcode;

325                 if ((Pr = proc_arg_grab(arg = *argv++, PR_ARG_ANY,
326                     pgrab_flags, &gcode)) == NULL) {
327                         warnx(gettext("cannot examine %s: %s"),
328                             arg, Pgrab_error(gcode));
329                         ret = 1;
330                         continue;
331                 }

333                 (void) memcpy(&psinfo, Ppsinfo(Pr), sizeof (psinfo_t));
334                 proc_unctrl_psinfo(&psinfo);

336                 if (Pstate(Pr) == PS_DEAD) {
337                         (void) printf(gettext("core '%s' of %d:\t%.70s\n"),
338                             arg, (int)psinfo.pr_pid, psinfo.pr_psargs);
339                 } else {
340                         (void) printf("%d:\t%.70s\n",
341                             (int)psinfo.pr_pid, psinfo.pr_psargs);
342                 }

344                 print_flags("E", Pstatus(Pr)->pr_secflags.psf_effective);
345                 print_flags("I", Pstatus(Pr)->pr_secflags.psf_inherit);

347                 Prelease(Pr, 0);
348         }

350         return (ret);
351 }
352 #endif /* ! codereview */
```

```
**********************************************************
   53635 Wed May 27 19:48:57 2015
new/usr/src/cmd/sgs/dump/common/dump.c
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
______unchanged_portion_omitted_


/*
 * Print dynamic linking information.  Input is an ELF
 * file descriptor, the SCNTAB structure, the number of
 * sections, and the filename.
 */
static void
dump_dynamic(Elf *elf_file, SCNTAB *p_scns, int num_scns, char *filename)
{
#define pdyn_Fmtptr     "%#llx"

        Elf_Data        *dyn_data;
        GElf_Dyn        p_dyn;
        GElf_Phdr       p_phdr;
        GElf_Ehdr       p_ehdr;
        int             index = 1;
        int             lib_scns = num_scns;
        SCNTAB          *l_scns = p_scns;
        int             header_num = 0;
        const char      *str;

        (void) gelf_getehdr(elf_file, &p_ehdr);

        if (!p_flag)
                (void) printf("\n  **** DYNAMIC SECTION INFORMATION ****\n");

        for (; num_scns > 0; num_scns--, p_scns++) {
                GElf_Word       link;
                int             ii;


                if (p_scns->p_shdr.sh_type != SHT_DYNAMIC)
                        continue;

                if (!p_flag) {
                        (void) printf("%s:\n", p_scns->scn_name);
                        (void) printf("[INDEX]\tTag         Value\n");
                }

                if ((dyn_data = elf_getdata(p_scns->p_sd, NULL)) == 0) {
                        (void) fprintf(stderr, "%s: %s: no data in "
                            "%s section\n", prog_name, filename,
                            p_scns->scn_name);
                        return;
                }

                link = p_scns->p_shdr.sh_link;
                ii = 0;

                (void) gelf_getdyn(dyn_data, ii++, &p_dyn);
                while (p_dyn.d_tag != DT_NULL) {
                        union {
                                Conv_inv_buf_t          inv;
                                Conv_dyn_flag_buf_t     dyn_flag;
                                Conv_dyn_flag1_buf_t    dyn_flag1;
                                Conv_dyn_feature1_buf_t dyn_feature1;
```



```
                                Conv_dyn_posflag1_buf_t dyn_posflag1;
                        } conv_buf;

                        (void) printf("[%d]\t%-15.15s ", index++,
                            conv_dyn_tag(p_dyn.d_tag,
                            p_ehdr.e_ident[EI_OSABI], p_ehdr.e_machine,
                            DUMP_CONVFMT, &conv_buf.inv));

                        /*
                         * It would be nice to use a table driven loop
                         * here, but the address space is too sparse
                         * and irregular. A switch is simple and robust.
                         */
                        switch (p_dyn.d_tag) {
                        /*
                         * Items with an address value
                         */
                        case DT_PLTGOT:
                        case DT_HASH:
                        case DT_STRTAB:
                        case DT_RELA:
                        case DT_SYMTAB:
                        case DT_INIT:
                        case DT_FINI:
                        case DT_REL:
                        case DT_DEBUG:
                        case DT_TEXTREL:
                        case DT_JMPREL:
                        case DT_INIT_ARRAY:
                        case DT_FINI_ARRAY:
                        case DT_INIT_ARRAYSZ:
                        case DT_FINI_ARRAYSZ:
                        case DT_PREINIT_ARRAY:
                        case DT_PREINIT_ARRAYSZ:
                        case DT_SUNW_RTLDINF:
                        case DT_SUNW_CAP:
                        case DT_SUNW_CAPINFO:
                        case DT_SUNW_CAPCHAIN:
                        case DT_SUNW_SYMTAB:
                        case DT_SUNW_SYMSORT:
                        case DT_SUNW_TLSSORT:
                        case DT_PLTPAD:
                        case DT_MOVETAB:
                        case DT_SYMINFO:
                        case DT_RELACOUNT:
                        case DT_RELCOUNT:
                        case DT_VERSYM:
                        case DT_VERDEF:
                        case DT_VERDEFNUM:
                        case DT_VERNEED:
                                (void) printf(pdyn_Fmtptr,
                                    EC_ADDR(p_dyn.d_un.d_ptr));
                                break;

                        /*
                         * Items with a string value
                         */
                        case DT_NEEDED:
                        case DT_SONAME:
                        case DT_RPATH:
                        case DT_RUNPATH:
                        case DT_SUNW_AUXILIARY:
                        case DT_SUNW_FILTER:
                        case DT_CONFIG:
                        case DT_DEPAUDIT:
                        case DT_AUDIT:
```

```
1186                        case DT_AUXILIARY:
1187                        case DT_USED:
1188                        case DT_FILTER:
1189                                if (v_flag) {   /* Look up the string */
1190                                        str = (char *)elf_strptr(elf_file, link,
1191                                            p_dyn.d_un.d_ptr);
1192                                        if (!(str && *str))
1193                                                str = (char *)UNKNOWN;
1194                                        (void) printf("%s", str);
1195                                } else {        /* Show the address */
1196                                        (void) printf(pdyn_Fmtptr,
1197                                            EC_ADDR(p_dyn.d_un.d_ptr));
1198                                }
1199                                break;

1201                        /*
1202                         * Items with a literal value
1203                         */
1204                        case DT_PLTRELSZ:
1205                        case DT_RELASZ:
1206                        case DT_RELAENT:
1207                        case DT_STRSZ:
1208                        case DT_SYMENT:
1209                        case DT_RELSZ:
1210                        case DT_RELENT:
1211                        case DT_PLTREL:
1212                        case DT_BIND_NOW:
1213                        case DT_CHECKSUM:
1214                        case DT_PLTPADSZ:
1215                        case DT_MOVEENT:
1216                        case DT_MOVESZ:
1217                        case DT_SYMINSZ:
1218                        case DT_SYMINENT:
1219                        case DT_VERNEEDNUM:
1220                        case DT_SPARC_REGISTER:
1221                        case DT_SUNW_SYMSZ:
1222                        case DT_SUNW_SORTENT:
1223                        case DT_SUNW_SYMSORTSZ:
1224                        case DT_SUNW_TLSSORTSZ:
1225                        case DT_SUNW_STRPAD:
1226                        case DT_SUNW_CAPCHAINENT:
1227                        case DT_SUNW_CAPCHAINSZ:
1228                        case DT_SUNW_ASLR:
1229 #endif /* ! codereview */
1230                                (void) printf(pdyn_Fmtptr,
1231                                    EC_XWORD(p_dyn.d_un.d_val));
1232                                break;

1234                        /*
1235                         * Integer items that are bitmasks, or which
1236                         * can be otherwise formatted in symbolic form.
1237                         */
1238                        case DT_FLAGS:
1239                        case DT_FEATURE_1:
1240                        case DT_POSFLAG_1:
1241                        case DT_FLAGS_1:
1242                        case DT_SUNW_LDMACH:
1243                                str = NULL;
1244                                if (v_flag) {
1245                                        switch (p_dyn.d_tag) {
1246                                        case DT_FLAGS:
1247                                                str = conv_dyn_flag(
1248                                                    p_dyn.d_un.d_val,
1249                                                    DUMP_CONVFMT,
1250                                                    &conv_buf.dyn_flag);
1251                                                break;
```



```
1252                                        case DT_FEATURE_1:
1253                                                str = conv_dyn_feature1(
1254                                                    p_dyn.d_un.d_val,
1255                                                    DUMP_CONVFMT,
1256                                                    &conv_buf.dyn_feature1);
1257                                                break;
1258                                        case DT_POSFLAG_1:
1259                                                str = conv_dyn_posflag1(
1260                                                    p_dyn.d_un.d_val,
1261                                                    DUMP_CONVFMT,
1262                                                    &conv_buf.dyn_posflag1);
1263                                                break;
1264                                        case DT_FLAGS_1:
1265                                                str = conv_dyn_flag1(
1266                                                    p_dyn.d_un.d_val, 0,
1267                                                    &conv_buf.dyn_flag1);
1268                                                break;
1269                                        case DT_SUNW_LDMACH:
1270                                                str = conv_ehdr_mach(
1271                                                    p_dyn.d_un.d_val, 0,
1272                                                    &conv_buf.inv);
1273                                                break;
1274                                        }
1275                                }
1276                                if (str) {      /* Show as string */
1277                                        (void) printf("%s", str);
1278                                } else {        /* Numeric form */
1279                                        (void) printf(pdyn_Fmtptr,
1280                                            EC_ADDR(p_dyn.d_un.d_ptr));
1281                                }
1282                                break;

1284                        /*
1285                         * Depreciated items with a literal value
1286                         */
1287                        case DT_DEPRECATED_SPARC_REGISTER:
1288                                (void) printf(pdyn_Fmtptr
1289                                    "  (deprecated value)",
1290                                    EC_XWORD(p_dyn.d_un.d_val));
1291                                break;

1293                        /* Ignored items */
1294                        case DT_SYMBOLIC:
1295                                (void) printf("(ignored)");
1296                                break;
1297                        }
1298                        (void) printf("\n");
1299                        (void) gelf_getdyn(dyn_data, ii++, &p_dyn);
1300                }
1301        }

1303        /*
1304         * Check for existence of static shared library information.
1305         */
1306        while (header_num < p_ehdr.e_phnum) {
1307                (void) gelf_getphdr(elf_file, header_num, &p_phdr);
1308                if (p_phdr.p_type == PT_SHLIB) {
1309                        while (--lib_scns > 0) {
1310                                if (strcmp(l_scns->scn_name, ".lib") == 0) {
1311                                        print_static(l_scns, filename);
1312                                }
1313                                l_scns++;
1314                        }
1315                }
1316                header_num++;
1317        }
```

```
#undef  pdyn_Fmtptr
}

/*
 * Print the ELF header.  Input is an ELF file descriptor
 * and the filename.  If f_flag is set, the ELF header is
 * printed to stdout, otherwise the function returns after
 * setting the pointer to the ELF header.  Any values which
 * are not known are printed in decimal.  Fields must be updated
 * as new values are added.
 */
static GElf_Ehdr *
dump_elf_header(Elf *elf_file, char *filename, GElf_Ehdr * elf_head_p)
{
        int class;
        int field;

        if (gelf_getehdr(elf_file, elf_head_p) == NULL) {
                (void) fprintf(stderr, "%s: %s: %s\n", prog_name, filename,
                    elf_errmsg(-1));
                return (NULL);
        }

        class = (int)elf_head_p->e_ident[4];

        if (class == ELFCLASS64)
                field = 21;
        else
                field = 13;

        if (!f_flag)
                return (elf_head_p);

        if (!p_flag) {
                (void) printf("\n                    **** ELF HEADER ****\n");
                (void) printf("%-*s%-11s%-*sMachine     Version\n",
                    field, "Class", "Data", field, "Type");
                (void) printf("%-*s%-11s%-*sFlags       Ehsize\n",
                    field, "Entry", "Phoff", field, "Shoff");
                (void) printf("%-*s%-11s%-*sShnum       Shstrndx\n\n",
                    field, "Phentsize", "Phnum", field, "Shentsz");
        }

        if (!v_flag) {
                (void) printf("%-*d%-11d%-*d%-12d%d\n",
                    field, elf_head_p->e_ident[4], elf_head_p->e_ident[5],
                    field, (int)elf_head_p->e_type, (int)elf_head_p->e_machine,
                    elf_head_p->e_version);
        } else {
                Conv_inv_buf_t  inv_buf;

                (void) printf("%-*s", field,
                    conv_ehdr_class(class, DUMP_CONVFMT, &inv_buf));
                (void) printf("%-11s",
                    conv_ehdr_data(elf_head_p->e_ident[5], DUMP_CONVFMT,
                    &inv_buf));
                (void) printf("%-*s", field,
                    conv_ehdr_type(elf_head_p->e_ident[EI_OSABI],
                    elf_head_p->e_type, DUMP_CONVFMT, &inv_buf));
                (void) printf("%-12s",
                    conv_ehdr_mach(elf_head_p->e_machine, DUMP_CONVFMT,
                    &inv_buf));
                (void) printf("%s\n",
                    conv_ehdr_vers(elf_head_p->e_version, DUMP_CONVFMT,
                    &inv_buf));
        }
```



```
        (void) printf("%-#*llx%-#11llx%-#*llx%-#12x%#x\n",
            field, EC_ADDR(elf_head_p->e_entry), EC_OFF(elf_head_p->e_phoff),
            field, EC_OFF(elf_head_p->e_shoff), EC_WORD(elf_head_p->e_flags),
            EC_WORD(elf_head_p->e_ehsize));
        if (!v_flag || (elf_head_p->e_shstrndx != SHN_XINDEX)) {
                (void) printf("%-#*x%-11u%-#*x%-12u%u\n",
                    field, EC_WORD(elf_head_p->e_phentsize),
                    EC_WORD(elf_head_p->e_phnum),
                    field, EC_WORD(elf_head_p->e_shentsize),
                    EC_WORD(elf_head_p->e_shnum),
                    EC_WORD(elf_head_p->e_shstrndx));
        } else {
                (void) printf("%-#*x%-11u%-#*x%-12uXINDEX\n",
                    field, EC_WORD(elf_head_p->e_phentsize),
                    EC_WORD(elf_head_p->e_phnum),
                    field, EC_WORD(elf_head_p->e_shentsize),
                    EC_WORD(elf_head_p->e_shnum));
        }
        if ((elf_head_p->e_shnum == 0) && (elf_head_p->e_shoff > 0)) {
                Elf_Scn         *scn;
                GElf_Shdr       shdr0;
                int             field;

                if (gelf_getclass(elf_file) == ELFCLASS64)
                        field = 21;
                else
                        field = 13;
                if (!p_flag) {
                        (void) printf("\n          **** SECTION HEADER[0] "
                            "{Elf Extensions} ****\n");
                        (void) printf(
                            "[No]\tType\tFlags\t%-*s %-*s%-*s%sName\n",
                            field, "Addr", field, "Offset", field,
                            "Size(shnum)",
                            /* compatibility:  tab for elf32 */
                            (field == 13) ? "\t" : "  ");
                        (void) printf("\tLn(strndx) Info\t%-*s Entsize\n",
                            field, "Adralgn");
                }
                if ((scn = elf_getscn(elf_file, 0)) == NULL) {
                        (void) fprintf(stderr,
                            "%s: %s: elf_getscn failed: %s\n",
                            prog_name, filename, elf_errmsg(-1));
                        return (NULL);
                }
                if (gelf_getshdr(scn, &shdr0) == 0) {
                        (void) fprintf(stderr,
                            "%s: %s: gelf_getshdr: %s\n",
                            prog_name, filename, elf_errmsg(-1));
                        return (NULL);
                }
                (void) printf("[0]\t%u\t%llu\t", EC_WORD(shdr0.sh_type),
                    EC_XWORD(shdr0.sh_flags));

                (void) printf("%-#*llx %-#*llx%-*llu%s%-*u\n",
                    field, EC_ADDR(shdr0.sh_addr),
                    field, EC_OFF(shdr0.sh_offset),
                    field, EC_XWORD(shdr0.sh_size),
                    /* compatibility:  tab for elf32 */
                    ((field == 13) ? "\t" : "  "),
                    field, EC_WORD(shdr0.sh_name));

                (void) printf("\t%u\t%u\t%-#*llx %-#*llx\n",
                    EC_WORD(shdr0.sh_link),
                    EC_WORD(shdr0.sh_info),
                    field, EC_XWORD(shdr0.sh_addralign),
```

```
1450                     field, EC_XWORD(shdr0.sh_entsize));
1451         }
1452         (void) printf("\n");

1454         return (elf_head_p);
1455 }

1457 /*
1458  * Print section contents.  Input is an ELF file descriptor,
1459  * the ELF header, the SCNTAB structure,
1460  * the number of symbols, and the filename.
1461  * The number of sections,
1462  * and the offset into the SCNTAB structure will be
1463  * set in dump_section if d_flag or n_flag are set.
1464  * If v_flag is set, sections which can be interpreted will
1465  * be interpreted, otherwise raw data will be output in hexidecimal.
1466  */
1467 static void
1468 print_section(Elf *elf_file,
1469         GElf_Ehdr *p_ehdr, SCNTAB *p, int num_scns, char *filename)
1470 {
1471         unsigned char    *p_sec;
1472         int     i;
1473         size_t  size;

1475         for (i = 0; i < num_scns; i++, p++) {
1476                 GElf_Shdr shdr;

1478                 size = 0;
1479                 if (s_flag && !v_flag)
1480                         p_sec = (unsigned char *)get_rawscn(p->p_sd, &size);
1481                 else
1482                         p_sec = (unsigned char *)get_scndata(p->p_sd, &size);

1484                 if ((gelf_getshdr(p->p_sd, &shdr) != NULL) &&
1485                     (shdr.sh_type == SHT_NOBITS)) {
1486                         continue;
1487                 }
1488                 if (s_flag && !v_flag) {
1489                         (void) printf("\n%s:\n", p->scn_name);
1490                         print_rawdata(p_sec, size);
1491                         continue;
1492                 }
1493                 if (shdr.sh_type == SHT_SYMTAB) {
1494                         dump_symbol_table(elf_file, p, filename);
1495                         continue;
1496                 }
1497                 if (shdr.sh_type == SHT_DYNSYM) {
1498                         dump_symbol_table(elf_file, p, filename);
1499                         continue;
1500                 }
1501                 if (shdr.sh_type == SHT_STRTAB) {
1502                         dump_string_table(p, 1);
1503                         continue;
1504                 }
1505                 if (shdr.sh_type == SHT_RELA) {
1506                         dump_reloc_table(elf_file, p_ehdr, p, 1, filename);
1507                         continue;
1508                 }
1509                 if (shdr.sh_type == SHT_REL) {
1510                         dump_reloc_table(elf_file, p_ehdr, p, 1, filename);
1511                         continue;
1512                 }
1513                 if (shdr.sh_type == SHT_DYNAMIC) {
1514                         dump_dynamic(elf_file, p, 1, filename);
1515                         continue;
```



```
1516                 }

1518                 (void) printf("\n%s:\n", p->scn_name);
1519                 print_rawdata(p_sec, size);
1520         }
1521         (void) printf("\n");
1522 }

1524 /*
1525  * Print section contents. This function does not print the contents
1526  * of the sections but sets up the parameters and then calls
1527  * print_section to print the contents.  Calling another function to print
1528  * the contents allows both -d and -n to work correctly
1529  * simultaneously. Input is an ELF file descriptor, the ELF header,
1530  * the SCNTAB structure, the number of sections, and the filename.
1531  * Set the range of sections if d_flag, and set section name if
1532  * n_flag.
1533  */
1534 static void
1535 dump_section(Elf *elf_file,
1536         GElf_Ehdr *p_ehdr, SCNTAB *s, int num_scns, char *filename)
1537 {
1538         SCNTAB *n_range, *d_range; /* for use with -n and -d modifiers */
1539         int i;
1540         int found_it = 0;  /* for use with -n section_name */

1542         if (n_flag) {
1543                 n_range = s;

1545                 for (i = 0; i < num_scns; i++, n_range++) {
1546                         if ((strcmp(name, n_range->scn_name)) != 0)
1547                                 continue;
1548                         else {
1549                                 found_it = 1;
1550                                 print_section(elf_file, p_ehdr,
1551                                     n_range, 1, filename);
1552                         }
1553                 }

1555                 if (!found_it) {
1556                         (void) fprintf(stderr, "%s: %s: %s not found\n",
1557                             prog_name, filename, name);
1558                 }
1559         } /* end n_flag */

1561         if (d_flag) {
1562                 d_range = s;
1563                 d_num = check_range(d_low, d_hi, num_scns, filename);
1564                 if (d_num < 0)
1565                         return;
1566                 d_range += d_low - 1;

1568                 print_section(elf_file, p_ehdr, d_range, d_num, filename);
1569         }       /* end d_flag */

1571         if (!n_flag && !d_flag)
1572                 print_section(elf_file, p_ehdr, s, num_scns, filename);
1573 }

1575 /*
1576  * Print the section header table. This function does not print the contents
1577  * of the section headers but sets up the parameters and then calls
1578  * print_shdr to print the contents.  Calling another function to print
1579  * the contents allows both -d and -n to work correctly
1580  * simultaneously.  Input is the SCNTAB structure,
1581  * the number of sections from the ELF header, and the filename.
```

```
 * Set the range of section headers to print if d_flag, and set
 * name of section header to print if n_flag.
 */
static void
dump_shdr(Elf *elf_file, SCNTAB *s, int num_scns, char *filename)
{

        SCNTAB *n_range, *d_range;      /* for use with -n and -d modifiers */
        int field;
        int i;
        int found_it = 0;  /* for use with -n section_name */

        if (gelf_getclass(elf_file) == ELFCLASS64)
                field = 21;
        else
                field = 13;

        if (!p_flag) {
                (void) printf("\n          **** SECTION HEADER TABLE ****\n");
                (void) printf("[No]\tType\tFlags\t%-*s %-*s %-*s%sName\n",
                    field, "Addr", field, "Offset", field, "Size",
                    /* compatibility:  tab for elf32 */
                    (field == 13) ? "\t" : "  ");
                (void) printf("\tLink\tInfo\t%-*s Entsize\n\n",
                    field, "Adralgn");
        }

        if (n_flag) {
                n_range = s;

                for (i = 1; i <= num_scns; i++, n_range++) {
                        if ((strcmp(name, n_range->scn_name)) != 0)
                                continue;
                        else {
                                found_it = 1;
                                print_shdr(elf_file, n_range, 1, i);
                        }
                }

                if (!found_it) {
                        (void) fprintf(stderr, "%s: %s: %s not found\n",
                            prog_name, filename, name);
                }
        } /* end n_flag */

        if (d_flag) {
                d_range = s;
                d_num = check_range(d_low, d_hi, num_scns, filename);
                if (d_num < 0)
                        return;
                d_range += d_low - 1;

                print_shdr(elf_file, d_range, d_num, d_low);
        }       /* end d_flag */

        if (!n_flag && !d_flag)
                print_shdr(elf_file, s, num_scns, 1);
}

/*
 * Process all of the command line options (except
 * for -a, -g, -f, and -o).  All of the options processed
 * by this function require the presence of the section
 * header table and will not be processed if it is not present.
 * Set up a buffer containing section name, section header,
 * and section descriptor for each section in the file.  This
```



```
 * structure is used to avoid duplicate calls to libelf functions.
 * Structure members for the symbol table, the debugging information,
 * and the line number information are global.  All of the
 * rest are local.
 */
static void
dump_section_table(Elf *elf_file, GElf_Ehdr *elf_head_p, char *filename)
{

        static SCNTAB   *buffer, *p_scns;
        Elf_Scn         *scn = 0;
        char            *s_name = NULL;
        int             found = 0;
        unsigned int    num_scns;
        size_t          shstrndx;
        size_t          shnum;


        if (elf_getshdrnum(elf_file, &shnum) == -1) {
                (void) fprintf(stderr,
                    "%s: %s: elf_getshdrnum failed: %s\n",
                    prog_name, filename, elf_errmsg(-1));
                return;
        }
        if (elf_getshdrstrndx(elf_file, &shstrndx) == -1) {
                (void) fprintf(stderr,
                    "%s: %s: elf_getshdrstrndx failed: %s\n",
                    prog_name, filename, elf_errmsg(-1));
                return;
        }

        if ((buffer = calloc(shnum, sizeof (SCNTAB))) == NULL) {
                (void) fprintf(stderr, "%s: %s: cannot calloc space\n",
                    prog_name, filename);
                return;
        }
        /* LINTED */
        num_scns = (int)shnum - 1;

        p_symtab = (SCNTAB *)0;
        p_dynsym = (SCNTAB *)0;
        p_scns = buffer;
        p_head_scns = buffer;

        while ((scn = elf_nextscn(elf_file, scn)) != 0) {
                if ((gelf_getshdr(scn, &buffer->p_shdr)) == 0) {
                        (void) fprintf(stderr,
                            "%s: %s: %s\n", prog_name, filename,
                            elf_errmsg(-1));
                        return;
                }
                s_name = (char *)
                    elf_strptr(elf_file, shstrndx, buffer->p_shdr.sh_name);
                buffer->scn_name = s_name ? s_name : (char *)UNKNOWN;
                buffer->p_sd   =  scn;

                if (buffer->p_shdr.sh_type == SHT_SYMTAB) {
                        found += 1;
                        p_symtab = buffer;
                }
                if (buffer->p_shdr.sh_type == SHT_DYNSYM)
                        p_dynsym = buffer;
                buffer++;
        }

        /*
```

```
1714          * These functions depend upon the presence of the section header table
1715          * and will not be invoked in its absence
1716          */
1717         if (h_flag) {
1718                 dump_shdr(elf_file, p_scns, num_scns, filename);
1719         }
1720         if (p_symtab && (t_flag || T_flag)) {
1721                 dump_symbol_table(elf_file, p_symtab, filename);
1722         }
1723         if (c_flag) {
1724                 dump_string_table(p_scns, num_scns);
1725         }
1726         if (r_flag) {
1727                 dump_reloc_table(elf_file, elf_head_p,
1728                     p_scns, num_scns, filename);
1729         }
1730         if (L_flag) {
1731                 dump_dynamic(elf_file, p_scns, num_scns, filename);
1732         }
1733         if (s_flag) {
1734                 dump_section(elf_file, elf_head_p, p_scns,
1735                     num_scns, filename);
1736         }
1737 }

1739 /*
1740  * Load the archive string table(s) (for extended-length strings)
1741  * into an in-core table/list
1742  */
1743 static struct stab_list_s *
1744 load_arstring_table(struct stab_list_s *STabList,
1745         int fd, Elf *elf_file, Elf_Arhdr *p_ar, char *filename)
1746 {
1747         off_t here;
1748         struct stab_list_s *STL_entry, *STL_next;

1750         if (p_ar) {
1751                 STL_entry = malloc(sizeof (struct stab_list_s));
1752                 STL_entry->next    = 0;
1753                 STL_entry->strings = 0;
1754                 STL_entry->size    = 0;

1756                 if (!STabList)
1757                         STabList = STL_entry;
1758                 else {
1759                         STL_next = STabList;
1760                         while (STL_next->next != (void *)0)
1761                                 STL_next = STL_next->next;
1762                         STL_next->next = STL_entry;
1763                 }

1765                 STL_entry->size    = p_ar->ar_size;
1766                 STL_entry->strings = malloc(p_ar->ar_size);
1767                 here = elf_getbase(elf_file);
1768                 if ((lseek(fd, here, 0)) != here) {
1769                         (void) fprintf(stderr,
1770                             "%s: %s: could not lseek\n", prog_name, filename);
1771                 }

1773                 if ((read(fd, STL_entry->strings, p_ar->ar_size)) == -1) {
1774                         (void) fprintf(stderr,
1775                             "%s: %s: could not read\n", prog_name, filename);
1776                 }
1777         }
1778         return (STabList);
1779 }
```



```
1781 /*
1782  * Print the archive header for each member of an archive.
1783  * Also call ar_sym_read to print the symbols in the
1784  * archive symbol table if g_flag.  Input is a file descriptor,
1785  * an ELF file descriptor, and the filename.  Putting the call
1786  * to dump the archive symbol table in this function is more
1787  * efficient since it is necessary to examine the archive member
1788  * name in the archive header to determine which member is the
1789  * symbol table.
1790  */
1791 static void
1792 dump_ar_hdr(int fd, Elf *elf_file, char *filename)
1793 {
1794         extern int v_flag, g_flag, a_flag, p_flag;
1795         Elf_Arhdr  *p_ar;
1796         Elf *arf;
1797         Elf_Cmd cmd;
1798         int title = 0;
1799         int err = 0;

1801         char buf[DATESIZE];

1803         cmd = ELF_C_READ;
1804         while ((arf = elf_begin(fd, cmd, elf_file)) != 0) {
1805                 p_ar = elf_getarhdr(arf);
1806                 if (p_ar == NULL) {
1807                         (void) fprintf(stderr,
1808                             "%s: %s: %s\n", prog_name, filename,
1809                             elf_errmsg(-1));
1810                         continue;
1811                 }
1812                 if ((strcmp(p_ar->ar_name, "/") == 0) ||
1813                     (strcmp(p_ar->ar_name, "/SYM64/") == 0)) {
1814                         if (g_flag)
1815                                 ar_sym_read(elf_file, filename);
1816                 } else if (strcmp(p_ar->ar_name, "//") == 0) {
1817                         StringTableList = load_arstring_table(
1818                             StringTableList, fd, arf, p_ar, filename);
1819                         cmd = elf_next(arf);
1820                         (void) elf_end(arf);
1821                         continue;
1822                 } else {
1823                         if (a_flag) {
1824                                 (void) printf("%s[%s]:\n", filename,
1825                                     p_ar->ar_name);
1826                                 if (!p_flag && title == 0) {
1827                                         if (!v_flag)
1828                                                 (void) printf(
1829 "\n\n\t\t\t***ARCHIVE HEADER***"
1830 "\n     Date          Uid     Gid    Mode      Size      Member Name\n\n");
1831                                         else
1832                                                 (void) printf(
1833 "\n\n\t\t\t***ARCHIVE HEADER***"
1834 "\n     Date                   Uid    Gid   Mode     Size     Member Name\n\n");
1835                                         title = 1;
1836                                 }
1837                                 if (!v_flag) {
1838                                         (void) printf(
1839 "\t0x%.8lx  %6d  %6d  0%.6ho  0x%.8lx  %-s\n\n",
1840                                             p_ar->ar_date, (int)p_ar->ar_uid,
1841                                             (int)p_ar->ar_gid,
1842                                             (int)p_ar->ar_mode,
1843                                             p_ar->ar_size, p_ar->ar_name);
1844                                 } else {
1845                                         if ((strftime(buf, DATESIZE,
```

```
1846                                            "%b %d %H:%M:%S %Y",
1847                                            localtime(
1848                                            &(p_ar->ar_date)))) == 0) {
1849                                                (void) fprintf(stderr,
1850 "%s: %s: don't have enough space to store the date\n", prog_name, filename);
1851                                                exit(1);
1852                                        }
1853                                        (void) printf(
1854 "\t%s %6d %6d 0%.6ho 0x%.8lx %-s\n\n",
1855                                            buf, (int)p_ar->ar_uid,
1856                                            (int)p_ar->ar_gid,
1857                                            (int)p_ar->ar_mode,
1858                                            p_ar->ar_size, p_ar->ar_name);
1859                                    }
1860                            }
1861                    }
1862                    cmd = elf_next(arf);
1863                    (void) elf_end(arf);
1864            } /* end while */

1866            err = elf_errno();
1867            if (err != 0) {
1868                    (void) fprintf(stderr,
1869                        "%s: %s: %s\n", prog_name, filename, elf_errmsg(err));
1870            }
1871    }

1873    /*
1874     * Process member files of an archive.  This function provides
1875     * a loop through an archive equivalent the processing of
1876     * each_file for individual object files.
1877     */
1878    static void
1879    dump_ar_files(int fd, Elf *elf_file, char *filename)
1880    {
1881            Elf_Arhdr  *p_ar;
1882            Elf *arf;
1883            Elf_Cmd cmd;
1884            Elf_Kind file_type;
1885            GElf_Ehdr elf_head;
1886            char *fullname;

1888            cmd = ELF_C_READ;
1889            while ((arf = elf_begin(fd, cmd, elf_file)) != 0) {
1890                    size_t  len;

1892                    p_ar = elf_getarhdr(arf);
1893                    if (p_ar == NULL) {
1894                            (void) fprintf(stderr, "%s: %s: %s\n",
1895                                prog_name, filename, elf_errmsg(-1));
1896                            return;
1897                    }
1898                    if (p_ar->ar_name[0] == '/') {
1899                            cmd = elf_next(arf);
1900                            (void) elf_end(arf);
1901                            continue;
1902                    }

1904                    len = strlen(filename) + strlen(p_ar->ar_name) + 3;
1905                    if ((fullname = malloc(len)) == NULL)
1906                            return;
1907                    (void) snprintf(fullname, len, "%s[%s]", filename,
1908                        p_ar->ar_name);
1909                    (void) printf("\n%s:\n", fullname);
1910                    file_type = elf_kind(arf);
1911                    if (file_type == ELF_K_ELF) {
```



```
1912                            if (dump_elf_header(arf, fullname, &elf_head) == NULL)
1913                                    return;
1914                            if (o_flag)
1915                                    dump_exec_header(arf,
1916                                        (unsigned)elf_head.e_phnum, fullname);
1917                            if (x_flag)
1918                                    dump_section_table(arf, &elf_head, fullname);
1919                    } else {
1920                            (void) fprintf(stderr, "%s: %s: invalid file type\n",
1921                                prog_name, fullname);
1922                            cmd = elf_next(arf);
1923                            (void) elf_end(arf);
1924                            continue;
1925                    }

1927                    cmd = elf_next(arf);
1928                    (void) elf_end(arf);
1929            } /* end while */
1930    }

1932    /*
1933     * Takes a filename as input.  Test first for a valid version
1934     * of libelf.a and exit on error.  Process each valid file
1935     * or archive given as input on the command line.  Check
1936     * for file type.  If it is an archive, process the archive-
1937     * specific options first, then files within the archive.
1938     * If it is an ELF object file, process it; otherwise
1939     * warn that it is an invalid file type.
1940     * All options except the archive-specific and program
1941     * execution header are processed in the function, dump_section_table.
1942     */
1943    static void
1944    each_file(char *filename)
1945    {
1946            Elf *elf_file;
1947            GElf_Ehdr elf_head;
1948            int fd;
1949            Elf_Kind   file_type;

1951            struct stat buf;

1953            Elf_Cmd cmd;
1954            errno = 0;

1956            if (stat(filename, &buf) == -1) {
1957                    int     err = errno;
1958                    (void) fprintf(stderr, "%s: %s: %s", prog_name, filename,
1959                        strerror(err));
1960                    return;
1961            }

1963            if ((fd = open((filename), O_RDONLY)) == -1) {
1964                    (void) fprintf(stderr, "%s: %s: cannot read\n", prog_name,
1965                        filename);
1966                    return;
1967            }
1968            cmd = ELF_C_READ;
1969            if ((elf_file = elf_begin(fd, cmd, (Elf *)0)) == NULL) {
1970                    (void) fprintf(stderr, "%s: %s: %s\n", prog_name, filename,
1971                        elf_errmsg(-1));
1972                    return;
1973            }

1975            file_type = elf_kind(elf_file);
1976            if (file_type == ELF_K_AR) {
1977                    if (a_flag || g_flag) {
```

```
1978                        dump_ar_hdr(fd, elf_file, filename);
1979                        elf_file = elf_begin(fd, cmd, (Elf *)0);
1980                }
1981                if (z_flag)
1982                        dump_ar_files(fd, elf_file, filename);
1983        } else {
1984                if (file_type == ELF_K_ELF) {
1985                        (void) printf("\n%s:\n", filename);
1986                        if (dump_elf_header(elf_file, filename, &elf_head)) {
1987                                if (o_flag)
1988                                        dump_exec_header(elf_file,
1989                                            (unsigned)elf_head.e_phnum,
1990                                            filename);
1991                                if (x_flag)
1992                                        dump_section_table(elf_file,
1993                                            &elf_head, filename);
1994                        }
1995                } else {
1996                        (void) fprintf(stderr, "%s: %s: invalid file type\n",
1997                            prog_name, filename);
1998                }
1999        }
2000        (void) elf_end(elf_file);
2001        (void) close(fd);
2002 }

2004 /*
2005  * Sets up flags for command line options given and then
2006  * calls each_file() to process each file.
2007  */
2008 int
2009 main(int argc, char *argv[], char *envp[])
2010 {
2011        char *optstr = OPTSTR; /* option string used by getopt() */
2012        int optchar;

2014        /*
2015         * Check for a binary that better fits this architecture.
2016         */
2017        (void) conv_check_native(argv, envp);

2019        prog_name = argv[0];

2021        (void) setlocale(LC_ALL, "");
2022        while ((optchar = getopt(argc, argv, optstr)) != -1) {
2023                switch (optchar) {
2024                case 'a':
2025                        a_flag = 1;
2026                        x_flag = 1;
2027                        break;
2028                case 'g':
2029                        g_flag = 1;
2030                        x_flag = 1;
2031                        break;
2032                case 'v':
2033                        v_flag = 1;
2034                        break;
2035                case 'p':
2036                        p_flag = 1;
2037                        break;
2038                case 'f':
2039                        f_flag = 1;
2040                        z_flag = 1;
2041                        break;
2042                case 'o':
2043                        o_flag = 1;
```



```
2044                        z_flag = 1;
2045                        break;
2046                case 'h':
2047                        h_flag = 1;
2048                        x_flag = 1;
2049                        z_flag = 1;
2050                        break;
2051                case 's':
2052                        s_flag = 1;
2053                        x_flag = 1;
2054                        z_flag = 1;
2055                        break;
2056                case 'd':
2057                        d_flag = 1;
2058                        x_flag = 1;
2059                        z_flag = 1;
2060                        set_range(optarg, &d_low, &d_hi);
2061                        break;
2062                case 'n':
2063                        n_flag++;
2064                        x_flag = 1;
2065                        z_flag = 1;
2066                        name = optarg;
2067                        break;
2068                case 'r':
2069                        r_flag = 1;
2070                        x_flag = 1;
2071                        z_flag = 1;
2072                        break;
2073                case 't':
2074                        t_flag = 1;
2075                        x_flag = 1;
2076                        z_flag = 1;
2077                        break;
2078                case 'C':
2079                        C_flag = 1;
2080                        t_flag = 1;
2081                        x_flag = 1;
2082                        z_flag = 1;
2083                        break;
2084                case 'T':
2085                        T_flag = 1;
2086                        x_flag = 1;
2087                        z_flag = 1;
2088                        set_range(optarg, &T_low, &T_hi);
2089                        break;
2090                case 'c':
2091                        c_flag = 1;
2092                        x_flag = 1;
2093                        z_flag = 1;
2094                        break;
2095                case 'L':
2096                        L_flag = 1;
2097                        x_flag = 1;
2098                        z_flag = 1;
2099                        break;
2100                case 'V':
2101                        V_flag = 1;
2102                        (void) fprintf(stderr, "dump: %s %s\n",
2103                            (const char *)SGU_PKG,
2104                            (const char *)SGU_REL);
2105                        break;
2106                case '?':
2107                        errflag += 1;
2108                        break;
2109                default:
```

```
2110                            break;
2111                    }
2112            }

2114            if (errflag || (optind >= argc) || (!z_flag && !x_flag)) {
2115                    if (!(V_flag && (argc == 2))) {
2116                            usage();
2117                            exit(269);
2118                    }
2119            }

2121            if (elf_version(EV_CURRENT) == EV_NONE) {
2122                    (void) fprintf(stderr, "%s: libelf is out of date\n",
2123                        prog_name);
2124                    exit(101);
2125            }

2127            while (optind < argc) {
2128                    each_file(argv[optind]);
2129                    optind++;
2130            }
2131            return (0);
2132    }
```

```
**********************************************************
   56411 Wed May 27 19:48:58 2015
new/usr/src/cmd/sgs/elfdump/common/corenote.c
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
______unchanged_portion_omitted_


 427 /*
 428  * Output information from auxv_t structure.
 429  */
 430 static void
 431 dump_auxv(note_state_t *state, const char *title)
 432 {
 433         const sl_auxv_layout_t  *layout = state->ns_arch->auxv;
 434         union {
 435                 Conv_cap_val_hw1_buf_t          hw1;
 436                 Conv_cap_val_hw2_buf_t          hw2;
 437                 Conv_cnote_auxv_af_buf_t        auxv_af;
 438                 Conv_ehdr_flags_buf_t           ehdr_flags;
 439                 Conv_secflags_buf_t             secflags;
 440 #endif /* ! codereview */
 441                 Conv_inv_buf_t                  inv;
 442         } conv_buf;
 443         sl_fmtbuf_t     buf;
 444         int             ndx, ndx_start;
 445         Word            sizeof_auxv;

 447         sizeof_auxv = layout->sizeof_struct.slf_eltlen;

 449         indent_enter(state, title, &layout->sizeof_struct);

 451         /*
 452          * Immediate indent_exit() restores the indent level to
 453          * that of the title. We include indentation as part of
 454          * the index string, which is right justified, and don't
 455          * want the usual indentation spacing.
 456          */
 457         indent_exit(state);

 459         ndx = 0;
 460         while (state->ns_len > sizeof_auxv) {
 461                 char            index[(MAXNDXSIZE * 2) + 1];
 462                 sl_fmt_num_t    num_fmt = SL_FMT_NUM_ZHEX;
 463                 const char      *vstr = NULL;
 464                 Word            w;
 465                 int             type;
 466                 sl_field_t      a_type_next;

 468                 type = extract_as_word(state, &layout->a_type);
 469                 ndx_start = ndx;
 470                 switch (type) {
 471                 case AT_NULL:
 472                         a_type_next = layout->a_type;
 473                         a_type_next.slf_offset += sizeof_auxv;
 474                         while ((state->ns_len - sizeof_auxv) >= sizeof_auxv) {
 475                                 type = extract_as_word(state, &a_type_next);
 476                                 if (type != AT_NULL)
 477                                         break;
 478                                 ndx++;
 479                                 state->ns_data += sizeof_auxv;
 480                                 state->ns_len -= sizeof_auxv;
 481                         }
```



```
 482                         num_fmt = SL_FMT_NUM_HEX;
 483                         break;



 487                 case AT_IGNORE:
 488                 case AT_SUN_IFLUSH:
 489                         num_fmt = SL_FMT_NUM_HEX;
 490                         break;

 492                 case AT_SUN_SECFLAGS:
 493                         w = extract_as_word(state, &layout->a_val);
 494                         vstr = conv_psecflags(w, 0, &conv_buf.secflags);
 495                         break;

 497 #endif /* ! codereview */
 498                 case AT_EXECFD:
 499                 case AT_PHENT:
 500                 case AT_PHNUM:
 501                 case AT_PAGESZ:
 502                 case AT_SUN_UID:
 503                 case AT_SUN_RUID:
 504                 case AT_SUN_GID:
 505                 case AT_SUN_RGID:
 506                 case AT_SUN_LPAGESZ:
 507                         num_fmt = SL_FMT_NUM_DEC;
 508                         break;

 510                 case AT_FLAGS:  /* processor flags */
 511                         w = extract_as_word(state, &layout->a_val);
 512                         vstr = conv_ehdr_flags(state->ns_mach, w,
 513                             0, &conv_buf.ehdr_flags);
 514                         break;

 516                 case AT_SUN_HWCAP:
 517                         w = extract_as_word(state, &layout->a_val);
 518                         vstr = conv_cap_val_hw1(w, state->ns_mach,
 519                             0, &conv_buf.hw1);
 520                         /*
 521                          * conv_cap_val_hw1() produces output like:
 522                          *
 523                          *      0xfff [ flg1 flg2 0xff]
 524                          *
 525                          * where the first hex value is the complete value,
 526                          * and the second is the leftover bits. We only
 527                          * want the part in brackets, and failing that,
 528                          * would rather fall back to formatting the full
 529                          * value ourselves.
 530                          */
 531                         while ((*vstr != '\0') && (*vstr != '['))
 532                                 vstr++;
 533                         if (*vstr != '[')
 534                                 vstr = NULL;
 535                         num_fmt = SL_FMT_NUM_HEX;
 536                         break;
 537                 case AT_SUN_HWCAP2:
 538                         w = extract_as_word(state, &layout->a_val);
 539                         vstr = conv_cap_val_hw2(w, state->ns_mach,
 540                             0, &conv_buf.hw2);
 541                         /*
 542                          * conv_cap_val_hw2() produces output like:
 543                          *
 544                          *      0xfff [ flg1 flg2 0xff]
 545                          *
 546                          * where the first hex value is the complete value,
 547                          * and the second is the leftover bits. We only
```

```
548                          * want the part in brackets, and failing that,
549                          * would rather fall back to formatting the full
550                          * value ourselves.
551                          */
552                         while ((*vstr != '\0') && (*vstr != '['))
553                                 vstr++;
554                         if (*vstr != '[')
555                                 vstr = NULL;
556                         num_fmt = SL_FMT_NUM_HEX;
557                         break;



561                 case AT_SUN_AUXFLAGS:
562                         w = extract_as_word(state, &layout->a_val);
563                         vstr = conv_cnote_auxv_af(w, 0, &conv_buf.auxv_af);
564                         num_fmt = SL_FMT_NUM_HEX;
565                         break;
566                 }

568                 if (ndx == ndx_start)
569                         (void) snprintf(index, sizeof (index),
570                             MSG_ORIG(MSG_FMT_INDEX2), EC_WORD(ndx));
571                 else
572                         (void) snprintf(index, sizeof (index),
573                             MSG_ORIG(MSG_FMT_INDEXRNG),
574                             EC_WORD(ndx_start), EC_WORD(ndx));

576                 if (vstr == NULL)
577                         vstr = fmt_num(state, &layout->a_val, num_fmt, buf);
578                 dbg_print(0, MSG_ORIG(MSG_CNOTE_FMT_AUXVLINE), INDENT, index,
579                     state->ns_vcol - state->ns_indent,
580                     conv_cnote_auxv_type(type, CONV_FMT_DECIMAL,
581                     &conv_buf.inv), vstr);

583                 state->ns_data += sizeof_auxv;
584                 state->ns_len -= sizeof_auxv;
585                 ndx++;
586         }
587 }


590 /*
591  * Output information from fltset_t structure.
592  */
593 static void
594 dump_fltset(note_state_t *state, const char *title)
595 {
596 #define NELTS 4

598         const sl_fltset_layout_t        *layout = state->ns_arch->fltset;
599         Conv_cnote_fltset_buf_t buf;
600         sl_field_t              fdesc;
601         uint32_t                mask[NELTS];
602         int                     i, nelts;

604         if (!data_present(state, &layout->sizeof_struct))
605                 return;

607         fdesc = layout->word;
608         nelts = fdesc.slf_nelts;
609         if (nelts > NELTS)      /* Type has grown? Show what we understand */
610                 nelts = NELTS;
611         for (i = 0; i < nelts; i++) {
612                 mask[i] = extract_as_word(state, &fdesc);
613                 fdesc.slf_offset += fdesc.slf_eltlen;
```



```
614         }

616         print_str(state, title, conv_cnote_fltset(mask, nelts, 0, &buf));

618 #undef NELTS
619 }


622 /*
623  * Output information from sigset_t structure.
624  */
625 static void
626 dump_sigset(note_state_t *state, const char *title)
627 {
628 #define NELTS 4

630         const sl_sigset_layout_t        *layout = state->ns_arch->sigset;
631         Conv_cnote_sigset_buf_t buf;
632         sl_field_t              fdesc;
633         uint32_t                mask[NELTS];
634         int                     i, nelts;

636         if (!data_present(state, &layout->sizeof_struct))
637                 return;

639         fdesc = layout->sigbits;
640         nelts = fdesc.slf_nelts;
641         if (nelts > NELTS)      /* Type has grown? Show what we understand */
642                 nelts = NELTS;
643         for (i = 0; i < nelts; i++) {
644                 mask[i] = extract_as_word(state, &fdesc);
645                 fdesc.slf_offset += fdesc.slf_eltlen;
646         }

648         print_str(state, title, conv_cnote_sigset(mask, nelts, 0, &buf));

650 #undef NELTS
651 }


654 /*
655  * Output information from sigaction structure.
656  */
657 static void
658 dump_sigaction(note_state_t *state, const char *title)
659 {
660         const sl_sigaction_layout_t     *layout = state->ns_arch->sigaction;
661         Conv_cnote_sa_flags_buf_t       conv_buf;
662         Word    w;

664         indent_enter(state, title, &layout->sa_flags);

666         if (data_present(state, &layout->sa_flags)) {
667                 w = extract_as_word(state, &layout->sa_flags);
668                 print_str(state, MSG_ORIG(MSG_CNOTE_T_SA_FLAGS),
669                     conv_cnote_sa_flags(w, 0, &conv_buf));
670         }

672         PRINT_ZHEX_2UP(MSG_ORIG(MSG_CNOTE_T_SA_HANDLER), sa_hand,
673             MSG_ORIG(MSG_CNOTE_T_SA_SIGACTION), sa_sigact);
674         PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_SA_MASK), sa_mask, dump_sigset);

676         indent_exit(state);
677 }
```

```
680 /*
681  * Output information from siginfo structure.
682  */
683 static void
684 dump_siginfo(note_state_t *state, const char *title)
685 {
686         const sl_siginfo_layout_t       *layout = state->ns_arch->siginfo;
687         Conv_inv_buf_t  inv_buf;
688         Word            w;
689         int             v_si_code, v_si_signo;

691         if (!data_present(state, &layout->sizeof_struct))
692                 return;

694         indent_enter(state, title, &layout->f_si_signo);

696         v_si_signo = extract_as_sword(state, &layout->f_si_signo);
697         print_str(state, MSG_ORIG(MSG_CNOTE_T_SI_SIGNO),
698             conv_cnote_signal(v_si_signo, CONV_FMT_DECIMAL, &inv_buf));

700         w = extract_as_word(state, &layout->f_si_errno);
701         print_str(state, MSG_ORIG(MSG_CNOTE_T_SI_ERRNO),
702             conv_cnote_errno(w, CONV_FMT_DECIMAL, &inv_buf));

704         v_si_code = extract_as_sword(state, &layout->f_si_code);
705         print_str(state, MSG_ORIG(MSG_CNOTE_T_SI_CODE),
706             conv_cnote_si_code(state->ns_mach, v_si_signo, v_si_code,
707             CONV_FMT_DECIMAL, &inv_buf));

709         if ((v_si_signo == 0) || (v_si_code == SI_NOINFO)) {
710                 indent_exit(state);
711                 return;
712         }

714         /* User generated signals have (si_code <= 0) */
715         if (v_si_code <= 0) {
716                 PRINT_DEC(MSG_ORIG(MSG_CNOTE_T_SI_PID), f_si_pid);
717                 PRINT_DEC(MSG_ORIG(MSG_CNOTE_T_SI_UID), f_si_uid);
718                 PRINT_DEC(MSG_ORIG(MSG_CNOTE_T_SI_CTID), f_si_ctid);
719                 PRINT_DEC(MSG_ORIG(MSG_CNOTE_T_SI_ZONEID), f_si_zoneid);
720                 switch (v_si_code) {
721                 case SI_QUEUE:
722                 case SI_TIMER:
723                 case SI_ASYNCIO:
724                 case SI_MESGQ:
725                         indent_enter(state, MSG_ORIG(MSG_CNOTE_T_SI_VALUE),
726                             &layout->f_si_value_int);
727                         PRINT_ZHEX(MSG_ORIG(MSG_CNOTE_T_SIVAL_INT),
728                             f_si_value_int);
729                         PRINT_ZHEX(MSG_ORIG(MSG_CNOTE_T_SIVAL_PTR),
730                             f_si_value_ptr);
731                         indent_exit(state);
732                         break;
733                 }
734                 indent_exit(state);
735                 return;
736         }

738         /*
739          * Remaining cases are kernel generated signals. Output any
740          * signal or code specific information.
741          */
742         if (v_si_code == SI_RCTL)
743                 PRINT_HEX(MSG_ORIG(MSG_CNOTE_T_SI_ENTITY), f_si_entity);
744         switch (v_si_signo) {
745         case SIGILL:
```



```
746         case SIGFPE:
747         case SIGSEGV:
748         case SIGBUS:
749                 PRINT_ZHEX(MSG_ORIG(MSG_CNOTE_T_SI_ADDR), f_si_addr);
750                 break;
751         case SIGCHLD:
752                 PRINT_DEC(MSG_ORIG(MSG_CNOTE_T_SI_PID), f_si_pid);
753                 PRINT_DEC(MSG_ORIG(MSG_CNOTE_T_SI_STATUS), f_si_status);
754                 break;
755         case SIGPOLL:
756                 PRINT_DEC(MSG_ORIG(MSG_CNOTE_T_SI_BAND), f_si_band);
757                 break;
758         }

760         indent_exit(state);
761 }


764 /*
765  * Output information from stack_t structure.
766  */
767 static void
768 dump_stack(note_state_t *state, const char *title)
769 {
770         const sl_stack_layout_t         *layout = state->ns_arch->stack;
771         Conv_cnote_ss_flags_buf_t       conv_buf;
772         Word            w;

774         indent_enter(state, title, &layout->ss_size);

776         print_num_2up(state, MSG_ORIG(MSG_CNOTE_T_SS_SP), &layout->ss_sp,
777             SL_FMT_NUM_ZHEX, MSG_ORIG(MSG_CNOTE_T_SS_SIZE), &layout->ss_size,
778             SL_FMT_NUM_HEX);

780         if (data_present(state, &layout->ss_flags)) {
781                 w = extract_as_word(state, &layout->ss_flags);
782                 print_str(state, MSG_ORIG(MSG_CNOTE_T_SS_FLAGS),
783                     conv_cnote_ss_flags(w, 0, &conv_buf));
784         }

786         indent_exit(state);
787 }


790 /*
791  * Output information from sysset_t structure.
792  */
793 static void
794 dump_sysset(note_state_t *state, const char *title)
795 {
796 #define NELTS 16

798         const sl_sysset_layout_t        *layout = state->ns_arch->sysset;
799         Conv_cnote_sysset_buf_t buf;
800         sl_field_t              fdesc;
801         uint32_t                mask[NELTS];
802         int                     i, nelts;

804         if (!data_present(state, &layout->sizeof_struct))
805                 return;

807         fdesc = layout->word;
808         nelts = fdesc.slf_nelts;
809         if (nelts > NELTS)      /* Type has grown? Show what we understand */
810                 nelts = NELTS;
811         for (i = 0; i < nelts; i++) {
```

```
812                 mask[i] = extract_as_word(state, &fdesc);
813                 fdesc.slf_offset += fdesc.slf_eltlen;
814         }

816         print_str(state, title, conv_cnote_sysset(mask, nelts, 0, &buf));

818 #undef NELTS
819 }


822 /*
823  * Output information from timestruc_t structure.
824  */
825 static void
826 dump_timestruc(note_state_t *state, const char *title)
827 {
828         const sl_timestruc_layout_t *layout = state->ns_arch->timestruc;

830         indent_enter(state, title, &layout->tv_sec);

832         PRINT_DEC_2UP(MSG_ORIG(MSG_CNOTE_T_TV_SEC), tv_sec,
833             MSG_ORIG(MSG_CNOTE_T_TV_NSEC), tv_nsec);

835         indent_exit(state);
836 }

838 /*
839  * Output information from psecflags_t structure.
840  */
841 static void
842 dump_secflags(note_state_t *state, const char *title)
843 {
844         const sl_psecflags_layout_t *layout = state->ns_arch->psecflags;
845         Conv_secflags_buf_t inv;
846         Word w;

848         indent_enter(state, title, &layout->psf_effective);

850         w = extract_as_word(state, &layout->psf_effective);
851         print_str(state, MSG_ORIG(MSG_CNOTE_T_PSF_EFFECTIVE),
852             conv_psecflags(w, 0, &inv));

854         w = extract_as_word(state, &layout->psf_inherit);
855         print_str(state, MSG_ORIG(MSG_CNOTE_T_PSF_INHERIT),
856             conv_psecflags(w, 0, &inv));
857 }
858 #endif /* ! codereview */

860 /*
861  * Output information from utsname structure.
862  */
863 static void
864 dump_utsname(note_state_t *state, const char *title)
865 {
866         const sl_utsname_layout_t       *layout = state->ns_arch->utsname;

868         indent_enter(state, title, &layout->sysname);

870         PRINT_STRBUF(MSG_ORIG(MSG_CNOTE_T_UTS_SYSNAME), sysname);
871         PRINT_STRBUF(MSG_ORIG(MSG_CNOTE_T_UTS_NODENAME), nodename);
872         PRINT_STRBUF(MSG_ORIG(MSG_CNOTE_T_UTS_RELEASE), release);
873         PRINT_STRBUF(MSG_ORIG(MSG_CNOTE_T_UTS_VERSION), version);
874         PRINT_STRBUF(MSG_ORIG(MSG_CNOTE_T_UTS_MACHINE), machine);

876         indent_exit(state);
877 }
```



```


880 /*
881  * Dump register contents
882  */
883 static void
884 dump_prgregset(note_state_t *state, const char *title)
885 {
886         sl_field_t      fdesc1, fdesc2;
887         sl_fmtbuf_t     buf1, buf2;
888         Conv_inv_buf_t  inv_buf1, inv_buf2;
889         Word            w;

891         fdesc1 = fdesc2 = state->ns_arch->prgregset->elt0;
892         indent_enter(state, title, &fdesc1);

894         for (w = 0; w < fdesc1.slf_nelts; ) {
895                 if (w == (fdesc1.slf_nelts - 1)) {
896                         /* One last register is left */
897                         if (!data_present(state, &fdesc1))
898                                 break;
899                         dbg_print(0, MSG_ORIG(MSG_CNOTE_FMT_LINE),
900                             INDENT, state->ns_vcol - state->ns_indent,
901                             conv_cnote_pr_regname(state->ns_mach, w,
902                             CONV_FMT_DECIMAL, &inv_buf1),
903                             fmt_num(state, &fdesc1, SL_FMT_NUM_ZHEX, buf1));
904                         fdesc1.slf_offset += fdesc1.slf_eltlen;
905                         w++;
906                         continue;
907                 }

909                 /* There are at least 2 more registers left. Show 2 up */
910                 fdesc2.slf_offset = fdesc1.slf_offset + fdesc1.slf_eltlen;
911                 if (!(data_present(state, &fdesc1) &&
912                     data_present(state, &fdesc2)))
913                         break;
914                 dbg_print(0, MSG_ORIG(MSG_CNOTE_FMT_LINE_2UP), INDENT,
915                     state->ns_vcol - state->ns_indent,
916                     conv_cnote_pr_regname(state->ns_mach, w,
917                     CONV_FMT_DECIMAL, &inv_buf1),
918                     state->ns_t2col - state->ns_vcol,
919                     fmt_num(state, &fdesc1, SL_FMT_NUM_ZHEX, buf1),
920                     state->ns_v2col - state->ns_t2col,
921                     conv_cnote_pr_regname(state->ns_mach, w + 1,
922                     CONV_FMT_DECIMAL, &inv_buf2),
923                     fmt_num(state, &fdesc2, SL_FMT_NUM_ZHEX, buf2));
924                 fdesc1.slf_offset += 2 * fdesc1.slf_eltlen;
925                 w += 2;
926         }

928         indent_exit(state);
929 }

931 /*
932  * Output information from lwpstatus_t structure.
933  */
934 static void
935 dump_lwpstatus(note_state_t *state, const char *title)
936 {
937         const sl_lwpstatus_layout_t     *layout = state->ns_arch->lwpstatus;
938         Word            w, w2;
939         int32_t         i;
940         union {
941                 Conv_inv_buf_t                  inv;
942                 Conv_cnote_pr_flags_buf_t       flags;
943         } conv_buf;
```

```
945         indent_enter(state, title, &layout->pr_flags);
947         if (data_present(state, &layout->pr_flags)) {
948                 w = extract_as_word(state, &layout->pr_flags);
949                 print_str(state, MSG_ORIG(MSG_CNOTE_T_PR_FLAGS),
950                     conv_cnote_pr_flags(w, 0, &conv_buf.flags));
951         }
953         PRINT_DEC(MSG_ORIG(MSG_CNOTE_T_PR_LWPID), pr_lwpid);
955         if (data_present(state, &layout->pr_why)) {
956                 w = extract_as_word(state, &layout->pr_why);
957                 print_str(state, MSG_ORIG(MSG_CNOTE_T_PR_WHY),
958                     conv_cnote_pr_why(w, 0, &conv_buf.inv));
960                 if (data_present(state, &layout->pr_what)) {
961                         w2 = extract_as_word(state, &layout->pr_what);
962                         print_str(state, MSG_ORIG(MSG_CNOTE_T_PR_WHAT),
963                             conv_cnote_pr_what(w, w2, 0, &conv_buf.inv));
964                 }
965         }
967         if (data_present(state, &layout->pr_cursig)) {
968                 w = extract_as_word(state, &layout->pr_cursig);
969                 print_str(state, MSG_ORIG(MSG_CNOTE_T_PR_CURSIG),
970                     conv_cnote_signal(w, CONV_FMT_DECIMAL, &conv_buf.inv));
971         }
973         PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_INFO), pr_info, dump_siginfo);
974         PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_LWPPEND), pr_lwppend,
975             dump_sigset);
976         PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_LWPHOLD), pr_lwphold,
977             dump_sigset);
978         PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_ACTION), pr_action,
979             dump_sigaction);
980         PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_ALTSTACK), pr_altstack,
981             dump_stack);
983         PRINT_ZHEX(MSG_ORIG(MSG_CNOTE_T_PR_OLDCONTEXT), pr_oldcontext);
985         if (data_present(state, &layout->pr_syscall)) {
986                 w = extract_as_word(state, &layout->pr_syscall);
987                 print_str(state, MSG_ORIG(MSG_CNOTE_T_PR_SYSCALL),
988                     conv_cnote_syscall(w, CONV_FMT_DECIMAL, &conv_buf.inv));
989         }
991         PRINT_DEC(MSG_ORIG(MSG_CNOTE_T_PR_NSYSARG), pr_nsysarg);
993         if (data_present(state, &layout->pr_errno)) {
994                 w = extract_as_word(state, &layout->pr_errno);
995                 print_str(state, MSG_ORIG(MSG_CNOTE_T_PR_ERRNO),
996                     conv_cnote_errno(w, CONV_FMT_DECIMAL, &conv_buf.inv));
997         }
999         if (data_present(state, &layout->pr_nsysarg)) {
1000                w2 = extract_as_word(state, &layout->pr_nsysarg);
1001                print_array(state, &layout->pr_sysarg, SL_FMT_NUM_ZHEX, w2, 1,
1002                    MSG_ORIG(MSG_CNOTE_T_PR_SYSARG));
1003        }
1005        PRINT_HEX_2UP(MSG_ORIG(MSG_CNOTE_T_PR_RVAL1), pr_rval1,
1006            MSG_ORIG(MSG_CNOTE_T_PR_RVAL2), pr_rval2);
1007        PRINT_STRBUF(MSG_ORIG(MSG_CNOTE_T_PR_CLNAME), pr_clname);
1008        PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_TSTAMP), pr_tstamp,
1009            dump_timestruc);
```



```
1010        PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_UTIME), pr_utime, dump_timestruc);
1011        PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_STIME), pr_stime, dump_timestruc);
1013        if (data_present(state, &layout->pr_errpriv)) {
1014                i = extract_as_sword(state, &layout->pr_errpriv);
1015                print_str(state, MSG_ORIG(MSG_CNOTE_T_PR_ERRPRIV),
1016                    conv_cnote_priv(i, CONV_FMT_DECIMAL, &conv_buf.inv));
1017        }
1019        PRINT_ZHEX_2UP(MSG_ORIG(MSG_CNOTE_T_PR_USTACK), pr_ustack,
1020            MSG_ORIG(MSG_CNOTE_T_PR_INSTR), pr_instr);
1022        /*
1023         * In order to line up all the values in a single column,
1024         * we would have to set vcol to a very high value, which results
1025         * in ugly looking output that runs off column 80. So, we use
1026         * two levels of vcol, one for the contents so far, and a
1027         * higher one for the pr_reg sub-struct.
1028         */
1029        state->ns_vcol += 3;
1030        state->ns_t2col += 3;
1031        state->ns_v2col += 2;
1032        PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_REG), pr_reg, dump_prgregset);
1033        state->ns_vcol -= 3;
1034        state->ns_t2col -= 3;
1035        state->ns_v2col -= 2;
1037        /*
1038         * The floating point register state is complex, and highly
1039         * platform dependent. For now, we simply display it as
1040         * a hex dump. This can be replaced if better information
1041         * is required.
1042         */
1043        if (data_present(state, &layout->pr_fpreg)) {
1044                indent_enter(state, MSG_ORIG(MSG_CNOTE_T_PR_FPREG),
1045                    &layout->pr_fpreg);
1046                dump_hex_bytes(layout->pr_fpreg.slf_offset + state->ns_data,
1047                    layout->pr_fpreg.slf_eltlen, state->ns_indent, 4, 3);
1048                indent_exit(state);
1049        }
1051        indent_exit(state);
1052 }


1055 /*
1056  * Output information from pstatus_t structure.
1057  */
1058 static void
1059 dump_pstatus(note_state_t *state, const char *title)
1060 {
1061         const sl_pstatus_layout_t       *layout = state->ns_arch->pstatus;
1062         Word                            w;
1063         union {
1064                 Conv_inv_buf_t                  inv;
1065                 Conv_cnote_pr_flags_buf_t       flags;
1066         } conv_buf;
1068         indent_enter(state, title, &layout->pr_flags);
1070         if (data_present(state, &layout->pr_flags)) {
1071                 w = extract_as_word(state, &layout->pr_flags);
1072                 print_str(state, MSG_ORIG(MSG_CNOTE_T_PR_FLAGS),
1073                     conv_cnote_pr_flags(w, 0, &conv_buf.flags));
1074         }
```

```
1076         PRINT_DEC(MSG_ORIG(MSG_CNOTE_T_PR_NLWP), pr_nlwp);
1077         PRINT_DEC_2UP(MSG_ORIG(MSG_CNOTE_T_PR_PID), pr_pid,
1078             MSG_ORIG(MSG_CNOTE_T_PR_PPID), pr_ppid);
1079         PRINT_DEC_2UP(MSG_ORIG(MSG_CNOTE_T_PR_PGID), pr_pgid,
1080             MSG_ORIG(MSG_CNOTE_T_PR_SID), pr_sid);
1081         PRINT_DEC_2UP(MSG_ORIG(MSG_CNOTE_T_PR_ASLWPID), pr_aslwpid,
1082             MSG_ORIG(MSG_CNOTE_T_PR_AGENTID), pr_agentid);
1083         PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_SIGPEND), pr_sigpend,
1084             dump_sigset);
1085         print_num_2up(state, MSG_ORIG(MSG_CNOTE_T_PR_BRKBASE),
1086             &layout->pr_brkbase, SL_FMT_NUM_ZHEX,
1087             MSG_ORIG(MSG_CNOTE_T_PR_BRKSIZE),
1088             &layout->pr_brksize, SL_FMT_NUM_HEX);
1089         print_num_2up(state, MSG_ORIG(MSG_CNOTE_T_PR_STKBASE),
1090             &layout->pr_stkbase, SL_FMT_NUM_ZHEX,
1091             MSG_ORIG(MSG_CNOTE_T_PR_STKSIZE),
1092             &layout->pr_stksize, SL_FMT_NUM_HEX);
1093         PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_UTIME), pr_utime, dump_timestruc);
1094         PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_STIME), pr_stime, dump_timestruc);
1095         PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_CUTIME), pr_cutime,
1096             dump_timestruc);
1097         PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_CSTIME), pr_cstime,
1098             dump_timestruc);
1099         PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_SIGTRACE), pr_sigtrace,
1100             dump_sigset);
1101         PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_FLTTRACE), pr_flttrace,
1102             dump_fltset);
1103         PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_SYSENTRY), pr_sysentry,
1104             dump_sysset);
1105         PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_SYSEXIT), pr_sysexit,
1106             dump_sysset);

1108         if (data_present(state, &layout->pr_dmodel)) {
1109                 w = extract_as_word(state, &layout->pr_dmodel);
1110                 print_str(state, MSG_ORIG(MSG_CNOTE_T_PR_DMODEL),
1111                     conv_cnote_pr_dmodel(w, 0, &conv_buf.inv));
1112         }

1114         PRINT_DEC_2UP(MSG_ORIG(MSG_CNOTE_T_PR_TASKID), pr_taskid,
1115             MSG_ORIG(MSG_CNOTE_T_PR_PROJID), pr_projid);
1116         PRINT_DEC_2UP(MSG_ORIG(MSG_CNOTE_T_PR_NZOMB), pr_nzomb,
1117             MSG_ORIG(MSG_CNOTE_T_PR_ZONEID), pr_zoneid);

1119         /*
1120          * In order to line up all the values in a single column,
1121          * we would have to set vcol to a very high value, which results
1122          * in ugly looking output that runs off column 80. So, we use
1123          * two levels of vcol, one for the contents so far, and a
1124          * higher one for the pr_lwp sub-struct.
1125          */
1126         state->ns_vcol += 5;
1127         state->ns_t2col += 5;
1128         state->ns_v2col += 5;

1130         PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_SECFLAGS), pr_secflags,
1131             dump_secflags);

1133 #endif /* ! codereview */
1134         PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_LWP), pr_lwp, dump_lwpstatus);
1135         state->ns_vcol -= 5;
1136         state->ns_t2col -= 5;
1137         state->ns_v2col -= 5;

1139         indent_exit(state);
1140 }
```



```
1143 /*
1144  * Output information from prstatus_t (<sys/old_procfs.h>) structure.
1145  */
1146 static void
1147 dump_prstatus(note_state_t *state, const char *title)
1148 {
1149         const sl_prstatus_layout_t      *layout = state->ns_arch->prstatus;
1150         Word                            w, w2;
1151         int                             i;
1152         union {
1153                 Conv_inv_buf_t                  inv;
1154                 Conv_cnote_old_pr_flags_buf_t   flags;
1155         } conv_buf;

1157         indent_enter(state, title, &layout->pr_flags);

1159         if (data_present(state, &layout->pr_flags)) {
1160                 w = extract_as_word(state, &layout->pr_flags);
1161                 print_str(state, MSG_ORIG(MSG_CNOTE_T_PR_FLAGS),
1162                     conv_cnote_old_pr_flags(w, 0, &conv_buf.flags));
1163         }

1165         if (data_present(state, &layout->pr_why)) {
1166                 w = extract_as_word(state, &layout->pr_why);
1167                 print_str(state, MSG_ORIG(MSG_CNOTE_T_PR_WHY),
1168                     conv_cnote_pr_why(w, 0, &conv_buf.inv));


1171                 if (data_present(state, &layout->pr_what)) {
1172                         w2 = extract_as_word(state, &layout->pr_what);
1173                         print_str(state, MSG_ORIG(MSG_CNOTE_T_PR_WHAT),
1174                             conv_cnote_pr_what(w, w2, 0, &conv_buf.inv));
1175                 }
1176         }

1178         PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_INFO), pr_info, dump_siginfo);

1180         if (data_present(state, &layout->pr_cursig)) {
1181                 w = extract_as_word(state, &layout->pr_cursig);
1182                 print_str(state, MSG_ORIG(MSG_CNOTE_T_PR_CURSIG),
1183                     conv_cnote_signal(w, CONV_FMT_DECIMAL, &conv_buf.inv));
1184         }

1186         PRINT_DEC(MSG_ORIG(MSG_CNOTE_T_PR_NLWP), pr_nlwp);
1187         PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_SIGPEND), pr_sigpend,
1188             dump_sigset);
1189         PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_SIGHOLD), pr_sighold,
1190             dump_sigset);
1191         PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_ALTSTACK), pr_altstack,
1192             dump_stack);
1193         PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_ACTION), pr_action,
1194             dump_sigaction);
1195         PRINT_DEC_2UP(MSG_ORIG(MSG_CNOTE_T_PR_PID), pr_pid,
1196             MSG_ORIG(MSG_CNOTE_T_PR_PPID), pr_ppid);
1197         PRINT_DEC_2UP(MSG_ORIG(MSG_CNOTE_T_PR_PGRP), pr_pgrp,
1198             MSG_ORIG(MSG_CNOTE_T_PR_SID), pr_sid);
1199         PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_UTIME), pr_utime, dump_timestruc);
1200         PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_STIME), pr_stime, dump_timestruc);
1201         PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_CUTIME), pr_cutime,
1202             dump_timestruc);
1203         PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_CSTIME), pr_cstime,
1204             dump_timestruc);
1205         PRINT_STRBUF(MSG_ORIG(MSG_CNOTE_T_PR_CLNAME), pr_clname);

1207         if (data_present(state, &layout->pr_syscall)) {
```

```
1208                 w = extract_as_word(state, &layout->pr_syscall);
1209                 print_str(state, MSG_ORIG(MSG_CNOTE_T_PR_SYSCALL),
1210                     conv_cnote_syscall(w, CONV_FMT_DECIMAL, &conv_buf.inv));
1211         }

1213         PRINT_DEC(MSG_ORIG(MSG_CNOTE_T_PR_NSYSARG), pr_nsysarg);

1215         if (data_present(state, &layout->pr_nsysarg)) {
1216                 w2 = extract_as_word(state, &layout->pr_nsysarg);
1217                 print_array(state, &layout->pr_sysarg, SL_FMT_NUM_ZHEX, w2, 1,
1218                     MSG_ORIG(MSG_CNOTE_T_PR_SYSARG));
1219         }

1221         PRINT_DEC(MSG_ORIG(MSG_CNOTE_T_PR_WHO), pr_who);
1222         PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_LWPPEND), pr_sigpend,
1223             dump_sigset);
1224         PRINT_ZHEX(MSG_ORIG(MSG_CNOTE_T_PR_OLDCONTEXT), pr_oldcontext);
1225         print_num_2up(state, MSG_ORIG(MSG_CNOTE_T_PR_BRKBASE),
1226             &layout->pr_brkbase, SL_FMT_NUM_ZHEX,
1227             MSG_ORIG(MSG_CNOTE_T_PR_BRKSIZE),
1228             &layout->pr_brksize, SL_FMT_NUM_HEX);
1229         print_num_2up(state, MSG_ORIG(MSG_CNOTE_T_PR_STKBASE),
1230             &layout->pr_stkbase, SL_FMT_NUM_ZHEX,
1231             MSG_ORIG(MSG_CNOTE_T_PR_STKSIZE),
1232             &layout->pr_stksize, SL_FMT_NUM_HEX);
1233         PRINT_DEC(MSG_ORIG(MSG_CNOTE_T_PR_PROCESSOR), pr_processor);

1235         if (data_present(state, &layout->pr_bind)) {
1236                 i = extract_as_sword(state, &layout->pr_bind);
1237                 print_str(state, MSG_ORIG(MSG_CNOTE_T_PR_BIND),
1238                     conv_cnote_psetid(i, CONV_FMT_DECIMAL, &conv_buf.inv));
1239         }

1241         PRINT_ZHEX(MSG_ORIG(MSG_CNOTE_T_PR_INSTR), pr_instr);
1242         PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_REG), pr_reg, dump_prgregset);

1244         indent_exit(state);
1245 }


1248 /*
1249  * Print percent from 16-bit binary fraction [0 .. 1]
1250  * Round up .01 to .1 to indicate some small percentage (the 0x7000 below).
1251  *
1252  * Note: This routine was copied from ps(1) and then modified.
1253  */
1254 static const char *
1255 prtpct_value(note_state_t *state, const sl_field_t *fdesc,
1256     sl_fmtbuf_t buf)
1257 {
1258         uint_t value;           /* need 32 bits to compute with */

1260         value = extract_as_word(state, fdesc);
1261         value = ((value * 1000) + 0x7000) >> 15;        /* [0 .. 1000] */
1262         if (value >= 1000)
1263                 value = 999;

1265         (void) snprintf(buf, sizeof (sl_fmtbuf_t),
1266             MSG_ORIG(MSG_CNOTE_FMT_PRTPCT), value / 10, value % 10);

1268         return (buf);
1269 }



1273 /*
```



```
1274  * Version of prtpct() used for a 2-up display of two adjacent percentages.
1275  */
1276 static void
1277 prtpct_2up(note_state_t *state, const sl_field_t *fdesc1,
1278     const char *title1, const sl_field_t *fdesc2, const char *title2)
1279 {
1280         sl_fmtbuf_t     buf1, buf2;

1282         if (!(data_present(state, fdesc1) &&
1283             data_present(state, fdesc2)))
1284                 return;

1286         dbg_print(0, MSG_ORIG(MSG_CNOTE_FMT_LINE_2UP), INDENT,
1287             state->ns_vcol - state->ns_indent, title1,
1288             state->ns_t2col - state->ns_vcol,
1289             prtpct_value(state, fdesc1, buf1),
1290             state->ns_v2col - state->ns_t2col, title2,
1291             prtpct_value(state, fdesc2, buf2));
1292 }


1295 /*
1296  * The psinfo_t and prpsinfo_t structs have pr_state and pr_sname
1297  * fields that we wish to print in a 2up format. The pr_state is
1298  * an integer, while pr_sname is a single character.
1299  */
1300 static void
1301 print_state_sname_2up(note_state_t *state,
1302     const sl_field_t *state_fdesc,
1303     const sl_field_t *sname_fdesc)
1304 {
1305         sl_fmtbuf_t     buf1, buf2;
1306         int             sname;

1308         /*
1309          * If the field slf_offset and extent fall past the end of the
1310          * available data, then return without doing anything. That note
1311          * is from an older core file that doesn't have all the fields
1312          * that we know about.
1313          */
1314         if (!(data_present(state, state_fdesc) &&
1315             data_present(state, sname_fdesc)))
1316                 return;

1318         sname = extract_as_sword(state, sname_fdesc);
1319         buf2[0] = sname;
1320         buf2[1] = '\0';

1322         dbg_print(0, MSG_ORIG(MSG_CNOTE_FMT_LINE_2UP), INDENT,
1323             state->ns_vcol - state->ns_indent, MSG_ORIG(MSG_CNOTE_T_PR_STATE),
1324             state->ns_t2col - state->ns_vcol,
1325             fmt_num(state, state_fdesc, SL_FMT_NUM_DEC, buf1),
1326             state->ns_v2col - state->ns_t2col, MSG_ORIG(MSG_CNOTE_T_PR_SNAME),
1327             buf2);
1328 }

1330 /*
1331  * Output information from lwpsinfo_t structure.
1332  */
1333 static void
1334 dump_lwpsinfo(note_state_t *state, const char *title)
1335 {
1336         const sl_lwpsinfo_layout_t      *layout = state->ns_arch->lwpsinfo;
1337         Word                    w;
1338         int32_t                 i;
1339         union {
```

```
1340                Conv_cnote_proc_flag_buf_t      proc_flag;
1341                Conv_inv_buf_t                  inv;
1342        } conv_buf;

1344        indent_enter(state, title, &layout->pr_flag);

1346        if (data_present(state, &layout->pr_flag)) {
1347                w = extract_as_word(state, &layout->pr_flag);
1348                print_str(state, MSG_ORIG(MSG_CNOTE_T_PR_FLAG),
1349                    conv_cnote_proc_flag(w, 0, &conv_buf.proc_flag));
1350        }

1352        print_num_2up(state, MSG_ORIG(MSG_CNOTE_T_PR_LWPID), &layout->pr_lwpid,
1353            SL_FMT_NUM_DEC, MSG_ORIG(MSG_CNOTE_T_PR_ADDR), &layout->pr_addr,
1354            SL_FMT_NUM_ZHEX);
1355        PRINT_HEX(MSG_ORIG(MSG_CNOTE_T_PR_WCHAN), pr_wchan);

1357        if (data_present(state, &layout->pr_stype)) {
1358                w = extract_as_word(state, &layout->pr_stype);
1359                print_str(state, MSG_ORIG(MSG_CNOTE_T_PR_STYPE),
1360                    conv_cnote_pr_stype(w, CONV_FMT_DECIMAL, &conv_buf.inv));
1361        }

1363        print_state_sname_2up(state, &layout->pr_state, &layout->pr_sname);

1365        PRINT_DEC(MSG_ORIG(MSG_CNOTE_T_PR_NICE), pr_nice);

1367        if (data_present(state, &layout->pr_syscall)) {
1368                w = extract_as_word(state, &layout->pr_syscall);
1369                print_str(state, MSG_ORIG(MSG_CNOTE_T_PR_SYSCALL),
1370                    conv_cnote_syscall(w, CONV_FMT_DECIMAL, &conv_buf.inv));
1371        }

1373        PRINT_DEC_2UP(MSG_ORIG(MSG_CNOTE_T_PR_OLDPRI), pr_oldpri,
1374            MSG_ORIG(MSG_CNOTE_T_PR_CPU), pr_cpu);

1376        if (data_present(state, &layout->pr_pri) &&
1377            data_present(state, &layout->pr_pctcpu)) {
1378                sl_fmtbuf_t     buf1, buf2;

1380                dbg_print(0, MSG_ORIG(MSG_CNOTE_FMT_LINE_2UP), INDENT,
1381                    state->ns_vcol - state->ns_indent,
1382                    MSG_ORIG(MSG_CNOTE_T_PR_PRI),
1383                    state->ns_t2col - state->ns_vcol,
1384                    fmt_num(state, &layout->pr_pri, SL_FMT_NUM_DEC, buf1),
1385                    state->ns_v2col - state->ns_t2col,
1386                    MSG_ORIG(MSG_CNOTE_T_PR_PCTCPU),
1387                    prtpct_value(state, &layout->pr_pctcpu, buf2));
1388        }

1390        PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_START), pr_start, dump_timestruc);
1391        PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_TIME), pr_time, dump_timestruc);
1392        PRINT_STRBUF(MSG_ORIG(MSG_CNOTE_T_PR_CLNAME), pr_clname);
1393        PRINT_STRBUF(MSG_ORIG(MSG_CNOTE_T_PR_NAME), pr_name);
1394        PRINT_DEC_2UP(MSG_ORIG(MSG_CNOTE_T_PR_ONPRO), pr_onpro,
1395            MSG_ORIG(MSG_CNOTE_T_PR_BINDPRO), pr_bindpro);

1397        if (data_present(state, &layout->pr_bindpset)) {
1398                i = extract_as_sword(state, &layout->pr_bindpset);
1399                print_str(state, MSG_ORIG(MSG_CNOTE_T_PR_BINDPSET),
1400                    conv_cnote_psetid(i, CONV_FMT_DECIMAL, &conv_buf.inv));
1401        }

1403        PRINT_DEC(MSG_ORIG(MSG_CNOTE_T_PR_LGRP), pr_lgrp);

1405        indent_exit(state);
```



```
1406 }


1409 /*
1410  * Output information from psinfo_t structure.
1411  */
1412 static void
1413 dump_psinfo(note_state_t *state, const char *title)
1414 {
1415         const sl_psinfo_layout_t        *layout = state->ns_arch->psinfo;
1416         Word                            w;
1417         union {
1418                 Conv_cnote_proc_flag_buf_t      proc_flag;
1419                 Conv_inv_buf_t                  inv;
1420         } conv_buf;

1422         indent_enter(state, title, &layout->pr_flag);

1424         if (data_present(state, &layout->pr_flag)) {
1425                 w = extract_as_word(state, &layout->pr_flag);
1426                 print_str(state, MSG_ORIG(MSG_CNOTE_T_PR_FLAG),
1427                     conv_cnote_proc_flag(w, 0, &conv_buf.proc_flag));
1428         }

1430         PRINT_DEC(MSG_ORIG(MSG_CNOTE_T_PR_NLWP), pr_nlwp);
1431         PRINT_DEC_2UP(MSG_ORIG(MSG_CNOTE_T_PR_PID), pr_pid,
1432             MSG_ORIG(MSG_CNOTE_T_PR_PPID), pr_ppid);
1433         PRINT_DEC_2UP(MSG_ORIG(MSG_CNOTE_T_PR_PGID), pr_pgid,
1434             MSG_ORIG(MSG_CNOTE_T_PR_SID), pr_sid);
1435         PRINT_DEC_2UP(MSG_ORIG(MSG_CNOTE_T_PR_UID), pr_uid,
1436             MSG_ORIG(MSG_CNOTE_T_PR_EUID), pr_euid);
1437         PRINT_DEC_2UP(MSG_ORIG(MSG_CNOTE_T_PR_GID), pr_gid,
1438             MSG_ORIG(MSG_CNOTE_T_PR_EGID), pr_egid);
1439         print_num_2up(state, MSG_ORIG(MSG_CNOTE_T_PR_ADDR), &layout->pr_addr,
1440             SL_FMT_NUM_ZHEX, MSG_ORIG(MSG_CNOTE_T_PR_SIZE), &layout->pr_size,
1441             SL_FMT_NUM_HEX);
1442         print_num_2up(state, MSG_ORIG(MSG_CNOTE_T_PR_RSSIZE),
1443             &layout->pr_rssize, SL_FMT_NUM_HEX, MSG_ORIG(MSG_CNOTE_T_PR_TTYDEV),
1444             &layout->pr_ttydev, SL_FMT_NUM_DEC);
1445         prtpct_2up(state, &layout->pr_pctcpu, MSG_ORIG(MSG_CNOTE_T_PR_PCTCPU),
1446             &layout->pr_pctmem, MSG_ORIG(MSG_CNOTE_T_PR_PCTMEM));
1447         PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_START), pr_start, dump_timestruc);
1448         PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_TIME), pr_time, dump_timestruc);
1449         PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_CTIME), pr_ctime, dump_timestruc);
1450         PRINT_STRBUF(MSG_ORIG(MSG_CNOTE_T_PR_FNAME), pr_fname);
1451         PRINT_STRBUF(MSG_ORIG(MSG_CNOTE_T_PR_PSARGS), pr_psargs);
1452         print_num_2up(state, MSG_ORIG(MSG_CNOTE_T_PR_WSTAT), &layout->pr_wstat,
1453             SL_FMT_NUM_HEX, MSG_ORIG(MSG_CNOTE_T_PR_ARGC), &layout->pr_argc,
1454             SL_FMT_NUM_DEC);
1455         PRINT_ZHEX_2UP(MSG_ORIG(MSG_CNOTE_T_PR_ARGV), pr_argv,
1456             MSG_ORIG(MSG_CNOTE_T_PR_ENVP), pr_envp);

1458         if (data_present(state, &layout->pr_dmodel)) {
1459                 w = extract_as_word(state, &layout->pr_dmodel);
1460                 print_str(state, MSG_ORIG(MSG_CNOTE_T_PR_DMODEL),
1461                     conv_cnote_pr_dmodel(w, 0, &conv_buf.inv));
1462         }

1464         PRINT_DEC_2UP(MSG_ORIG(MSG_CNOTE_T_PR_TASKID), pr_taskid,
1465             MSG_ORIG(MSG_CNOTE_T_PR_PROJID), pr_projid);
1466         PRINT_DEC_2UP(MSG_ORIG(MSG_CNOTE_T_PR_NZOMB), pr_nzomb,
1467             MSG_ORIG(MSG_CNOTE_T_PR_POOLID), pr_poolid);
1468         PRINT_DEC_2UP(MSG_ORIG(MSG_CNOTE_T_PR_ZONEID), pr_zoneid,
1469             MSG_ORIG(MSG_CNOTE_T_PR_CONTRACT), pr_contract);

1471         PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_LWP), pr_lwp, dump_lwpsinfo);
```

```
1473         indent_exit(state);
1474 }

1476 /*
1477  * Output information from prpsinfo_t structure.
1478  */
1479 static void
1480 dump_prpsinfo(note_state_t *state, const char *title)
1481 {
1482         const sl_prpsinfo_layout_t      *layout = state->ns_arch->prpsinfo;
1483         Word                            w;
1484         union {
1485                 Conv_cnote_proc_flag_buf_t      proc_flag;
1486                 Conv_inv_buf_t                  inv;
1487         } conv_buf;

1489         indent_enter(state, title, &layout->pr_state);

1491         print_state_sname_2up(state, &layout->pr_state, &layout->pr_sname);
1492         PRINT_DEC_2UP(MSG_ORIG(MSG_CNOTE_T_PR_ZOMB), pr_zomb,
1493             MSG_ORIG(MSG_CNOTE_T_PR_NICE), pr_nice);

1495         if (data_present(state, &layout->pr_flag)) {
1496                 w = extract_as_word(state, &layout->pr_flag);
1497                 print_str(state, MSG_ORIG(MSG_CNOTE_T_PR_FLAG),
1498                     conv_cnote_proc_flag(w, 0, &conv_buf.proc_flag));
1499         }


1502         PRINT_DEC_2UP(MSG_ORIG(MSG_CNOTE_T_PR_UID), pr_uid,
1503             MSG_ORIG(MSG_CNOTE_T_PR_GID), pr_gid);
1504         PRINT_DEC_2UP(MSG_ORIG(MSG_CNOTE_T_PR_PID), pr_pid,
1505             MSG_ORIG(MSG_CNOTE_T_PR_PPID), pr_ppid);
1506         PRINT_DEC_2UP(MSG_ORIG(MSG_CNOTE_T_PR_PGRP), pr_pgrp,
1507             MSG_ORIG(MSG_CNOTE_T_PR_SID), pr_sid);
1508         print_num_2up(state, MSG_ORIG(MSG_CNOTE_T_PR_ADDR), &layout->pr_addr,
1509             SL_FMT_NUM_ZHEX, MSG_ORIG(MSG_CNOTE_T_PR_SIZE), &layout->pr_size,
1510             SL_FMT_NUM_HEX);
1511         PRINT_HEX_2UP(MSG_ORIG(MSG_CNOTE_T_PR_RSSIZE), pr_rssize,
1512             MSG_ORIG(MSG_CNOTE_T_PR_WCHAN), pr_wchan);
1513         PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_START), pr_start, dump_timestruc);
1514         PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_TIME), pr_time, dump_timestruc);
1515         PRINT_DEC_2UP(MSG_ORIG(MSG_CNOTE_T_PR_PRI), pr_pri,
1516             MSG_ORIG(MSG_CNOTE_T_PR_OLDPRI), pr_oldpri);
1517         PRINT_DEC(MSG_ORIG(MSG_CNOTE_T_PR_CPU), pr_cpu);
1518         PRINT_DEC_2UP(MSG_ORIG(MSG_CNOTE_T_PR_OTTYDEV), pr_ottydev,
1519             MSG_ORIG(MSG_CNOTE_T_PR_LTTYDEV), pr_lttydev);
1520         PRINT_STRBUF(MSG_ORIG(MSG_CNOTE_T_PR_CLNAME), pr_clname);
1521         PRINT_STRBUF(MSG_ORIG(MSG_CNOTE_T_PR_FNAME), pr_fname);
1522         PRINT_STRBUF(MSG_ORIG(MSG_CNOTE_T_PR_PSARGS), pr_psargs);

1524         if (data_present(state, &layout->pr_syscall)) {
1525                 w = extract_as_word(state, &layout->pr_syscall);
1526                 print_str(state, MSG_ORIG(MSG_CNOTE_T_PR_SYSCALL),
1527                     conv_cnote_syscall(w, CONV_FMT_DECIMAL, &conv_buf.inv));
1528         }

1530         PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_CTIME), pr_ctime, dump_timestruc);
1531         PRINT_HEX_2UP(MSG_ORIG(MSG_CNOTE_T_PR_BYSIZE), pr_bysize,
1532             MSG_ORIG(MSG_CNOTE_T_PR_BYRSSIZE), pr_byrssize);
1533         print_num_2up(state, MSG_ORIG(MSG_CNOTE_T_PR_ARGC), &layout->pr_argc,
1534             SL_FMT_NUM_DEC, MSG_ORIG(MSG_CNOTE_T_PR_ARGV), &layout->pr_argv,
1535             SL_FMT_NUM_ZHEX);
1536         print_num_2up(state, MSG_ORIG(MSG_CNOTE_T_PR_ENVP), &layout->pr_envp,
1537             SL_FMT_NUM_ZHEX, MSG_ORIG(MSG_CNOTE_T_PR_WSTAT), &layout->pr_wstat,
```



```
1538             SL_FMT_NUM_HEX);
1539         prtpct_2up(state, &layout->pr_pctcpu, MSG_ORIG(MSG_CNOTE_T_PR_PCTCPU),
1540             &layout->pr_pctmem, MSG_ORIG(MSG_CNOTE_T_PR_PCTMEM));
1541         PRINT_DEC_2UP(MSG_ORIG(MSG_CNOTE_T_PR_EUID), pr_euid,
1542             MSG_ORIG(MSG_CNOTE_T_PR_EGID), pr_egid);
1543         PRINT_DEC(MSG_ORIG(MSG_CNOTE_T_PR_ASLWPID), pr_aslwpid);

1545         if (data_present(state, &layout->pr_dmodel)) {
1546                 w = extract_as_word(state, &layout->pr_dmodel);
1547                 print_str(state, MSG_ORIG(MSG_CNOTE_T_PR_DMODEL),
1548                     conv_cnote_pr_dmodel(w, 0, &conv_buf.inv));
1549         }

1551         indent_exit(state);
1552 }


1555 /*
1556  * Output information from prcred_t structure.
1557  */
1558 static void
1559 dump_prcred(note_state_t *state, const char *title)
1560 {
1561         const sl_prcred_layout_t *layout = state->ns_arch->prcred;
1562         Word            ngroups;

1564         indent_enter(state, title, &layout->pr_euid);

1566         PRINT_DEC_2UP(MSG_ORIG(MSG_CNOTE_T_PR_EUID), pr_euid,
1567             MSG_ORIG(MSG_CNOTE_T_PR_RUID), pr_ruid);
1568         PRINT_DEC_2UP(MSG_ORIG(MSG_CNOTE_T_PR_SUID), pr_suid,
1569             MSG_ORIG(MSG_CNOTE_T_PR_EGID), pr_egid);
1570         PRINT_DEC_2UP(MSG_ORIG(MSG_CNOTE_T_PR_RGID), pr_rgid,
1571             MSG_ORIG(MSG_CNOTE_T_PR_SGID), pr_sgid);
1572         PRINT_DEC(MSG_ORIG(MSG_CNOTE_T_PR_NGROUPS), pr_ngroups);

1574         if (data_present(state, &layout->pr_ngroups)) {
1575                 ngroups = extract_as_word(state, &layout->pr_ngroups);
1576                 print_array(state, &layout->pr_groups, SL_FMT_NUM_DEC, ngroups,
1577                     0, MSG_ORIG(MSG_CNOTE_T_PR_GROUPS));
1578         }

1580         indent_exit(state);
1581 }


1584 /*
1585  * Output information from prpriv_t structure.
1586  */
1587 static void
1588 dump_prpriv(note_state_t *state, const char *title)
1589 {
1590         const sl_prpriv_layout_t *layout = state->ns_arch->prpriv;
1591         Word            nsets;

1593         indent_enter(state, title, &layout->pr_nsets);

1595         PRINT_DEC(MSG_ORIG(MSG_CNOTE_T_PR_NSETS), pr_nsets);
1596         PRINT_HEX(MSG_ORIG(MSG_CNOTE_T_PR_SETSIZE), pr_setsize);
1597         PRINT_HEX(MSG_ORIG(MSG_CNOTE_T_PR_INFOSIZE), pr_infosize);

1599         if (data_present(state, &layout->pr_nsets)) {
1600                 nsets = extract_as_word(state, &layout->pr_nsets);
1601                 print_array(state, &layout->pr_sets, SL_FMT_NUM_ZHEX, nsets,
1602                     0, MSG_ORIG(MSG_CNOTE_T_PR_SETS));
1603         }
```

```
1605         indent_exit(state);
1606 }

1608 static void
1609 dump_prfdinfo(note_state_t *state, const char *title)
1610 {
1611         const sl_prfdinfo_layout_t *layout = state->ns_arch->prfdinfo;
1612         char buf[1024];
1613         uint32_t fileflags, mode;

1615         indent_enter(state, title, &layout->pr_fd);

1617         PRINT_DEC(MSG_ORIG(MSG_CNOTE_T_PR_FD), pr_fd);
1618         mode = extract_as_word(state, &layout->pr_mode);

1620         print_str(state, MSG_ORIG(MSG_CNOTE_T_PR_MODE),
1621             conv_cnote_filemode(mode, 0, buf, sizeof (buf)));

1623         PRINT_DEC_2UP(MSG_ORIG(MSG_CNOTE_T_PR_UID), pr_uid,
1624             MSG_ORIG(MSG_CNOTE_T_PR_GID), pr_gid);

1626         PRINT_DEC_2UP(MSG_ORIG(MSG_CNOTE_T_PR_MAJOR), pr_major,
1627             MSG_ORIG(MSG_CNOTE_T_PR_MINOR), pr_minor);
1628         PRINT_DEC_2UP(MSG_ORIG(MSG_CNOTE_T_PR_RMAJOR), pr_rmajor,
1629             MSG_ORIG(MSG_CNOTE_T_PR_RMINOR), pr_rminor);

1631         PRINT_DEC(MSG_ORIG(MSG_CNOTE_T_PR_INO), pr_ino);

1633         PRINT_DEC_2UP(MSG_ORIG(MSG_CNOTE_T_PR_SIZE), pr_size,
1634             MSG_ORIG(MSG_CNOTE_T_PR_OFFSET), pr_offset);

1636         fileflags = extract_as_word(state, &layout->pr_fileflags);

1638         print_str(state, MSG_ORIG(MSG_CNOTE_T_PR_FILEFLAGS),
1639             conv_cnote_fileflags(fileflags, 0, buf, sizeof (buf)));

1641         PRINT_DEC(MSG_ORIG(MSG_CNOTE_T_PR_FDFLAGS), pr_fdflags);

1643         PRINT_STRBUF(MSG_ORIG(MSG_CNOTE_T_PR_PATH), pr_path);

1645         indent_exit(state);
1646 }

1648 /*
1649  * Output information from priv_impl_info_t structure.
1650  */
1651 static void
1652 dump_priv_impl_info(note_state_t *state, const char *title)
1653 {
1654         const sl_priv_impl_info_layout_t *layout;

1656         layout = state->ns_arch->priv_impl_info;
1657         indent_enter(state, title, &layout->priv_headersize);

1659         PRINT_HEX_2UP(MSG_ORIG(MSG_CNOTE_T_PRIV_HEADERSIZE), priv_headersize,
1660             MSG_ORIG(MSG_CNOTE_T_PRIV_FLAGS), priv_flags);

1662         print_num_2up(state, MSG_ORIG(MSG_CNOTE_T_PRIV_NSETS),
1663             &layout->priv_nsets, SL_FMT_NUM_DEC,
1664             MSG_ORIG(MSG_CNOTE_T_PRIV_SETSIZE), &layout->priv_setsize,
1665             SL_FMT_NUM_HEX);
1666         print_num_2up(state, MSG_ORIG(MSG_CNOTE_T_PRIV_MAX), &layout->priv_max,
1667             SL_FMT_NUM_DEC, MSG_ORIG(MSG_CNOTE_T_PRIV_INFOSIZE),
1668             &layout->priv_infosize, SL_FMT_NUM_HEX);
1669         PRINT_HEX(MSG_ORIG(MSG_CNOTE_T_PRIV_GLOBALINFOSIZE),
```



```
1670             priv_globalinfosize);

1672         indent_exit(state);
1673 }


1676 /*
1677  * Dump information from an asrset_t array. This data
1678  * structure is specific to sparcv9, and does not appear
1679  * on any other platform.
1680  *
1681  * asrset_t is a simple array, defined in <sys/regset.h> as
1682  *      typedef int64_t asrset_t[16];    %asr16 - > %asr31
1683  *
1684  * As such, we do not make use of the struct_layout facilities
1685  * for this routine.
1686  */
1687 static void
1688 dump_asrset(note_state_t *state, const char *title)
1689 {
1690         static const sl_field_t ftemplate = { 0, sizeof (int64_t), 16, 0 };
1691         sl_field_t      fdesc1, fdesc2;
1692         sl_fmtbuf_t     buf1, buf2;
1693         char            index1[MAXNDXSIZE * 2], index2[MAXNDXSIZE * 2];
1694         Word            w, nelts;

1696         fdesc1 = fdesc2 =  ftemplate;

1698         /* We expect 16 values, but will print whatever is actually there */
1699         nelts = state->ns_len / ftemplate.slf_eltlen;
1700         if (nelts == 0)
1701                 return;

1703         indent_enter(state, title, &fdesc1);

1705         for (w = 0; w < nelts; ) {
1706                 (void) snprintf(index1, sizeof (index1),
1707                     MSG_ORIG(MSG_FMT_ASRINDEX), w + 16);

1709                 if (w == (nelts - 1)) {
1710                         /* One last register is left */
1711                         dbg_print(0, MSG_ORIG(MSG_CNOTE_FMT_LINE),
1712                             INDENT, state->ns_vcol - state->ns_indent, index1,
1713                             fmt_num(state, &fdesc1, SL_FMT_NUM_ZHEX, buf1));
1714                         fdesc1.slf_offset += fdesc1.slf_eltlen;
1715                         w++;
1716                         continue;
1717                 }

1719                 /* There are at least 2 more registers left. Show 2 up */
1720                 (void) snprintf(index2, sizeof (index2),
1721                     MSG_ORIG(MSG_FMT_ASRINDEX), w + 17);

1723                 fdesc2.slf_offset = fdesc1.slf_offset + fdesc1.slf_eltlen;
1724                 dbg_print(0, MSG_ORIG(MSG_CNOTE_FMT_LINE_2UP), INDENT,
1725                     state->ns_vcol - state->ns_indent, index1,
1726                     state->ns_t2col - state->ns_vcol,
1727                     fmt_num(state, &fdesc1, SL_FMT_NUM_ZHEX, buf1),
1728                     state->ns_v2col - state->ns_t2col, index2,
1729                     fmt_num(state, &fdesc2, SL_FMT_NUM_ZHEX, buf2));
1730                 fdesc1.slf_offset += 2 * fdesc1.slf_eltlen;
1731                 w += 2;
1732         }

1734         indent_exit(state);
1735 }
```

```
1737 corenote_ret_t
1738 corenote(Half mach, int do_swap, Word type,
1739     const char *desc, Word descsz)
1740 {
1741         note_state_t            state;

1743         /*
1744          * Get the per-architecture layout definition
1745          */
1746         state.ns_mach = mach;
1747         state.ns_arch = sl_mach(state.ns_mach);
1748         if (sl_mach(state.ns_mach) == NULL)
1749                 return (CORENOTE_R_BADARCH);

1751         state.ns_swap = do_swap;
1752         state.ns_indent = 4;
1753         state.ns_t2col = state.ns_v2col = 0;
1754         state.ns_data = desc;
1755         state.ns_len = descsz;

1757         switch (type) {
1758         case NT_PRSTATUS:               /* prstatus_t <sys/old_procfs.h> */
1759                 state.ns_vcol = 26;
1760                 state.ns_t2col = 46;
1761                 state.ns_v2col = 60;
1762                 dump_prstatus(&state, MSG_ORIG(MSG_CNOTE_DESC_PRSTATUS_T));
1763                 return (CORENOTE_R_OK);

1765         case NT_PRFPREG:                /* prfpregset_t <sys/procfs_isa.h> */
1766                 return (CORENOTE_R_OK_DUMP);

1768         case NT_PRPSINFO:               /* prpsinfo_t   <sys/old_procfs.h> */
1769                 state.ns_vcol = 20;
1770                 state.ns_t2col = 41;
1771                 state.ns_v2col = 54;
1772                 dump_prpsinfo(&state, MSG_ORIG(MSG_CNOTE_DESC_PRPSINFO_T));
1773                 return (CORENOTE_R_OK);

1775         case NT_PRXREG:                 /* prxregset_t <sys/procfs_isa.h> */
1776                 return (CORENOTE_R_OK_DUMP);

1778         case NT_PLATFORM:               /* string from sysinfo(SI_PLATFORM) */
1779                 dbg_print(0, MSG_ORIG(MSG_NOTE_DESC));
1780                 dbg_print(0, MSG_ORIG(MSG_FMT_INDENT), safe_str(desc, descsz));
1781                 return (CORENOTE_R_OK);

1783         case NT_AUXV:                   /* auxv_t array <sys/auxv.h> */
1784                 state.ns_vcol = 18;
1785                 dump_auxv(&state, MSG_ORIG(MSG_CNOTE_DESC_AUXV_T));
1786                 return (CORENOTE_R_OK);

1788         case NT_GWINDOWS:               /* gwindows_t SPARC only */
1789                 return (CORENOTE_R_OK_DUMP);

1791         case NT_ASRS:                   /* asrset_t <sys/regset> sparcv9 only */
1792                 state.ns_vcol = 18;
1793                 state.ns_t2col = 38;
1794                 state.ns_v2col = 46;
1795                 dump_asrset(&state, MSG_ORIG(MSG_CNOTE_DESC_ASRSET_T));
1796                 return (CORENOTE_R_OK);

1798         case NT_LDT:                    /* ssd array <sys/sysi86.h> IA32 only */
1799                 return (CORENOTE_R_OK_DUMP);

1801         case NT_PSTATUS:                /* pstatus_t <sys/procfs.h> */
```



```
1802                 state.ns_vcol = 22;
1803                 state.ns_t2col = 42;
1804                 state.ns_v2col = 54;
1805                 dump_pstatus(&state, MSG_ORIG(MSG_CNOTE_DESC_PSTATUS_T));
1806                 return (CORENOTE_R_OK);

1808         case NT_PSINFO:                 /* psinfo_t <sys/procfs.h> */
1809                 state.ns_vcol = 25;
1810                 state.ns_t2col = 45;
1811                 state.ns_v2col = 58;
1812                 dump_psinfo(&state, MSG_ORIG(MSG_CNOTE_DESC_PSINFO_T));
1813                 return (CORENOTE_R_OK);

1815         case NT_PRCRED:                 /* prcred_t <sys/procfs.h> */
1816                 state.ns_vcol = 20;
1817                 state.ns_t2col = 34;
1818                 state.ns_v2col = 44;
1819                 dump_prcred(&state, MSG_ORIG(MSG_CNOTE_DESC_PRCRED_T));
1820                 return (CORENOTE_R_OK);

1822         case NT_UTSNAME:                /* struct utsname <sys/utsname.h> */
1823                 state.ns_vcol = 18;
1824                 dump_utsname(&state, MSG_ORIG(MSG_CNOTE_DESC_STRUCT_UTSNAME));
1825                 return (CORENOTE_R_OK);

1827         case NT_LWPSTATUS:              /* lwpstatus_t <sys/procfs.h> */
1828                 state.ns_vcol = 24;
1829                 state.ns_t2col = 44;
1830                 state.ns_v2col = 54;
1831                 dump_lwpstatus(&state, MSG_ORIG(MSG_CNOTE_DESC_LWPSTATUS_T));
1832                 return (CORENOTE_R_OK);

1834         case NT_LWPSINFO:               /* lwpsinfo_t <sys/procfs.h> */
1835                 state.ns_vcol = 22;
1836                 state.ns_t2col = 42;
1837                 state.ns_v2col = 54;
1838                 dump_lwpsinfo(&state, MSG_ORIG(MSG_CNOTE_DESC_LWPSINFO_T));
1839                 return (CORENOTE_R_OK);

1841         case NT_PRPRIV:                 /* prpriv_t <sys/procfs.h> */
1842                 state.ns_vcol = 21;
1843                 state.ns_t2col = 34;
1844                 state.ns_v2col = 38;
1845                 dump_prpriv(&state, MSG_ORIG(MSG_CNOTE_DESC_PRPRIV_T));
1846                 return (CORENOTE_R_OK);

1848         case NT_PRPRIVINFO:             /* priv_impl_info_t <sys/priv.h> */
1849                 state.ns_vcol = 29;
1850                 state.ns_t2col = 41;
1851                 state.ns_v2col = 56;
1852                 dump_priv_impl_info(&state,
1853                     MSG_ORIG(MSG_CNOTE_DESC_PRIV_IMPL_INFO_T));
1854                 return (CORENOTE_R_OK);

1856         case NT_CONTENT:                /* core_content_t <sys/corectl.h> */
1857                 if (sizeof (core_content_t) > descsz)
1858                         return (CORENOTE_R_BADDATA);
1859                 {
1860                         static sl_field_t fdesc = { 0, 8, 0, 0 };
1861                         Conv_cnote_cc_content_buf_t conv_buf;
1862                         core_content_t content;

1864                         state.ns_vcol = 8;
1865                         indent_enter(&state,
1866                             MSG_ORIG(MSG_CNOTE_DESC_CORE_CONTENT_T),
1867                             &fdesc);
```

```
1868                        content = extract_as_lword(&state, &fdesc);
1869                        print_str(&state, MSG_ORIG(MSG_STR_EMPTY),
1870                            conv_cnote_cc_content(content, 0, &conv_buf));
1871                        indent_exit(&state);
1872                }
1873                return (CORENOTE_R_OK);

1875        case NT_ZONENAME:               /* string from getzonenamebyid(3C) */
1876                dbg_print(0, MSG_ORIG(MSG_NOTE_DESC));
1877                dbg_print(0, MSG_ORIG(MSG_FMT_INDENT), safe_str(desc, descsz));
1878                return (CORENOTE_R_OK);


1881        case NT_FDINFO:
1882                state.ns_vcol = 22;
1883                state.ns_t2col = 41;
1884                state.ns_v2col = 54;
1885                dump_prfdinfo(&state, MSG_ORIG(MSG_CNOTE_DESC_PRFDINFO_T));
1886                return (CORENOTE_R_OK);

1888        case NT_SPYMASTER:
1889                state.ns_vcol = 25;
1890                state.ns_t2col = 45;
1891                state.ns_v2col = 58;
1892                dump_psinfo(&state, MSG_ORIG(MSG_CNOTE_DESC_PSINFO_T));
1893                return (CORENOTE_R_OK);
1894        }

1896        return (CORENOTE_R_BADTYPE);
1897 }
```

```
**********************************************************
   26338 Wed May 27 19:48:58 2015
new/usr/src/cmd/sgs/elfdump/common/elfdump.msg
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#

#
# Copyright (c) 1997, 2010, Oracle and/or its affiliates. All rights reserved.
# Copyright 2012 DEY Storage Systems, Inc.  All rights reserved.
#

@ _START_

# Message file for cmd/sgs/elfdump.

@ MSG_ID_ELFDUMP

# Usage Messages

@ MSG_USAGE_BRIEF       "usage: %s [-cCdegGhHiklmnrSsuvy] [-I index] \
                         [-N name] [-O osabi] [-T type] [-p | -w outfile] \
                         file...\n"
@ MSG_USAGE_DETAIL1     "\t[-c]\t\tdump section header information\n"
@ MSG_USAGE_DETAIL2     "\t[-C]\t\tdemangle C++ symbol names\n"
@ MSG_USAGE_DETAIL3     "\t[-d]\t\tdump the contents of the .dynamic section\n"
@ MSG_USAGE_DETAIL4     "\t[-e]\t\tdump the elf header\n"
@ MSG_USAGE_DETAIL5     "\t[-g]\t\tdump the contents of the .group sections\n"
@ MSG_USAGE_DETAIL6     "\t[-G]\t\tdump the contents of the .got section\n"
@ MSG_USAGE_DETAIL7     "\t[-h]\t\tdump the contents of the .hash section\n"
@ MSG_USAGE_DETAIL8     "\t[-H]\t\tdump the contents of the .SUNW_cap section\n"
@ MSG_USAGE_DETAIL9     "\t[-i]\t\tdump the contents of the .interp section\n"
@ MSG_USAGE_DETAIL10    "\t[-I index]\tqualify an option with an index\n"
@ MSG_USAGE_DETAIL11    "\t[-l]\t\tdump with no truncated section names\n"
@ MSG_USAGE_DETAIL12    "\t[-k]\t\tcalculate elf checksum\n"
@ MSG_USAGE_DETAIL13    "\t[-m]\t\tdump the contents of the .SUNW_move \
                         section\n"
@ MSG_USAGE_DETAIL14    "\t[-n]\t\tdump the contents of the .note section\n"
@ MSG_USAGE_DETAIL15    "\t[-N name]\tqualify an option with a 'name'\n"
@ MSG_USAGE_DETAIL16    "\t[-O osabi]\tuse given osabi to interpret object\n"
@ MSG_USAGE_DETAIL17    "\t[-P]\t\tuse program headers to generate \
                         section headers\n"
@ MSG_USAGE_DETAIL18    "\t[-p]\t\tdump the program headers\n"
@ MSG_USAGE_DETAIL19    "\t[-r]\t\tdump the contents of the relocation \
                         sections\n"
@ MSG_USAGE_DETAIL20    "\t[-S]\t\tdump the contents of the sort index \
                         sections\n"
@ MSG_USAGE_DETAIL21    "\t[-s]\t\tdump the contents of the symbol table \
                         sections\n"
@ MSG_USAGE_DETAIL22    "\t[-T type]\tqualify an option with a section or \
                         program header type\n"
@ MSG_USAGE_DETAIL23    "\t[-u]\t\tdump the contents of a frame unwind \
                         section\n"
@ MSG_USAGE_DETAIL24    "\t[-v]\t\tdump the contents of the version sections\n"
@ MSG_USAGE_DETAIL25    "\t[-w file]\twrite the contents of specified section \
                         to 'file'\n"
@ MSG_USAGE_DETAIL26    "\t[-y]\t\tdump the contents of the .SUNW_syminfo \
                         section\n"

# Errors

@ MSG_ERR_BADFILE       "%s: invalid file type\n"
@ MSG_ERR_BADREL1       "%s: bad relocation entry: %s: relocation requires \
                         symbol\n"
@ MSG_ERR_NOSHDR        "%s: section headers are not present: synthesizing \
                         from program headers (-P option)\n"
@ MSG_ERR_PNEEDSPH      "%s: file without program headers is incompatible \
                         with -P option\n"
@ MSG_ERR_NOSHSTRSEC    "%s: no header string table section (shstrtab). \
                         header names are not available\n"
@ MSG_ERR_SHSTRNULL     "%s: zero size section header string table \
                         (shstrtab). header names are not available\n"
@ MSG_ERR_FAILURE       "%s: %s failed: %s\n"
@ MSG_ERR_BADSHDRALIGN  "%s: bad section header array alignment\n"
@ MSG_ERR_BADPHDRALIGN  "%s: bad program header array alignment\n"
@ MSG_ERR_BADSHNAME     "%s: section[%d]: invalid sh_name: %lld\n"
@ MSG_ERR_BADSHLINK     "%s: %s: invalid sh_link: %d\n"
@ MSG_ERR_BADSHINFO     "%s: %s: invalid sh_info: %d\n"
@ MSG_ERR_BADSHTYPE     "%s: %s: invalid sh_type: %d\n"
@ MSG_ERR_BADALIGN      "%s: %s: bad sh_offset alignment\n"
@ MSG_ERR_BADSYM2       "%s: %s: index[%d]: bad symbol entry: %s: must be \
                         SHN_COMMON or defined in SHT_NOBITS section\n"
@ MSG_ERR_BADSYM3       "%s: %s: index[%d]: bad symbol entry: %s: must be \
                         defined in a SHF_TLS section\n"
@ MSG_ERR_BADSYM4       "%s: %s: index[%d]: bad symbol entry: %s: must be \
                         defined in a non-SHF_TLS section\n"
@ MSG_ERR_BADSYM5       "%s: %s: index[%d]: bad symbol entry: %s: \
                         invalid shndx: %d\n"
@ MSG_ERR_BADSYM6       "%s: %s: index[%d]: bad symbol entry: %s: section[%d] \
                         size: %#llx: symbol (address %#llx, size %#llx) \
                         lies outside of containing section\n"
@ MSG_ERR_BADSYM7       "%s: %s: index[%d]: suspicious global symbol entry: \
                         %s: lies within local symbol range (index < %lld)\n"
@ MSG_ERR_BADSYM8       "%s: %s: index[%d]: suspicious local symbol entry: \
                         %s: lies within global symbol range (index >= %lld)\n"

@ MSG_ERR_RELBADSYMNDX  "%s: bad symbol reference %d: from relocation \
                         entry: %d\n"
@ MSG_ERR_HSBADSYMNDX   "%s: bad symbol reference %d: from hash entry: %d\n"
@ MSG_ERR_BADSYMXINDEX1 "%s: %s: index[%d]: invalid SHN_XINDEX reference: \
                         SHT_SYMTAB_SHNDX section truncated: no entry for this \
                         symbol\n"
@ MSG_ERR_BADSYMXINDEX2 "%s: %s: index[%d]: invalid SHN_XINDEX reference: \
                         bad SHT_SYMTAB_SHNDX entry: invalid shndx: 0x%x\n"
@ MSG_ERR_BADSYMXINDEX3 "%s: %s: index[%d]: invalid SHN_XINDEX reference: \
                         SHT_SYMTAB_SHNDX section not found\n"

@ MSG_ERR_BADSTOFF      "%s: %s: index[%d]: bad %s offset: 0x%x: max 0x%x\n"
@ MSG_ERR_BADSZ         "%s: %s: zero size or zero entry size information\n"
@ MSG_ERR_BADMINFO      "%s: %s: invalid m_info: 0x%llx\n"
```

```
125 @ MSG_ERR_BADHASH       "%s: %s: bad hash entry: symbol %s: exists in bucket \
126                          %d, should be bucket %ld\n"
127 @ MSG_ERR_NODYNSYM      "%s: %s: associated SHT_DYNSYM section not found\n"
128 @ MSG_ERR_BADNDXSEC     "%s: %s: unexpected section type associated with \
129                          index section: %s\n"
130 @ MSG_ERR_BADSYMNDX     "%s: %s: bad symbol index: %d\n"
131 @ MSG_ERR_BADVER        "%s: %s: index[%d]: version %d is out of range: \
132                          version definitions available: 0-%d\n"
133 @ MSG_ERR_NOTSTRTAB     "%s: section[%d] is not a string table as expected \
134                          by section[%d]\n";

136 @ MSG_ERR_LDYNNOTADJ    "%s: bad dynamic symbol table layout: %s and %s \
137                          sections are not adjacent\n"
138 @ MSG_ERR_SECMEMOVER    "%s: memory overlap between section[%d]: %s: %llx:%llx \
139                          and section[%d]: %s: %llx:%llx\n"
140 @ MSG_ERR_SHDRMEMOVER   "%s: memory overlap between section header table: \
141                          %llx:%llx and section[%d]: %s: %llx:%llx\n"
142 @ MSG_ERR_MULTDYN       "%s: %d dynamic sections seen (1 expected)\n"
143 @ MSG_ERR_DYNNOBCKSEC   "%s: object lacks %s section required by %s dynamic \
144                          entry\n"
145 @ MSG_ERR_DYNBADADDR    "%s: %s (%#llx) does not match \
146                          shdr[%d: %s].sh_addr (%#llx)\n"
147 @ MSG_ERR_DYNBADSIZE    "%s: %s (%#llx) does not match \
148                          shdr[%d: %s].sh_size (%#llx)\n"
149 @ MSG_ERR_DYNBADENTSIZE "%s: %s (%#llx) does not match \
150                          shdr[%d: %s].sh_entsize (%#llx)\n"
151 @ MSG_ERR_DYNSYMVAL     "%s: %s: symbol value does not match \
152                          %s entry: %s: value: %#llx\n"
153 @ MSG_ERR_MALSTR        "%s: %s: malformed string table, initial or final \
154                          byte\n"
155 @ MSG_ERR_MULTEHFRMHDR  "%s: [%d: %s] multiple .eh_frame_hdr sections seen \
156                          (1 expected)\n"
157 @ MSG_ERR_BADEHFRMPTR   "%s: section[%d: %s] FramePtr (%#llx) does not match \
158                          shdr[%d: %s].sh_addr (%#llx)\n"
159 @ MSG_ERR_BADSORT       "%s: %s: index[%d]: invalid sort order\n"
160 @ MSG_ERR_BADSIDYNNDX   "%s: [%d: %s][%d]: dynamic section index out of \
161                          range (0 - %d): %d\n";
162 @ MSG_ERR_BADSIDYNTAG   "%s: [%d: %s][%d]: dynamic element \
163                          [%d: %s][%d] should have type %s: %s\n";
164 @ MSG_ERR_BADCIEFDELEN  "%s: %s: invalid CIE/FDE length: %#llx at %#llx\n"


167 @ MSG_WARN_INVINTERP1   "%s: PT_INTERP header has no associated section\n"
168 @ MSG_WARN_INVINTERP2   "%s: interp section: %s: and PT_INTERP program \
169                          header have conflicting size or offsets\n"
170 @ MSG_WARN_INVCAP1      "%s: PT_SUNWCAP header has no associated section\n"
171 @ MSG_WARN_INVCAP2      "%s: capabilities section[%d]: %s: requires PT_CAP \
172                          program header\n"
173 @ MSG_WARN_INVCAP3      "%s: capabilities section[%d]: %s: and PT_CAP program \
174                          header have conflicting size or offsets\n"
175 @ MSG_WARN_INVCAP4      "%s: capabilities section[%d]: %s: requires string \
176                          table: invalid sh_info: %d\n";
177 @ MSG_WARN_INADDR32SF1  "%s: capabilities section %s: software capability \
178                          ADDR32: is ineffective within a 32-bit object\n"
179 @ MSG_WARN_MULTEHFRM    "%s: section[%d: %s]: %s object has multiple \
180                         .eh_frame sections\n"

182 @ MSG_INFO_LINUXOSABI   "%s: %s object has Linux .note.ABI-tag section. \
183                         Assuming %s\n"

185 @ MSG_ERR_DWOVRFLW      "%s: %s: encoded DWARF data exceeds section size\n"
186 @ MSG_ERR_DWBADENC      "%s: %s: bad DWARF encoding: %#x\n"
187 @ MSG_ERR_DWNOCIE       "%s: %s: no CIE prior to FDE\n"

189 # exception_range_entry table entries.
190 # TRANSLATION_NOTE - the following entries provide for a series of one or more
```



```
191 # standard 32-bit and 64-bit .exception_ranges table entries that align with
192 # the initial title.

194 @ MSG_EXR_TITLE_32      "     index    offset     ret_addr   \
195                          length     handler    type_blk"
196 @ MSG_EXR_ENTRY_32      "%10.10s  0x%8.8llx 0x%8.8llx 0x%8.8llx 0x%8.8llx \
197                          0x%8.8llx"
198 @ MSG_EXR_TITLE_64      "     index    offset             ret_addr           \
199                          length             handler            type_blk"
200 @ MSG_EXR_ENTRY_64      "%10.10s  0x%16.16llx 0x%16.16llx 0x%16.16llx \
201                          0x%16.16llx 0x%16.16llx"

203 # Elf Output Messages

205 @ MSG_ELF_SHDR          "Section Header[%d]:  sh_name: %s"
206 @ MSG_ELF_PHDR          "Program Header[%d]:"

208 @ MSG_ELF_SCN_CAP       "Capabilities Section:  %s"
209 @ MSG_ELF_SCN_CAPCHAIN  "Capabilities Chain Section:  %s"
210 @ MSG_ELF_SCN_INTERP    "Interpreter Section:  %s"
211 @ MSG_ELF_SCN_VERDEF    "Version Definition Section:  %s"
212 @ MSG_ELF_SCN_VERNEED   "Version Needed Section:  %s"
213 @ MSG_ELF_SCN_SYMTAB    "Symbol Table Section:  %s"
214 @ MSG_ELF_SCN_RELOC     "Relocation Section:  %s"
215 @ MSG_ELF_SCN_UNWIND    "Unwind Section: %s"
216 @ MSG_ELF_SCN_DYNAMIC   "Dynamic Section:  %s"
217 @ MSG_ELF_SCN_NOTE      "Note Section:  %s"
218 @ MSG_ELF_SCN_HASH      "Hash Section:  %s"
219 @ MSG_ELF_SCN_SYMINFO   "Syminfo Section:  %s"
220 @ MSG_ELF_SCN_GOT       "Global Offset Table Section:  %s"
221 @ MSG_ELF_SCN_GRP       "Group Section:  %s"
222 @ MSG_ELF_SCN_MOVE      "Move Section:  %s"
223 @ MSG_ELF_SCN_SYMSORT1  "Symbol Sort Section:  %s (%s)"
224 @ MSG_ELF_SCN_SYMSORT2  "Symbol Sort Section:  %s (%s / %s)"

226 @ MSG_OBJ_CAP_TITLE     " Object Capabilities:"
227 @ MSG_SYM_CAP_TITLE     " Symbol Capabilities:"
228 @ MSG_CAPINFO_ENTRIES   "  Symbols:"
229 @ MSG_CAPCHAIN_TITLE    " Capabilities family: %s"
230 @ MSG_CAPCHAIN_ENTRY    "  chainndx  symndx      name"
231 @ MSG_ERR_INVCAP        "%s: capabilities section: %s: contains symbol \
232                          capabilities groups, but no capabilities information \
233                          section is defined: invalid sh_link: %d\n"
234 @ MSG_ERR_INVCAPINFO1   "%s: capabilities information section: %s: no symbol \
235                          table is defined: invalid sh_link: %d\n"
236 @ MSG_ERR_INVCAPINFO2   "%s: capabilities information section: %s: no \
237                          capabilities chain is defined: invalid sh_info: %d\n"
238 @ MSG_ERR_INVCAPINFO3   "%s: capabilities information section: %s: index %d: \
239                          bad capabilities chain index defined: %d\n"
240 @ MSG_ERR_CHBADSYMNDX   "%s: bad symbol reference %d: from capability chain: \
241                          %s entry: %d\n"

243 @ MSG_ELF_HASH_BKTS1    "%10.10s  buckets contain %8d symbols"
244 @ MSG_ELF_HASH_BKTS2    "%10.10s  buckets         %8d symbols (globals)"
245 @ MSG_ELF_HASH_INFO     "    bucket  symndx      name"
246 @ MSG_HASH_OVERFLW      "%s: warning: section %s: too many symbols to count, \
247                          bucket=%d count=%d"
248 @ MSG_ELF_ERR_SHDR      "\tunable to obtain section header: shstrtab[%lld]\n"
249 @ MSG_ELF_ERR_DATA      "\tunable to obtain section data: shstrtab[%lld]\n"
250 @ MSG_ELF_ERR_SCN       "\tunable to obtain section header: section[%d]\n"
251 @ MSG_ELF_ERR_SCNDATA   "\tunable to obtain section data: section[%d]\n"
252 @ MSG_ARCHIVE_SYMTAB_32 "\nSymbol Table: (archive, 32-bit offsets)"
253 @ MSG_ARCHIVE_SYMTAB_64 "\nSymbol Table: (archive, 64-bit offsets)"
254 @ MSG_ARCHIVE_FIELDS_32 "     index    offset    member name and symbol"
255 @ MSG_ARCHIVE_FIELDS_64 "     index    offset            member name and symbol"
```

```
257 @ MSG_GOT_MULTIPLE      "%s: multiple relocations against \
258                          the same GOT entry ndx: %d addr: 0x%llx\n"
259 @ MSG_GOT_UNEXPECTED    "%s: warning: section %s: section unexpected within \
260                          relocatable object\n"

262 # Miscellaneous clutter

264 @ MSG_STR_NULL          "(null)"
265 @ MSG_STR_DEPRECATED    "(deprecated value)"
266 @ MSG_STR_UNKNOWN       "<unknown>"
267 @ MSG_STR_SECTION       "%s (section)"
268 @ MSG_STR_CHECKSUM      "elf checksum: 0x%lx"

270 @ MSG_FMT_SCNNDX        "section[%d]"
271 @ MSG_FMT_NOTEENTNDX    "  entry [%d]";


274 @ MSG_ERR_MALLOC        "%s: malloc: %s\n"
275 @ MSG_ERR_OPEN          "%s: open: %s\n"
276 @ MSG_ERR_READ          "%s: read: %s\n"
277 @ MSG_ERR_WRITE         "%s: write: %s\n"
278 @ MSG_ERR_BAD_T_SHT     "%s: unrecognized section header type: %s\n"
279 @ MSG_ERR_BAD_T_PT      "%s: unrecognized program header type: %s\n"
280 @ MSG_ERR_BAD_T_OSABI   "%s: unrecognized operating system ABI: %s\n"
281 @ MSG_ERR_AMBIG_MATCH   "%s: ambiguous use of -I, -N, or -T. Remove \
282                          -p option or section selection option(s)\n"

284 #
285 # SHT_MOVE messages
286 #
287 @ MSG_MOVE_TITLE        "    symndx     offset   size repeat stride           \
288                          value  with respect to"
289 @ MSG_MOVE_ENTRY        "%10.10s %#10llx %6d %6d %6d %#16llx  %s"

291 #
292 # SHT_GROUP messages
293 #
294 @ MSG_GRP_TITLE         "     index    flags / section         signature symbol"
295 @ MSG_GRP_SIGNATURE     "       [0]   %-24s %s"
296 @ MSG_GRP_INVALSCN      "<invalid section>"

298 #
299 # SHT_NOTE messages
300 #
301 @ MSG_NOTE_BADDATASZ    "%s: %s: note header exceeds section size. \
302                          offset: 0x%x\n"
303 @ MSG_NOTE_BADNMSZ      "%s: %s: note name value exceeds section size. \
304                          offset: 0x%x namesize: 0x%x\n"
305 @ MSG_NOTE_BADDESZ      "%s: %s: note data size exceeds section size. \
306                          offset: 0x%x datasize: 0x%x\n"
307 @ MSG_NOTE_BADCOREARCH  "%s: elfdump core file note support not available for \
308                          architecture: %s\n"
309 @ MSG_NOTE_BADCOREDATA  "%s: elfdump core file note data truncated or \
310                          otherwise malformed\n"
311 @ MSG_NOTE_BADCORETYPE  "%s: unknown note type %#x\n"

313 @ _END_

315 # The following strings represent reserved words, files, pathnames and symbols.
316 # Reference to this strings is via the MSG_ORIG() macro, and thus no message
317 # translation is required.

319 @ MSG_STR_OSQBRKT       "["
320 @ MSG_STR_CSQBRKT       "]"

322 @ MSG_GRP_COMDAT        " COMDAT "
```



```
323 @ MSG_GRP_ENTRY         "%10.10s   %s [%lld]\n"
324 @ MSG_GRP_UNKNOWN       " 0x%x "

326 @ MSG_ELF_GOT           ".got"
327 @ MSG_ELF_INIT          ".init"
328 @ MSG_ELF_FINI          ".fini"
329 @ MSG_ELF_INTERP        ".interp"

331 @ MSG_ELF_GETEHDR       "elf_getehdr"
332 @ MSG_ELF_GETPHDR       "elf_getphdr"
333 @ MSG_ELF_GETSHDR       "elf_getshdr"
334 @ MSG_ELF_GETSCN        "elf_getscn"
335 @ MSG_ELF_GETDATA       "elf_getdata"
336 @ MSG_ELF_GETARHDR      "elf_getarhdr"
337 @ MSG_ELF_GETARSYM      "elf_getarsym"
338 @ MSG_ELF_RAND          "elf_rand"
339 @ MSG_ELF_BEGIN         "elf_begin"
340 @ MSG_ELF_GETPHDRNUM    "elf_getphdrnum"
341 @ MSG_ELF_GETSHDRNUM    "elf_getshdrnum"
342 @ MSG_ELF_GETSHDRSTRNDX "elf_getshdrstrndx"
343 @ MSG_ELF_XLATETOM      "elf_xlatetom"
344 @ MSG_ELF_ARSYM         "ARSYM"

346 @ MSG_SYM_INIT          "_init"
347 @ MSG_SYM_FINI          "_fini"
348 @ MSG_SYM_GOT           "_GLOBAL_OFFSET_TABLE_"

350 @ MSG_STR_OPTIONS       "CcdeGgHhiI:klmN:nO:PprSsT:uvw:y"

352 @ MSG_STR_8SP          "        "
353 @ MSG_STR_EMPTY        ""
354 @ MSG_STR_CORE         "CORE"
355 @ MSG_STR_NOTEABITAG    ".note.ABI-tag"
356 @ MSG_STR_GNU           "GNU"
357 @ MSG_STR_LOC           "loc"
358 @ MSG_STR_INITLOC       "initloc"

360 @ MSG_FMT_INDENT        "       %s"
361 @ MSG_FMT_INDEX         " [%lld]"
362 @ MSG_FMT_INDEX2        "[%d]"
363 @ MSG_FMT_ASRINDEX      "[ asr%d ]"
364 @ MSG_FMT_INDEXRNG      "[%d-%d]"
365 @ MSG_FMT_INTEGER       " %d"
366 @ MSG_FMT_HASH_INFO     "%10.10s  %-10s  %s"
367 @ MSG_FMT_CHAIN_INFO    "%10.10s  %-10s  %s"
368 @ MSG_FMT_ARSYM1_32     "%10.10s  0x%8.8llx  (%s):%s"
369 @ MSG_FMT_ARSYM2_32     "%10.10s  0x%8.8llx"
370 @ MSG_FMT_ARSYM1_64     "%10.10s  0x%16.16llx  (%s):%s"
371 @ MSG_FMT_ARSYM2_64     "%10.10s  0x%16.16llx"
372 @ MSG_FMT_ARNAME        "%s(%s)"
373 @ MSG_FMT_NLSTR         "\n%s:"
374 @ MSG_FMT_NLSTRNL       "\n%s:\n"
375 @ MSG_FMT_SECSYM        "%.*s%s"

377 @ MSG_HEXDUMP_ROW       "%*s%-*s%s"
378 @ MSG_HEXDUMP_TOK       "%2.2x"

380 @ MSG_SUNW_OST_SGS      "SUNW_OST_SGS"

382 # Unwind info

384 @ MSG_SCN_FRM           ".eh_frame"
385 @ MSG_SCN_FRMHDR        ".eh_frame_hdr"
386 @ MSG_SCN_EXRANGE       ".exception_ranges"

388 @ MSG_UNW_FRMHDR        "Frame Header:"
```

```
389 @ MSG_UNW_FRMVERS       "  Version: %d"
390 @ MSG_UNW_FRPTRENC      "  FramePtrEnc: %-20s  FramePtr: %#llx"
391 @ MSG_UNW_FDCNENC       "  FdeCntEnc:   %-20s  FdeCnt: %lld"
392 @ MSG_UNW_TABENC        "  TableEnc:    %-20s"
393 @ MSG_UNW_BINSRTAB1     "  Binary Search Table:"
394 @ MSG_UNW_BINSRTAB2_32  "      InitialLoc    FdeLoc"
395 @ MSG_UNW_BINSRTAB2_64  "      InitialLoc          FdeLoc"
396 @ MSG_UNW_BINSRTABENT_32 "    0x%08llx    0x%08llx"
397 @ MSG_UNW_BINSRTABENT_64 "    0x%016llx  0x%016llx"
398 @ MSG_UNW_ZEROTERM      "ZERO terminator: [0x00000000]"
399 @ MSG_UNW_CIE           "CIE: [%#llx]"
400 @ MSG_UNW_CIELNGTH      " length: 0x%02x cieid: %d"
401 @ MSG_UNW_CIEVERS       " version: %d  augmentation: '%s'"
402 @ MSG_UNW_CIECALGN      " codealign: %#llx  dataalign: %lld  \
403                         retaddr: %d"
404 @ MSG_UNW_CIEAXVAL      " Augmentation Data:"
405 @ MSG_UNW_CIEAXSIZ      "   size: %lld"
406 @ MSG_UNW_CIEAXPERS     "   personality:"
407 @ MSG_UNW_CIEAXPERSENC  "       encoding: 0x%02x %s"
408 @ MSG_UNW_CIEAXPERSRTN  "       routine:  %#08llx"
409 @ MSG_UNW_CIEAXCENC     "   code pointer encoding: 0x%02x %s"
410 @ MSG_UNW_CIEAXLSDA     "   lsda encoding: 0x%02x %s"
411 @ MSG_UNW_CIEAXUNEC     "   Unexpected aug val: %c"
412 @ MSG_UNW_CIECFI        " CallFrameInstructions:"

414 @ MSG_UNW_FDE           " FDE: [%#llx]"
415 @ MSG_UNW_FDELNGTH      "    length:  %#x  cieptr: %#x"
416 @ MSG_UNW_FDEINITLOC    "    initloc: %#llx  addrrange: %#llx  endloc: %#llx"
417 @ MSG_UNW_FDEAXVAL      "   Augmentation Data:"
418 @ MSG_UNW_FDEAXSIZE     "      size: %#llx"
419 @ MSG_UNW_FDEAXLSDA     "      lsda: %#llx"
420 @ MSG_UNW_FDECFI        "   CallFrameInstructions:"

422 # Unwind section Call Frame Instructions. These all start with a leading
423 # "%*s%s", used to insert leading white space and the opcode name.

425 @ MSG_CFA_ADV_LOC       "%*s%s: %s + %llu => %#llx"
426 @ MSG_CFA_CFAOFF        "%*s%s: %s, cfa%+lld"
427 @ MSG_CFA_CFASET        "%*s%s: cfa=%#llx"
428 @ MSG_CFA_LLD           "%*s%s: %lld"
429 @ MSG_CFA_LLU           "%*s%s: %llu"
430 @ MSG_CFA_REG           "%*s%s: %s"
431 @ MSG_CFA_REG_OFFLLD    "%*s%s: %s, offset=%lld"
432 @ MSG_CFA_REG_OFFLLU    "%*s%s: %s, offset=%llu"
433 @ MSG_CFA_REG_REG       "%*s%s: %s, %s"
434 @ MSG_CFA_SIMPLE        "%*s%s"
435 @ MSG_CFA_SIMPLEREP     "%*s%s [%d]"
436 @ MSG_CFA_EBLK          "%*s%s: expr(%llu bytes)"
437 @ MSG_CFA_REG_EBLK      "%*s%s: %s, expr(%llu bytes)"

439 # Architecture specific register name formats

441 @ MSG_REG_FMT_BASIC     "r%d"
442 @ MSG_REG_FMT_NAME      "r%d (%s)"


445 # Note messages

447 @ MSG_NOTE_TYPE         "    type:   %#x"
448 @ MSG_NOTE_TYPE_STR     "    type:   %s"
449 @ MSG_NOTE_NAMESZ       "    namesz: %#x"
450 @ MSG_NOTE_NAME         "    name:"
451 @ MSG_NOTE_DESCSZ       "    descsz: %#x"

453 @ MSG_NOTE_DESC                         "    desc:"
454 @ MSG_CNOTE_DESC_ASRSET_T              "desc: (asrset_t)"
```



```
455 @ MSG_CNOTE_DESC_AUXV_T                "desc: (auxv_t)"
456 @ MSG_CNOTE_DESC_CORE_CONTENT_T        "desc: (core_content_t)"
457 @ MSG_CNOTE_DESC_LWPSINFO_T            "desc: (lwpsinfo_t)"
458 @ MSG_CNOTE_DESC_LWPSTATUS_T           "desc: (lwpstatus_t)"
459 @ MSG_CNOTE_DESC_PRCRED_T              "desc: (prcred_t)"
460 @ MSG_CNOTE_DESC_PRIV_IMPL_INFO_T      "desc: (priv_impl_info_t)"
461 @ MSG_CNOTE_DESC_PRPRIV_T              "desc: (prpriv_t)"
462 @ MSG_CNOTE_DESC_PRPSINFO_T            "desc: (prpsinfo_t)"
463 @ MSG_CNOTE_DESC_PRSTATUS_T            "desc: (prstatus_t)"
464 @ MSG_CNOTE_DESC_PSINFO_T              "desc: (psinfo_t)"
465 @ MSG_CNOTE_DESC_PSTATUS_T             "desc: (pstatus_t)"
466 @ MSG_CNOTE_DESC_STRUCT_UTSNAME        "desc: (struct utsname)"
467 @ MSG_CNOTE_DESC_PRFDINFO_T            "desc: (prfdinfo_t)"


470 @ MSG_CNOTE_FMT_LINE            "%*s%-*s%s"
471 @ MSG_CNOTE_FMT_LINE_2UP        "%*s%-*s%-*s%-*s%s"
472 @ MSG_CNOTE_FMT_D               "%d"
473 @ MSG_CNOTE_FMT_LLD             "%lld"
474 @ MSG_CNOTE_FMT_U               "%u"
475 @ MSG_CNOTE_FMT_LLU             "%llu"
476 @ MSG_CNOTE_FMT_X               "%#x"
477 @ MSG_CNOTE_FMT_LLX             "%#llx"
478 @ MSG_CNOTE_FMT_Z2X             "0x%2.2x"
479 @ MSG_CNOTE_FMT_Z4X             "0x%4.4x"
480 @ MSG_CNOTE_FMT_Z8X             "0x%8.8x"
481 @ MSG_CNOTE_FMT_Z16LLX          "0x%16.16llx"
482 @ MSG_CNOTE_FMT_TITLE           "%*s%s"
483 @ MSG_CNOTE_FMT_AUXVLINE        "%*s%10.10s  %-*s  %s"
484 @ MSG_CNOTE_FMT_PRTPCT          "%u.%u%%"

486 @ MSG_CNOTE_T_PRIV_FLAGS               "priv_flags:"
487 @ MSG_CNOTE_T_PRIV_GLOBALINFOSIZE      "priv_globalinfosize:"
488 @ MSG_CNOTE_T_PRIV_HEADERSIZE          "priv_headersize:"
489 @ MSG_CNOTE_T_PRIV_INFOSIZE            "priv_infosize:"
490 @ MSG_CNOTE_T_PRIV_MAX                 "priv_max:"
491 @ MSG_CNOTE_T_PRIV_NSETS               "priv_nsets:"
492 @ MSG_CNOTE_T_PRIV_SETSIZE             "priv_setsize:"
493 @ MSG_CNOTE_T_PR_ACTION                "pr_action:"
494 @ MSG_CNOTE_T_PR_ADDR                  "pr_addr:"
495 @ MSG_CNOTE_T_PR_AGENTID               "pr_agentid:"
496 @ MSG_CNOTE_T_PR_ALTSTACK              "pr_altstack:"
497 @ MSG_CNOTE_T_PR_ARGC                  "pr_argc:"
498 @ MSG_CNOTE_T_PR_ARGV                  "pr_argv:"
499 @ MSG_CNOTE_T_PR_ASLWPID               "pr_aslwpid:"
500 @ MSG_CNOTE_T_PR_BIND                  "pr_bind:"
501 @ MSG_CNOTE_T_PR_BINDPRO               "pr_bindpro:"
502 @ MSG_CNOTE_T_PR_BINDPSET              "pr_bindpset:"
503 @ MSG_CNOTE_T_PR_BRKBASE               "pr_brkbase:"
504 @ MSG_CNOTE_T_PR_BRKSIZE               "pr_brksize:"
505 @ MSG_CNOTE_T_PR_BYRSSIZE              "pr_byrssize:"
506 @ MSG_CNOTE_T_PR_BYSIZE                "pr_bysize:"
507 @ MSG_CNOTE_T_PR_CLNAME                "pr_clname:"
508 @ MSG_CNOTE_T_PR_CONTRACT              "pr_contract:"
509 @ MSG_CNOTE_T_PR_CPU                   "pr_cpu:"
510 @ MSG_CNOTE_T_PR_CSTIME                "pr_cstime:"
511 @ MSG_CNOTE_T_PR_CTIME                 "pr_ctime:"
512 @ MSG_CNOTE_T_PR_CURSIG                "pr_cursig:"
513 @ MSG_CNOTE_T_PR_CUTIME                "pr_cutime:"
514 @ MSG_CNOTE_T_PR_DMODEL                "pr_dmodel:"
515 @ MSG_CNOTE_T_PR_EGID                  "pr_egid:"
516 @ MSG_CNOTE_T_PR_ENVP                  "pr_envp:"
517 @ MSG_CNOTE_T_PR_ERRNO                 "pr_errno:"
518 @ MSG_CNOTE_T_PR_ERRPRIV               "pr_errpriv:"
519 @ MSG_CNOTE_T_PR_EUID                  "pr_euid:"
520 @ MSG_CNOTE_T_PR_FLAG                  "pr_flag:"
```

```
@ MSG_CNOTE_T_PR_FLAGS                  "pr_flags:"
@ MSG_CNOTE_T_PR_FLTTRACE               "pr_flttrace:"
@ MSG_CNOTE_T_PR_FNAME                  "pr_fname:"
@ MSG_CNOTE_T_PR_FPREG                  "pr_fpreg:"
@ MSG_CNOTE_T_PR_GID                    "pr_gid:"
@ MSG_CNOTE_T_PR_GROUPS                 "pr_groups:"
@ MSG_CNOTE_T_PR_INFO                   "pr_info:"
@ MSG_CNOTE_T_PR_INFOSIZE               "pr_infosize:"
@ MSG_CNOTE_T_PR_INSTR                  "pr_instr:"
@ MSG_CNOTE_T_PR_LGRP                   "pr_lgrp:"
@ MSG_CNOTE_T_PR_LTTYDEV                "pr_lttydev:"
@ MSG_CNOTE_T_PR_LWP                    "pr_lwp:"
@ MSG_CNOTE_T_PR_LWPHOLD                "pr_lwphold:"
@ MSG_CNOTE_T_PR_LWPID                  "pr_lwpid:"
@ MSG_CNOTE_T_PR_LWPPEND                "pr_lwppend:"
@ MSG_CNOTE_T_PR_NAME                   "pr_name:"
@ MSG_CNOTE_T_PR_NGROUPS                "pr_ngroups:"
@ MSG_CNOTE_T_PR_NICE                   "pr_nice:"
@ MSG_CNOTE_T_PR_NLWP                   "pr_nlwp:"
@ MSG_CNOTE_T_PR_NSETS                  "pr_nsets:"
@ MSG_CNOTE_T_PR_NSYSARG                "pr_nsysarg:"
@ MSG_CNOTE_T_PR_NZOMB                  "pr_nzomb:"
@ MSG_CNOTE_T_PR_OLDCONTEXT             "pr_oldcontext:"
@ MSG_CNOTE_T_PR_OLDPRI                 "pr_oldpri:"
@ MSG_CNOTE_T_PR_ONPRO                  "pr_onpro:"
@ MSG_CNOTE_T_PR_OTTYDEV                "pr_ottydev:"
@ MSG_CNOTE_T_PR_PCTCPU                 "pr_pctcpu:"
@ MSG_CNOTE_T_PR_PCTMEM                 "pr_pctmem:"
@ MSG_CNOTE_T_PR_PGID                   "pr_pgid:"
@ MSG_CNOTE_T_PR_PGRP                   "pr_pgrp:"
@ MSG_CNOTE_T_PR_PID                    "pr_pid:"
@ MSG_CNOTE_T_PR_POOLID                 "pr_poolid:"
@ MSG_CNOTE_T_PR_PPID                   "pr_ppid:"
@ MSG_CNOTE_T_PR_PRI                    "pr_pri:"
@ MSG_CNOTE_T_PR_PROCESSOR              "pr_processor:"
@ MSG_CNOTE_T_PR_PROJID                 "pr_projid:"
@ MSG_CNOTE_T_PR_PSARGS                 "pr_psargs:"
@ MSG_CNOTE_T_PR_REG                    "pr_reg:"
@ MSG_CNOTE_T_PR_RGID                   "pr_rgid:"
@ MSG_CNOTE_T_PR_RSSIZE                 "pr_rssize:"
@ MSG_CNOTE_T_PR_RUID                   "pr_ruid:"
@ MSG_CNOTE_T_PR_RVAL1                  "pr_rval1:"
@ MSG_CNOTE_T_PR_RVAL2                  "pr_rval2:"
@ MSG_CNOTE_T_PR_SECFLAGS               "pr_secflags:"
#endif /* ! codereview */
@ MSG_CNOTE_T_PR_SETS                   "pr_sets:"
@ MSG_CNOTE_T_PR_SETSIZE                "pr_setsize:"
@ MSG_CNOTE_T_PR_SGID                   "pr_sgid:"
@ MSG_CNOTE_T_PR_SID                    "pr_sid:"
@ MSG_CNOTE_T_PR_SIGHOLD                "pr_sighold:"
@ MSG_CNOTE_T_PR_SIGPEND                "pr_sigpend:"
@ MSG_CNOTE_T_PR_SIGTRACE               "pr_sigtrace:"
@ MSG_CNOTE_T_PR_SIZE                   "pr_size:"
@ MSG_CNOTE_T_PR_SNAME                  "pr_sname:"
@ MSG_CNOTE_T_PR_START                  "pr_start:"
@ MSG_CNOTE_T_PR_STATE                  "pr_state:"
@ MSG_CNOTE_T_PR_STIME                  "pr_stime:"
@ MSG_CNOTE_T_PR_STKBASE                "pr_stkbase:"
@ MSG_CNOTE_T_PR_STKSIZE                "pr_stksize:"
@ MSG_CNOTE_T_PR_STYPE                  "pr_stype:"
@ MSG_CNOTE_T_PR_SUID                   "pr_suid:"
@ MSG_CNOTE_T_PR_SYSARG                 "pr_sysarg:"
@ MSG_CNOTE_T_PR_SYSCALL                "pr_syscall:"
@ MSG_CNOTE_T_PR_SYSENTRY               "pr_sysentry:"
@ MSG_CNOTE_T_PR_SYSEXIT                "pr_sysexit:"
@ MSG_CNOTE_T_PR_TASKID                 "pr_taskid:"
```



```
@ MSG_CNOTE_T_PR_TIME                   "pr_time:"
@ MSG_CNOTE_T_PR_TSTAMP                 "pr_tstamp:"
@ MSG_CNOTE_T_PR_TTYDEV                 "pr_ttydev:"
@ MSG_CNOTE_T_PR_UID                    "pr_uid:"
@ MSG_CNOTE_T_PR_USTACK                 "pr_ustack:"
@ MSG_CNOTE_T_PR_UTIME                  "pr_utime:"
@ MSG_CNOTE_T_PR_WCHAN                  "pr_wchan:"
@ MSG_CNOTE_T_PR_WHAT                   "pr_what:"
@ MSG_CNOTE_T_PR_WHO                    "pr_who:"
@ MSG_CNOTE_T_PR_WHY                    "pr_why:"
@ MSG_CNOTE_T_PR_WSTAT                  "pr_wstat:"
@ MSG_CNOTE_T_PR_ZOMB                   "pr_zomb:"
@ MSG_CNOTE_T_PR_ZONEID                 "pr_zoneid:"
@ MSG_CNOTE_T_PSF_EFFECTIVE             "psf_effective:"
@ MSG_CNOTE_T_PSF_INHERIT               "psf_inherit:"
#endif /* ! codereview */
@ MSG_CNOTE_T_SA_FLAGS                  "sa_flags:"
@ MSG_CNOTE_T_SA_HANDLER                "sa_handler:"
@ MSG_CNOTE_T_SA_MASK                   "sa_mask:"
@ MSG_CNOTE_T_SA_SIGACTION              "sa_sigaction:"
@ MSG_CNOTE_T_SIVAL_INT                 "sival_int:"
@ MSG_CNOTE_T_SIVAL_PTR                 "sival_ptr:"
@ MSG_CNOTE_T_SI_ADDR                   "si_addr:"
@ MSG_CNOTE_T_SI_BAND                   "si_band:"
@ MSG_CNOTE_T_SI_CODE                   "si_code:"
@ MSG_CNOTE_T_SI_CTID                   "si_ctid:"
@ MSG_CNOTE_T_SI_ENTITY                 "si_entity"
@ MSG_CNOTE_T_SI_ERRNO                  "si_errno:"
@ MSG_CNOTE_T_SI_PID                    "si_pid:"
@ MSG_CNOTE_T_SI_SIGNO                  "si_signo:"
@ MSG_CNOTE_T_SI_STATUS                 "si_status:"
@ MSG_CNOTE_T_SI_UID                    "si_uid:"
@ MSG_CNOTE_T_SI_VALUE                  "si_value:"
@ MSG_CNOTE_T_SI_ZONEID                 "si_zoneid:"
@ MSG_CNOTE_T_SS_FLAGS                  "ss_flags:"
@ MSG_CNOTE_T_SS_SIZE                   "ss_size:"
@ MSG_CNOTE_T_SS_SP                     "ss_sp:"
@ MSG_CNOTE_T_TV_NSEC                   "tv_nsec:"
@ MSG_CNOTE_T_TV_SEC                    "tv_sec:"
@ MSG_CNOTE_T_UTS_MACHINE               "machine:"
@ MSG_CNOTE_T_UTS_NODENAME              "nodename:"
@ MSG_CNOTE_T_UTS_RELEASE               "release:"
@ MSG_CNOTE_T_UTS_SYSNAME               "sysname:"
@ MSG_CNOTE_T_UTS_VERSION               "version:"
@ MSG_CNOTE_T_PR_FD                     "pr_fd:"
@ MSG_CNOTE_T_PR_MODE                   "pr_mode:"
@ MSG_CNOTE_T_PR_PATH                   "pr_path:"
@ MSG_CNOTE_T_PR_MAJOR                  "pr_major:"
@ MSG_CNOTE_T_PR_MINOR                  "pr_minor:"
@ MSG_CNOTE_T_PR_RMAJOR                 "pr_rmajor:"
@ MSG_CNOTE_T_PR_RMINOR                 "pr_rminor:"
@ MSG_CNOTE_T_PR_OFFSET                 "pr_offset:"
@ MSG_CNOTE_T_PR_INO                    "pr_ino:"
@ MSG_CNOTE_T_PR_FILEFLAGS              "pr_fileflags:"
@ MSG_CNOTE_T_PR_FDFLAGS                "pr_fdflags:"


# Names of fake sections generated from program header data
@ MSG_PHDRNAM_CAP               ".SUNW_cap(phdr)"
@ MSG_PHDRNAM_CAPINFO           ".SUNW_capinfo(phdr)"
@ MSG_PHDRNAM_CAPCHAIN          ".SUNW_capchain(phdr)"
@ MSG_PHDRNAM_DYN               ".dynamic(phdr)"
@ MSG_PHDRNAM_DYNSTR            ".dynstr(phdr)"
@ MSG_PHDRNAM_DYNSYM            ".dynsym(phdr)"
@ MSG_PHDRNAM_FINIARR           ".fini_array(phdr)"
@ MSG_PHDRNAM_HASH              ".hash(phdr)"
```

```
653 @ MSG_PHDRNAM_INITARR           ".init_array(phdr)"
654 @ MSG_PHDRNAM_INTERP            ".interp(phdr)"
655 @ MSG_PHDRNAM_LDYNSYM           ".SUNW_ldynsym(phdr)"
656 @ MSG_PHDRNAM_MOVE              ".move(phdr)"
657 @ MSG_PHDRNAM_NOTE              ".note(phdr)"
658 @ MSG_PHDRNAM_PREINITARR        ".preinit_array(phdr)"
659 @ MSG_PHDRNAM_REL               ".rel(phdr)"
660 @ MSG_PHDRNAM_RELA              ".rela(phdr)"
661 @ MSG_PHDRNAM_SYMINFO           ".syminfo(phdr)"
662 @ MSG_PHDRNAM_SYMSORT           ".SUNW_symsort(phdr)"
663 @ MSG_PHDRNAM_TLSSORT           ".SUNW_tlssort(phdr)"
664 @ MSG_PHDRNAM_UNWIND            ".eh_frame_hdr(phdr)"
665 @ MSG_PHDRNAM_VER               ".SUNW_version(phdr)"

667 #endif /* ! codereview */
```

```
**********************************************************
   24980 Wed May 27 19:48:59 2015
new/usr/src/cmd/sgs/elfdump/common/gen_struct_layout.c
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
______unchanged_portion_omitted_


 169 /* pstatus_t, <sys/procfs.h> */
 170 static void
 171 gen_pstatus(void)
 172 {
 173         START(pstatus, pstatus_t);

 175         SCALAR_FIELD(pstatus_t,         pr_flags,       1);
 176         SCALAR_FIELD(pstatus_t,         pr_nlwp,        1);
 177         SCALAR_FIELD(pstatus_t,         pr_pid,         0);
 178         SCALAR_FIELD(pstatus_t,         pr_ppid,        0);
 179         SCALAR_FIELD(pstatus_t,         pr_pgid,        0);
 180         SCALAR_FIELD(pstatus_t,         pr_sid,         0);
 181         SCALAR_FIELD(pstatus_t,         pr_aslwpid,     1);
 182         SCALAR_FIELD(pstatus_t,         pr_agentid,     1);
 183         SCALAR_FIELD(pstatus_t,         pr_sigpend,     0);
 184         SCALAR_FIELD(pstatus_t,         pr_brkbase,     0);
 185         SCALAR_FIELD(pstatus_t,         pr_brksize,     0);
 186         SCALAR_FIELD(pstatus_t,         pr_stkbase,     0);
 187         SCALAR_FIELD(pstatus_t,         pr_stksize,     0);
 188         SCALAR_FIELD(pstatus_t,         pr_utime,       0);
 189         SCALAR_FIELD(pstatus_t,         pr_stime,       0);
 190         SCALAR_FIELD(pstatus_t,         pr_cutime,      0);
 191         SCALAR_FIELD(pstatus_t,         pr_cstime,      0);
 192         SCALAR_FIELD(pstatus_t,         pr_sigtrace,    0);
 193         SCALAR_FIELD(pstatus_t,         pr_flttrace,    0);
 194         SCALAR_FIELD(pstatus_t,         pr_sysentry,    0);
 195         SCALAR_FIELD(pstatus_t,         pr_sysexit,     0);
 196         SCALAR_FIELD(pstatus_t,         pr_dmodel,      0);
 197         SCALAR_FIELD(pstatus_t,         pr_taskid,      1);
 198         SCALAR_FIELD(pstatus_t,         pr_projid,      1);
 199         SCALAR_FIELD(pstatus_t,         pr_nzomb,       1);
 200         SCALAR_FIELD(pstatus_t,         pr_zoneid,      1);
 201         SCALAR_FIELD(pstatus_t,         pr_secflags,    0);
 202 #endif /* ! codereview */
 203         SCALAR_FIELD(pstatus_t,         pr_lwp,         0);

 205         END;
 206 }


 209 /* prstatus_t, <sys/old_procfs.h> */
 210 static void
 211 gen_prstatus(void)
 212 {
 213         START(prstatus, prstatus_t);

 215         SCALAR_FIELD(prstatus_t,        pr_flags,       1);
 216         SCALAR_FIELD(prstatus_t,        pr_why,         1);
 217         SCALAR_FIELD(prstatus_t,        pr_what,        1);
 218         SCALAR_FIELD(prstatus_t,        pr_info,        0);
 219         SCALAR_FIELD(prstatus_t,        pr_cursig,      1);
 220         SCALAR_FIELD(prstatus_t,        pr_nlwp,        0);
 221         SCALAR_FIELD(prstatus_t,        pr_sigpend,     0);
 222         SCALAR_FIELD(prstatus_t,        pr_sighold,     0);
 223         SCALAR_FIELD(prstatus_t,        pr_altstack,    0);
```



```
 224         SCALAR_FIELD(prstatus_t,        pr_action,      0);
 225         SCALAR_FIELD(prstatus_t,        pr_pid,         0);
 226         SCALAR_FIELD(prstatus_t,        pr_ppid,        0);
 227         SCALAR_FIELD(prstatus_t,        pr_pgrp,        0);
 228         SCALAR_FIELD(prstatus_t,        pr_sid,         0);
 229         SCALAR_FIELD(prstatus_t,        pr_utime,       0);
 230         SCALAR_FIELD(prstatus_t,        pr_stime,       0);
 231         SCALAR_FIELD(prstatus_t,        pr_cutime,      0);
 232         SCALAR_FIELD(prstatus_t,        pr_cstime,      0);
 233         ARRAY_FIELD(prstatus_t,         pr_clname,      0);
 234         SCALAR_FIELD(prstatus_t,        pr_syscall,     1);
 235         SCALAR_FIELD(prstatus_t,        pr_nsysarg,     1);
 236         ARRAY_FIELD(prstatus_t,         pr_sysarg,      1);
 237         SCALAR_FIELD(prstatus_t,        pr_who,         0);
 238         SCALAR_FIELD(prstatus_t,        pr_lwppend,     0);
 239         SCALAR_FIELD(prstatus_t,        pr_oldcontext,  0);
 240         SCALAR_FIELD(prstatus_t,        pr_brkbase,     0);
 241         SCALAR_FIELD(prstatus_t,        pr_brksize,     0);
 242         SCALAR_FIELD(prstatus_t,        pr_stkbase,     0);
 243         SCALAR_FIELD(prstatus_t,        pr_stksize,     0);
 244         SCALAR_FIELD(prstatus_t,        pr_processor,   1);
 245         SCALAR_FIELD(prstatus_t,        pr_bind,        1);
 246         SCALAR_FIELD(prstatus_t,        pr_instr,       1);
 247         SCALAR_FIELD(prstatus_t,        pr_reg,         0);

 249         END;
 250 }


 253 /* psinfo_t, <sys/procfs.h> */
 254 static void
 255 gen_psinfo(void)
 256 {
 257         START(psinfo, psinfo_t);

 259         SCALAR_FIELD(psinfo_t,          pr_flag,        1);
 260         SCALAR_FIELD(psinfo_t,          pr_nlwp,        1);
 261         SCALAR_FIELD(psinfo_t,          pr_pid,         0);
 262         SCALAR_FIELD(psinfo_t,          pr_ppid,        0);
 263         SCALAR_FIELD(psinfo_t,          pr_pgid,        0);
 264         SCALAR_FIELD(psinfo_t,          pr_sid,         0);
 265         SCALAR_FIELD(psinfo_t,          pr_uid,         0);
 266         SCALAR_FIELD(psinfo_t,          pr_euid,        0);
 267         SCALAR_FIELD(psinfo_t,          pr_gid,         0);
 268         SCALAR_FIELD(psinfo_t,          pr_egid,        0);
 269         SCALAR_FIELD(psinfo_t,          pr_addr,        0);
 270         SCALAR_FIELD(psinfo_t,          pr_size,        0);
 271         SCALAR_FIELD(psinfo_t,          pr_rssize,      0);
 272         SCALAR_FIELD(psinfo_t,          pr_ttydev,      0);
 273         SCALAR_FIELD(psinfo_t,          pr_pctcpu,      0);
 274         SCALAR_FIELD(psinfo_t,          pr_pctmem,      0);
 275         SCALAR_FIELD(psinfo_t,          pr_start,       0);
 276         SCALAR_FIELD(psinfo_t,          pr_time,        0);
 277         SCALAR_FIELD(psinfo_t,          pr_ctime,       0);
 278         ARRAY_FIELD(psinfo_t,           pr_fname,       0);
 279         ARRAY_FIELD(psinfo_t,           pr_psargs,      0);
 280         SCALAR_FIELD(psinfo_t,          pr_wstat,       1);
 281         SCALAR_FIELD(psinfo_t,          pr_argc,        1);
 282         SCALAR_FIELD(psinfo_t,          pr_argv,        0);
 283         SCALAR_FIELD(psinfo_t,          pr_envp,        0);
 284         SCALAR_FIELD(psinfo_t,          pr_dmodel,      0);
 285         SCALAR_FIELD(psinfo_t,          pr_taskid,      0);
 286         SCALAR_FIELD(psinfo_t,          pr_projid,      0);
 287         SCALAR_FIELD(psinfo_t,          pr_nzomb,       1);
 288         SCALAR_FIELD(psinfo_t,          pr_poolid,      0);
 289         SCALAR_FIELD(psinfo_t,          pr_zoneid,      0);
```

```
290         SCALAR_FIELD(psinfo_t,          pr_contract,    0);
291         SCALAR_FIELD(psinfo_t,          pr_lwp,         0);

293         END;
294 }

296 /* prpsinfo_t, <sys/old_procfs.h> */
297 static void
298 gen_prpsinfo(void)
299 {
300         START(prpsinfo, prpsinfo_t);

302         SCALAR_FIELD(prpsinfo_t,        pr_state,       0);
303         SCALAR_FIELD(prpsinfo_t,        pr_sname,       0);
304         SCALAR_FIELD(prpsinfo_t,        pr_zomb,        0);
305         SCALAR_FIELD(prpsinfo_t,        pr_nice,        0);
306         SCALAR_FIELD(prpsinfo_t,        pr_flag,        0);
307         SCALAR_FIELD(prpsinfo_t,        pr_uid,         0);
308         SCALAR_FIELD(prpsinfo_t,        pr_gid,         0);
309         SCALAR_FIELD(prpsinfo_t,        pr_pid,         0);
310         SCALAR_FIELD(prpsinfo_t,        pr_ppid,        0);
311         SCALAR_FIELD(prpsinfo_t,        pr_pgrp,        0);
312         SCALAR_FIELD(prpsinfo_t,        pr_sid,         0);
313         SCALAR_FIELD(prpsinfo_t,        pr_addr,        0);
314         SCALAR_FIELD(prpsinfo_t,        pr_size,        0);
315         SCALAR_FIELD(prpsinfo_t,        pr_rssize,      0);
316         SCALAR_FIELD(prpsinfo_t,        pr_wchan,       0);
317         SCALAR_FIELD(prpsinfo_t,        pr_start,       0);
318         SCALAR_FIELD(prpsinfo_t,        pr_time,        0);
319         SCALAR_FIELD(prpsinfo_t,        pr_pri,         1);
320         SCALAR_FIELD(prpsinfo_t,        pr_oldpri,      0);
321         SCALAR_FIELD(prpsinfo_t,        pr_cpu,         0);
322         SCALAR_FIELD(prpsinfo_t,        pr_ottydev,     0);
323         SCALAR_FIELD(prpsinfo_t,        pr_lttydev,     0);
324         ARRAY_FIELD(prpsinfo_t,         pr_clname,      0);
325         ARRAY_FIELD(prpsinfo_t,         pr_fname,       0);
326         ARRAY_FIELD(prpsinfo_t,         pr_psargs,      0);
327         SCALAR_FIELD(prpsinfo_t,        pr_syscall,     1);
328         SCALAR_FIELD(prpsinfo_t,        pr_ctime,       0);
329         SCALAR_FIELD(prpsinfo_t,        pr_bysize,      0);
330         SCALAR_FIELD(prpsinfo_t,        pr_byrssize,    0);
331         SCALAR_FIELD(prpsinfo_t,        pr_argc,        1);
332         SCALAR_FIELD(prpsinfo_t,        pr_argv,        0);
333         SCALAR_FIELD(prpsinfo_t,        pr_envp,        0);
334         SCALAR_FIELD(prpsinfo_t,        pr_wstat,       1);
335         SCALAR_FIELD(prpsinfo_t,        pr_pctcpu,      0);
336         SCALAR_FIELD(prpsinfo_t,        pr_pctmem,      0);
337         SCALAR_FIELD(prpsinfo_t,        pr_euid,        0);
338         SCALAR_FIELD(prpsinfo_t,        pr_egid,        0);
339         SCALAR_FIELD(prpsinfo_t,        pr_aslwpid,     0);
340         SCALAR_FIELD(prpsinfo_t,        pr_dmodel,      0);

342         END;
343 }

345 /* lwpsinfo_t, <sys/procfs.h> */
346 static void
347 gen_lwpsinfo(void)
348 {
349         START(lwpsinfo, lwpsinfo_t);

351         SCALAR_FIELD(lwpsinfo_t,        pr_flag,        1);
352         SCALAR_FIELD(lwpsinfo_t,        pr_lwpid,       0);
353         SCALAR_FIELD(lwpsinfo_t,        pr_addr,        0);
354         SCALAR_FIELD(lwpsinfo_t,        pr_wchan,       0);
355         SCALAR_FIELD(lwpsinfo_t,        pr_stype,       0);
```



```
356         SCALAR_FIELD(lwpsinfo_t,        pr_state,       0);
357         SCALAR_FIELD(lwpsinfo_t,        pr_sname,       0);
358         SCALAR_FIELD(lwpsinfo_t,        pr_nice,        0);
359         SCALAR_FIELD(lwpsinfo_t,        pr_syscall,     0);
360         SCALAR_FIELD(lwpsinfo_t,        pr_oldpri,      0);
361         SCALAR_FIELD(lwpsinfo_t,        pr_cpu,         0);
362         SCALAR_FIELD(lwpsinfo_t,        pr_pri,         1);
363         SCALAR_FIELD(lwpsinfo_t,        pr_pctcpu,      0);
364         SCALAR_FIELD(lwpsinfo_t,        pr_start,       0);
365         SCALAR_FIELD(lwpsinfo_t,        pr_time,        0);
366         ARRAY_FIELD(lwpsinfo_t,         pr_clname,      0);
367         ARRAY_FIELD(lwpsinfo_t,         pr_name,        0);
368         SCALAR_FIELD(lwpsinfo_t,        pr_onpro,       1);
369         SCALAR_FIELD(lwpsinfo_t,        pr_bindpro,     1);
370         SCALAR_FIELD(lwpsinfo_t,        pr_bindpset,    1);
371         SCALAR_FIELD(lwpsinfo_t,        pr_lgrp,        1);

373         END;
374 }

376 /* prcred_t, <sys/procfs.h> */
377 static void
378 gen_prcred(void)
379 {
380         START(prcred, prcred_t);

382         SCALAR_FIELD(prcred_t,          pr_euid,        0);
383         SCALAR_FIELD(prcred_t,          pr_ruid,        0);
384         SCALAR_FIELD(prcred_t,          pr_suid,        0);
385         SCALAR_FIELD(prcred_t,          pr_egid,        0);
386         SCALAR_FIELD(prcred_t,          pr_rgid,        0);
387         SCALAR_FIELD(prcred_t,          pr_sgid,        0);
388         SCALAR_FIELD(prcred_t,          pr_ngroups,     1);
389         ARRAY_FIELD(prcred_t,           pr_groups,      0);

391         END;
392 }

394 /* prpriv_t, <sys/procfs.h> */
395 static void
396 gen_prpriv(void)
397 {
398         START(prpriv, prpriv_t);

400         SCALAR_FIELD(prpriv_t,          pr_nsets,       0);
401         SCALAR_FIELD(prpriv_t,          pr_setsize,     0);
402         SCALAR_FIELD(prpriv_t,          pr_infosize,    0);
403         ARRAY_FIELD(prpriv_t,           pr_sets,        0);

405         END;
406 }


409 /* priv_impl_info_t, <sys/priv.h> */
410 static void
411 gen_priv_impl_info(void)
412 {
413         START(priv_impl_info, priv_impl_info_t);

415         SCALAR_FIELD(priv_impl_info_t,  priv_headersize,        0);
416         SCALAR_FIELD(priv_impl_info_t,  priv_flags,             0);
417         SCALAR_FIELD(priv_impl_info_t,  priv_nsets,             0);
418         SCALAR_FIELD(priv_impl_info_t,  priv_setsize,           0);
419         SCALAR_FIELD(priv_impl_info_t,  priv_max,               0);
420         SCALAR_FIELD(priv_impl_info_t,  priv_infosize,          0);
421         SCALAR_FIELD(priv_impl_info_t,  priv_globalinfosize,    0);
```

```
423         END;
424 }


427 /* fltset_t, <sys/fault.h> */
428 static void
429 gen_fltset(void)
430 {
431         START(fltset, fltset_t);

433         ARRAY_FIELD(fltset_t,   word,   0);

435         END;
436 }

438 /*
439  * Layout description of siginfo_t, <sys/siginfo.h>
440  *
441  * Note: many siginfo_t members are #defines mapping to
442  * long dotted members of sub-structs or unions, and
443  * we need the full member spec (with dots) for those.
444  */
445 static void
446 gen_siginfo(void)
447 {
448         START(siginfo, siginfo_t);

450         SCALAR_FIELD(siginfo_t,         si_signo,               0);
451         SCALAR_FIELD(siginfo_t,         si_errno,               0);
452         SCALAR_FIELD(siginfo_t,         si_code,                1);

454         SCALAR_FIELD4(siginfo_t,        si_value.sival_int,     0,
455             "__data.__proc.__pdata.__kill.__value.sival_int");

457         SCALAR_FIELD4(siginfo_t,        si_value.sival_ptr,     0,
458             "__data.__proc.__pdata.__kill.__value.sival_ptr");

460         SCALAR_FIELD4(siginfo_t,        si_pid,                 0,
461             "__data.__proc.__pid");

463         SCALAR_FIELD4(siginfo_t,        si_uid,                 0,
464             "__data.__proc.__pdata.__kill.__uid");

466         SCALAR_FIELD4(siginfo_t,        si_ctid,                0,
467             "__data.__proc.__ctid");

469         SCALAR_FIELD4(siginfo_t,        si_zoneid,              0,
470             "__data.__proc.__zoneid");

472         SCALAR_FIELD4(siginfo_t,        si_entity,              0,
473             "__data.__rctl.__entity");

475         SCALAR_FIELD4(siginfo_t,        si_addr,                0,
476             "__data.__fault.__addr");

478         SCALAR_FIELD4(siginfo_t,        si_status,              0,
479             "__data.__proc.__pdata.__cld.__status");

481         SCALAR_FIELD4(siginfo_t,        si_band,                0,
482             "__data.__file.__band");

484         END;
485 }

487 /* sigset_t, <sys/signal.h> */
```



```
488 static void
489 gen_sigset(void)
490 {
491         START(sigset, sigset_t);

493         ARRAY_FIELD(sigset_t,   __sigbits,      0);

495         END;
496 }


499 /* struct sigaction, <sys/signal.h> */
500 static void
501 gen_sigaction(void)
502 {
503         START(sigaction, struct sigaction);

505         SCALAR_FIELD(struct sigaction,  sa_flags,       0);

507         SCALAR_FIELD4(struct sigaction, sa_handler,     0,
508             "_funcptr._handler");

510         SCALAR_FIELD4(struct sigaction, sa_sigaction,   0,
511             "_funcptr._sigaction");

513         SCALAR_FIELD(struct sigaction,  sa_mask,        0);

515         END;
516 }

518 /* stack_t, <sys/signal.h> */
519 static void
520 gen_stack(void)
521 {
522         START(stack, stack_t);

524         SCALAR_FIELD(stack_t,   ss_sp,          0);
525         SCALAR_FIELD(stack_t,   ss_size,        0);
526         SCALAR_FIELD(stack_t,   ss_flags,       0);

528         END;
529 }

531 /* sysset_t, <sys/syscall.h> */
532 static void
533 gen_sysset(void)
534 {
535         START(sysset, sysset_t);

537         ARRAY_FIELD(sysset_t,   word,   0);

539         END;
540 }

542 /* timestruc_t, <sys/time_impl.h> */
543 static void
544 gen_timestruc(void)
545 {
546         START(timestruc, timestruc_t);

548         SCALAR_FIELD(timestruc_t,       tv_sec,         0);
549         SCALAR_FIELD(timestruc_t,       tv_nsec,        0);

551         END;
552 }
```

```
554 /* struct utsname, <sys/utsname.h> */
555 static void
556 gen_utsname(void)
557 {
558         START(utsname, struct utsname);

560         ARRAY_FIELD(struct utsname,     sysname,        0);
561         ARRAY_FIELD(struct utsname,     nodename,       0);
562         ARRAY_FIELD(struct utsname,     release,        0);
563         ARRAY_FIELD(struct utsname,     version,        0);
564         ARRAY_FIELD(struct utsname,     machine,        0);

566         END;
567 }

569 static void
570 gen_prfdinfo(void)
571 {
572         START(prfdinfo, prfdinfo_t);

574         SCALAR_FIELD(prfdinfo_t,        pr_fd,          0);
575         SCALAR_FIELD(prfdinfo_t,        pr_mode,        0);
576         SCALAR_FIELD(prfdinfo_t,        pr_uid,         0);
577         SCALAR_FIELD(prfdinfo_t,        pr_gid,         0);
578         SCALAR_FIELD(prfdinfo_t,        pr_major,       0);
579         SCALAR_FIELD(prfdinfo_t,        pr_minor,       0);
580         SCALAR_FIELD(prfdinfo_t,        pr_rmajor,      0);
581         SCALAR_FIELD(prfdinfo_t,        pr_rminor,      0);
582         SCALAR_FIELD(prfdinfo_t,        pr_ino,         0);
583         SCALAR_FIELD(prfdinfo_t,        pr_offset,      0);
584         SCALAR_FIELD(prfdinfo_t,        pr_size,        0);
585         SCALAR_FIELD(prfdinfo_t,        pr_fileflags,   0);
586         SCALAR_FIELD(prfdinfo_t,        pr_fdflags,     0);
587         ARRAY_FIELD(prfdinfo_t,         pr_path,        0);

589         END;
590 }

592 static void
593 gen_psecflags(void)
594 {
595         START(psecflags, psecflags_t);
596         SCALAR_FIELD(psecflags_t, psf_effective, 0);
597         SCALAR_FIELD(psecflags_t, psf_inherit, 0);
598         END;
599 }

601 #endif /* ! codereview */

603 /*ARGSUSED*/
604 int
605 main(int argc, char *argv[])
606 {
607         const char *fmt = "\t&%s_layout,\n";

609         /* get obj file for input */
610         if (argc < 3) {
611                 (void) fprintf(stderr,
612                     "usage: %s {object_file} {MACH}\n", argv[0]);
613                 exit(1);
614         }

616         objfile = argv[1];
617         machname = argv[2];

619         get_ctf_file(objfile);
```



```
621         (void) printf("#include <struct_layout.h>\n");

623         gen_auxv();
624         gen_prgregset();
625         gen_lwpstatus();
626         gen_pstatus();
627         gen_prstatus();
628         gen_psinfo();
629         gen_prpsinfo();
630         gen_lwpsinfo();
631         gen_prcred();
632         gen_prpriv();
633         gen_priv_impl_info();
634         gen_fltset();
635         gen_siginfo();
636         gen_sigset();
637         gen_sigaction();
638         gen_stack();
639         gen_sysset();
640         gen_timestruc();
641         gen_utsname();
642         gen_prfdinfo();
643         gen_psecflags();
644 #endif /* ! codereview */


647         /*
648          * Generate the full arch_layout description
649          */
650         (void) printf(
651             "\n\n\n\nstatic const sl_arch_layout_t layout_%s = {\n",
652             machname);
653         (void) printf(fmt, "auxv");
654         (void) printf(fmt, "fltset");
655         (void) printf(fmt, "lwpsinfo");
656         (void) printf(fmt, "lwpstatus");
657         (void) printf(fmt, "prcred");
658         (void) printf(fmt, "priv_impl_info");
659         (void) printf(fmt, "prpriv");
660         (void) printf(fmt, "psinfo");
661         (void) printf(fmt, "pstatus");
662         (void) printf(fmt, "prgregset");
663         (void) printf(fmt, "prpsinfo");
664         (void) printf(fmt, "prstatus");
665         (void) printf(fmt, "sigaction");
666         (void) printf(fmt, "siginfo");
667         (void) printf(fmt, "sigset");
668         (void) printf(fmt, "stack");
669         (void) printf(fmt, "sysset");
670         (void) printf(fmt, "timestruc");
671         (void) printf(fmt, "utsname");
672         (void) printf(fmt, "prfdinfo");
673         (void) printf(fmt, "psecflags");
674 #endif /* ! codereview */
675         (void) printf("};\n");

677         /*
678          * A public function, to make the information available
679          */
680         (void) printf("\n\nconst sl_arch_layout_t *\n");
681         (void) printf("struct_layout_%s(void)\n", machname);
682         (void) printf("{\n\treturn (&layout_%s);\n}\n", machname);

684         return (0);
685 }
```

```
687 /*
688  * Helper functions using the CTF library to get type info.
689  */

691 static void
692 get_ctf_file(char *fname)
693 {
694         int ctferr;

696         objfile = fname;
697         if ((ctf = ctf_open(objfile, &ctferr)) == NULL) {
698                 errx(1, "Couldn't open object file %s: %s\n", objfile,
699                     ctf_errmsg(ctferr));
700         }
701 }

703 static void
704 print_row(int boff, int eltlen, int nelts, int issigned, char *comment)
705 {
706         (void) printf("\t{ %d,\t%d,\t%d,\t%d },\t\t/* %s */\n",
707             boff, eltlen, nelts, issigned, comment);
708 }

710 static void
711 do_start(char *sname, char *tname)
712 {
713         do_start_name(sname);
714         do_start_sizeof(tname, NULL);
715 }

717 static void
718 do_start_name(char *sname)
719 {
720         (void) printf("\n\nstatic const sl_%s_layout_t %s_layout = {\n",
721             sname, sname);
722 }

724 static void
725 do_end(void)
726 {
727         (void) printf("};\n");
728 }

730 static void
731 do_start_sizeof(char *tname, char *rtname)
732 {
733         char comment[100];
734         ctf_id_t stype;
735         int sz;

737         if (rtname == NULL)
738                 rtname = tname;

740         if ((stype = ctf_lookup_by_name(ctf, rtname)) == CTF_ERR)
741                 errx(1, "Couldn't find type %s", rtname);
742         if ((stype = ctf_type_resolve(ctf, stype)) == CTF_ERR)
743                 errx(1, "Couldn't resolve type %s", tname);

745         if ((sz = (int)ctf_type_size(ctf, stype)) < 0) {
746                 errx(1, "Couldn't get size for type %s", tname);
747         } else if (sz == 0) {
748                 errx(1, "Invalid type size 0 for %s", tname);
749         }

751         (void) snprintf(comment, sizeof (comment), "sizeof (%s)", tname);
```



```
752         print_row(0, sz, 0, 0, comment);
753 }

755 static void
756 do_scalar_field(char *tname, char *fname, int _sign, char *dotfield)
757 {
758         int rc, off, sz, ftype;

760         rc = get_field_info(tname, fname, dotfield, &off, &ftype);
761         if (rc < 0)
762                 errx(1, "Can't get field info for %s->%s", tname, fname);

764         if ((ftype = ctf_type_resolve(ctf, ftype)) == CTF_ERR)
765                 errx(1, "Couldn't resolve type of %s->%s", tname, fname);

767         if ((sz = (int)ctf_type_size(ctf, ftype)) < 0) {
768                 errx(1, "Couldn't get size for type ID %d", ftype);
769         } else if (sz == 0) {
770                 errx(1, "Invalid type size 0 for type ID %d", ftype);
771         }

773         print_row(off, sz, 0, _sign, fname);
774 }

776 static void
777 do_array_field(char *tname, char *fname,
778         int _sign, char *dotfield)
779 {
780         char comment[100];
781         ctf_arinfo_t ai;
782         int typekind;
783         int esz, rc, off, ftype;

785         rc = get_field_info(tname, fname, dotfield, &off, &ftype);
786         if (rc < 0)
787                 errx(1, "Can't get field info for %s->%s", tname, fname);

789         if ((ftype = ctf_type_resolve(ctf, ftype)) == CTF_ERR)
790                 errx(1, "Couldn't resolve type of %s->%s", tname, fname);

792         typekind = ctf_type_kind(ctf, ftype);
793         if (typekind != CTF_K_ARRAY)
794                 errx(1, "Wrong type for %s->%s", tname, fname);

796         rc = ctf_array_info(ctf, ftype, &ai);
797         if (rc != 0)
798                 errx(1, "Can't get array info for %s->%s\n", tname, fname);
799         esz = ctf_type_size(ctf, ai.ctr_contents);
800         if (esz < 0)
801                 errx(1, "Can't get element size for %s->%s\n", tname, fname);

803         (void) snprintf(comment, sizeof (comment), "%s[]", fname);
804         print_row(off, esz, ai.ctr_nelems, _sign, comment);
805 }

807 static void
808 do_array_type(char *tname, char *fname, int _sign)
809 {
810         ctf_arinfo_t ai;
811         int stype, typekind;
812         int esz, rc;

814         if ((stype = ctf_lookup_by_name(ctf, tname)) == CTF_ERR)
815                 errx(1, "Couldn't find type %s", tname);
816         if ((stype = ctf_type_resolve(ctf, stype)) == CTF_ERR)
817                 errx(1, "Couldn't resolve type %s", tname);
```

```
819         typekind = ctf_type_kind(ctf, stype);
820         if (typekind != CTF_K_ARRAY)
821                 errx(1, "Wrong type for %s->%s", tname, fname);

823         rc = ctf_array_info(ctf, stype, &ai);
824         if (rc != 0)
825                 errx(1, "Can't get array info for %s->%s\n", tname, fname);
826         esz = ctf_type_size(ctf, ai.ctr_contents);
827         if (esz < 0)
828                 errx(1, "Can't get element size for %s->%s\n", tname, fname);

830         print_row(0, esz, ai.ctr_nelems, _sign, fname);
831 }


834 struct gfinfo {
835         char *tname;    /* top type name, i.e. the struct */
836         char *fname;    /* field name */
837         char *dotname;  /* full field name with dots (optional) */
838         char *prefix;   /* current field search prefix */
839         int base_off;
840         int fld_off;
841         int fld_type;
842 };

844 static int gfi_iter(const char *fname, ctf_id_t mbrtid,
845         ulong_t off, void *varg);

847 /*
848  * Lookup field "fname" in type "tname".  If "dotname" is non-NULL,
849  * that's the full field name with dots, i.e. a_un.un_foo, which
850  * we must search for by walking the struct CTF recursively.
851  */
852 static int
853 get_field_info(char *tname, char *fname, char *dotname,
854         int *offp, int *tidp)
855 {
856         struct gfinfo gfi;
857         ctf_id_t stype;
858         int typekind;
859         int rc;

861         if ((stype = ctf_lookup_by_name(ctf, tname)) == CTF_ERR)
862                 errx(1, "Couldn't find type %s", tname);
863         if ((stype = ctf_type_resolve(ctf, stype)) == CTF_ERR)
864                 errx(1, "Couldn't resolve type %s", tname);

866         /* If fname has a dot, use it as dotname too. */
867         if (dotname == NULL && strchr(fname, '.') != NULL)
868                 dotname = fname;

870         gfi.tname = tname;
871         gfi.fname = fname;
872         gfi.dotname = dotname;
873         gfi.prefix = "";
874         gfi.base_off = 0;
875         gfi.fld_off = 0;
876         gfi.fld_type = 0;

878         typekind = ctf_type_kind(ctf, stype);
879         switch (typekind) {

881         case CTF_K_STRUCT:
882         case CTF_K_UNION:
883                 rc = ctf_member_iter(ctf, stype, gfi_iter, &gfi);
```



```
884                 break;

886         default:
887                 errx(1, "Unexpected top-level type for %s", tname);
888                 break;
889         }

891         if (rc < 0)
892                 errx(1, "Error getting info for %s.%s", stype, fname);
893         if (rc == 0)
894                 errx(1, "Did not find %s.%s", tname, fname);

896         *offp = gfi.fld_off;
897         *tidp = gfi.fld_type;

899         return (0);
900 }

902 /*
903  * Iteration callback for ctf_member_iter
904  * Return <0 on error, 0 to keep looking, >0 for found.
905  *
906  * If no dotname, simple search for fieldname.
907  * If we're asked to search with dotname, we need to do a full
908  * recursive walk of the types under the dotname.
909  */
910 int
911 gfi_iter(const char *fieldname, ctf_id_t mbrtid, ulong_t off, void *varg)
912 {
913         char namebuf[100];
914         struct gfinfo *gfi = varg;
915         char *saveprefix;
916         int saveoff;
917         int typekind;
918         int byteoff;
919         int len, rc;

921         byteoff = gfi->base_off + (int)(off >> 3);

923         /* Easy cases first: no dotname */
924         if (gfi->dotname == NULL) {
925                 if (strcmp(gfi->fname, fieldname) == 0) {
926                         gfi->fld_off = byteoff;
927                         gfi->fld_type = mbrtid;
928                         return (1);
929                 }
930                 return (0);
931         }

933         /* Exact match on the dotname? */
934         (void) snprintf(namebuf, sizeof (namebuf), "%s%s",
935             gfi->prefix, fieldname);
936         if (strcmp(gfi->dotname, namebuf) == 0) {
937                 gfi->fld_off = byteoff;
938                 gfi->fld_type = mbrtid;
939                 return (1);
940         }

942         /*
943          * May need to recurse under this field, but
944          * only if there's a match through '.'
945          */
946         (void) strlcat(namebuf, ".", sizeof (namebuf));
947         len = strlen(namebuf);
948         if (strncmp(gfi->dotname, namebuf, len) != 0)
949                 return (0);
```

```
951         typekind = ctf_type_kind(ctf, mbrtid);
952         switch (typekind) {
953         case CTF_K_STRUCT:
954         case CTF_K_UNION:
955                 break;
956         default:
957                 return (0);
958         }

960         /* Recursively walk members */
961         saveprefix = gfi->prefix;
962         saveoff = gfi->base_off;
963         gfi->prefix = namebuf;
964         gfi->base_off = byteoff;
965         rc = ctf_member_iter(ctf, mbrtid, gfi_iter, gfi);
966         gfi->prefix = saveprefix;
967         gfi->base_off = saveoff;

969         return (rc);
970 }
```

```
**********************************************************
   15604 Wed May 27 19:48:59 2015
new/usr/src/cmd/sgs/elfdump/common/struct_layout.h
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
______unchanged_portion_omitted_

 180 /*
 181  * Layout description of pstatus_t, from <sys/procfs.h>.
 182  */
 183 typedef struct {
 184         sl_field_t              sizeof_struct;
 185         sl_field_t              pr_flags;
 186         sl_field_t              pr_nlwp;
 187         sl_field_t              pr_pid;
 188         sl_field_t              pr_ppid;
 189         sl_field_t              pr_pgid;
 190         sl_field_t              pr_sid;
 191         sl_field_t              pr_aslwpid;
 192         sl_field_t              pr_agentid;
 193         sl_field_t              pr_sigpend;
 194         sl_field_t              pr_brkbase;
 195         sl_field_t              pr_brksize;
 196         sl_field_t              pr_stkbase;
 197         sl_field_t              pr_stksize;
 198         sl_field_t              pr_utime;
 199         sl_field_t              pr_stime;
 200         sl_field_t              pr_cutime;
 201         sl_field_t              pr_cstime;
 202         sl_field_t              pr_sigtrace;
 203         sl_field_t              pr_flttrace;
 204         sl_field_t              pr_sysentry;
 205         sl_field_t              pr_sysexit;
 206         sl_field_t              pr_dmodel;
 207         sl_field_t              pr_taskid;
 208         sl_field_t              pr_projid;
 209         sl_field_t              pr_nzomb;
 210         sl_field_t              pr_zoneid;
 211         sl_field_t              pr_secflags;
 212 #endif /* ! codereview */
 213         sl_field_t              pr_lwp;
 214 } sl_pstatus_layout_t;

 216 /*
 217  * Layout description of prstatus_t, from <sys/old_procfs.h>.
 218  */
 219 typedef struct {
 220         sl_field_t              sizeof_struct;
 221         sl_field_t              pr_flags;
 222         sl_field_t              pr_why;
 223         sl_field_t              pr_what;
 224         sl_field_t              pr_info;
 225         sl_field_t              pr_cursig;
 226         sl_field_t              pr_nlwp;
 227         sl_field_t              pr_sigpend;
 228         sl_field_t              pr_sighold;
 229         sl_field_t              pr_altstack;
 230         sl_field_t              pr_action;
 231         sl_field_t              pr_pid;
 232         sl_field_t              pr_ppid;
 233         sl_field_t              pr_pgrp;
 234         sl_field_t              pr_sid;
 235         sl_field_t              pr_utime;
```



```
 236         sl_field_t              pr_stime;
 237         sl_field_t              pr_cutime;
 238         sl_field_t              pr_cstime;
 239         sl_field_t              pr_clname;
 240         sl_field_t              pr_syscall;
 241         sl_field_t              pr_nsysarg;
 242         sl_field_t              pr_sysarg;
 243         sl_field_t              pr_who;
 244         sl_field_t              pr_lwppend;
 245         sl_field_t              pr_oldcontext;
 246         sl_field_t              pr_brkbase;
 247         sl_field_t              pr_brksize;
 248         sl_field_t              pr_stkbase;
 249         sl_field_t              pr_stksize;
 250         sl_field_t              pr_processor;
 251         sl_field_t              pr_bind;
 252         sl_field_t              pr_instr;
 253         sl_field_t              pr_reg;
 254 } sl_prstatus_layout_t;

 256 /*
 257  * Layout description of psinfo_t, from <sys/procfs.h>.
 258  */
 259 typedef struct {
 260         sl_field_t              sizeof_struct;
 261         sl_field_t              pr_flag;
 262         sl_field_t              pr_nlwp;
 263         sl_field_t              pr_pid;
 264         sl_field_t              pr_ppid;
 265         sl_field_t              pr_pgid;
 266         sl_field_t              pr_sid;
 267         sl_field_t              pr_uid;
 268         sl_field_t              pr_euid;
 269         sl_field_t              pr_gid;
 270         sl_field_t              pr_egid;
 271         sl_field_t              pr_addr;
 272         sl_field_t              pr_size;
 273         sl_field_t              pr_rssize;
 274         sl_field_t              pr_ttydev;
 275         sl_field_t              pr_pctcpu;
 276         sl_field_t              pr_pctmem;
 277         sl_field_t              pr_start;
 278         sl_field_t              pr_time;
 279         sl_field_t              pr_ctime;
 280         sl_field_t              pr_fname;
 281         sl_field_t              pr_psargs;
 282         sl_field_t              pr_wstat;
 283         sl_field_t              pr_argc;
 284         sl_field_t              pr_argv;
 285         sl_field_t              pr_envp;
 286         sl_field_t              pr_dmodel;
 287         sl_field_t              pr_taskid;
 288         sl_field_t              pr_projid;
 289         sl_field_t              pr_nzomb;
 290         sl_field_t              pr_poolid;
 291         sl_field_t              pr_zoneid;
 292         sl_field_t              pr_contract;
 293         sl_field_t              pr_lwp;
 294 } sl_psinfo_layout_t;

 296 /*
 297  * Layout description of prpsinfo_t, from <sys/old_procfs.h>.
 298  */
 299 typedef struct {
 300         sl_field_t              sizeof_struct;
 301         sl_field_t              pr_state;
```

```
302         sl_field_t              pr_sname;
303         sl_field_t              pr_zomb;
304         sl_field_t              pr_nice;
305         sl_field_t              pr_flag;
306         sl_field_t              pr_uid;
307         sl_field_t              pr_gid;
308         sl_field_t              pr_pid;
309         sl_field_t              pr_ppid;
310         sl_field_t              pr_pgrp;
311         sl_field_t              pr_sid;
312         sl_field_t              pr_addr;
313         sl_field_t              pr_size;
314         sl_field_t              pr_rssize;
315         sl_field_t              pr_wchan;
316         sl_field_t              pr_start;
317         sl_field_t              pr_time;
318         sl_field_t              pr_pri;
319         sl_field_t              pr_oldpri;
320         sl_field_t              pr_cpu;
321         sl_field_t              pr_ottydev;
322         sl_field_t              pr_lttydev;
323         sl_field_t              pr_clname;
324         sl_field_t              pr_fname;
325         sl_field_t              pr_psargs;
326         sl_field_t              pr_syscall;
327         sl_field_t              pr_ctime;
328         sl_field_t              pr_bysize;
329         sl_field_t              pr_byrssize;
330         sl_field_t              pr_argc;
331         sl_field_t              pr_argv;
332         sl_field_t              pr_envp;
333         sl_field_t              pr_wstat;
334         sl_field_t              pr_pctcpu;
335         sl_field_t              pr_pctmem;
336         sl_field_t              pr_euid;
337         sl_field_t              pr_egid;
338         sl_field_t              pr_aslwpid;
339         sl_field_t              pr_dmodel;
340 } sl_prpsinfo_layout_t;

342 /*
343  * Layout description of lwpsinfo_t, from <sys/procfs.h>.
344  */
345 typedef struct {
346         sl_field_t              sizeof_struct;
347         sl_field_t              pr_flag;
348         sl_field_t              pr_lwpid;
349         sl_field_t              pr_addr;
350         sl_field_t              pr_wchan;
351         sl_field_t              pr_stype;
352         sl_field_t              pr_state;
353         sl_field_t              pr_sname;
354         sl_field_t              pr_nice;
355         sl_field_t              pr_syscall;
356         sl_field_t              pr_oldpri;
357         sl_field_t              pr_cpu;
358         sl_field_t              pr_pri;
359         sl_field_t              pr_pctcpu;
360         sl_field_t              pr_start;
361         sl_field_t              pr_time;
362         sl_field_t              pr_clname;
363         sl_field_t              pr_name;
364         sl_field_t              pr_onpro;
365         sl_field_t              pr_bindpro;
366         sl_field_t              pr_bindpset;
367         sl_field_t              pr_lgrp;
```



```
368 } sl_lwpsinfo_layout_t;

370 /*
371  * Layout description of prcred_t, from <sys/procfs.h>.
372  */
373 typedef struct {
374         sl_field_t              sizeof_struct;
375         sl_field_t              pr_euid;
376         sl_field_t              pr_ruid;
377         sl_field_t              pr_suid;
378         sl_field_t              pr_egid;
379         sl_field_t              pr_rgid;
380         sl_field_t              pr_sgid;
381         sl_field_t              pr_ngroups;
382         sl_field_t              pr_groups;
383 } sl_prcred_layout_t;

385 /*
386  * Layout description of prpriv_t, from <sys/procfs.h>.
387  */
388 typedef struct {
389         sl_field_t              sizeof_struct;
390         sl_field_t              pr_nsets;
391         sl_field_t              pr_setsize;
392         sl_field_t              pr_infosize;
393         sl_field_t              pr_sets;
394 } sl_prpriv_layout_t;

396 /*
397  * Layout description of priv_impl_info_t, from <sys/priv.h>.
398  */
399 typedef struct {
400         sl_field_t              sizeof_struct;
401         sl_field_t              priv_headersize;
402         sl_field_t              priv_flags;
403         sl_field_t              priv_nsets;
404         sl_field_t              priv_setsize;
405         sl_field_t              priv_max;
406         sl_field_t              priv_infosize;
407         sl_field_t              priv_globalinfosize;
408 } sl_priv_impl_info_layout_t;

410 /*
411  * Layout description of fltset_t, from <sys/fault.h>.
412  */
413 typedef struct {
414         sl_field_t              sizeof_struct;
415         sl_field_t              word;
416 } sl_fltset_layout_t;

418 /*
419  * Layout description of siginfo_t, from <sys/siginfo.h>.
420  *
421  * siginfo_t is unusual, in that it contains a large union
422  * full of private fields. There are macros defined to give
423  * access to these fields via the names documented in the
424  * siginfo manpage. We stick to the documented names
425  * rather than try to unravel the undocumented blob. Hence,
426  * the layout description below is a "logical" view of siginfo_t.
427  * The fields below are not necessarily in the same order as
428  * they appear in siginfo_t, nor are they everything that is in
429  * that struct. They may also overlap each other, if they are
430  * contained within of the union.
431  *
432  * The f_ prefixes are used to prevent our field names from
433  * clashing with the macros defined in siginfo.h.
```

```
 */
typedef struct {
        sl_field_t              sizeof_struct;
        sl_field_t              f_si_signo;
        sl_field_t              f_si_errno;
        sl_field_t              f_si_code;
        sl_field_t              f_si_value_int;
        sl_field_t              f_si_value_ptr;
        sl_field_t              f_si_pid;
        sl_field_t              f_si_uid;
        sl_field_t              f_si_ctid;
        sl_field_t              f_si_zoneid;
        sl_field_t              f_si_entity;
        sl_field_t              f_si_addr;
        sl_field_t              f_si_status;
        sl_field_t              f_si_band;
} sl_siginfo_layout_t;

/*
 * Layout description of sigset_t, from <sys/signal.h>.
 */
typedef struct {
        sl_field_t              sizeof_struct;
        sl_field_t              sigbits;
} sl_sigset_layout_t;

/*
 * Layout description of struct sigaction, from <sys/signal.h>.
 */
typedef struct {
        sl_field_t              sizeof_struct;
        sl_field_t              sa_flags;
        sl_field_t              sa_hand;
        sl_field_t              sa_sigact;
        sl_field_t              sa_mask;
} sl_sigaction_layout_t;

/*
 * Layout description of stack_t, from <sys/signal.h>.
 */
typedef struct {
        sl_field_t              sizeof_struct;
        sl_field_t              ss_sp;
        sl_field_t              ss_size;
        sl_field_t              ss_flags;
} sl_stack_layout_t;

/*
 * Layout description of sysset_t, from <sys/syscall.h>.
 */
typedef struct {
        sl_field_t              sizeof_struct;
        sl_field_t              word;
} sl_sysset_layout_t;

/*
 * Layout description of timestruc_t, from <sys/time_impl.h>.
 */
typedef struct {
        sl_field_t              sizeof_struct;
        sl_field_t              tv_sec;
        sl_field_t              tv_nsec;
} sl_timestruc_layout_t;

/*
 * Layout description of struct utsname, from <sys/utsname.h>.
```



```
 */
typedef struct {
        sl_field_t              sizeof_struct;
        sl_field_t              sysname;
        sl_field_t              nodename;
        sl_field_t              release;
        sl_field_t              version;
        sl_field_t              machine;
} sl_utsname_layout_t;

/*
 * Layout description of prdinfo_t, from <sys/procfs.h>.
 */
typedef struct {
        sl_field_t              sizeof_struct;
        sl_field_t              pr_fd;
        sl_field_t              pr_mode;
        sl_field_t              pr_uid;
        sl_field_t              pr_gid;
        sl_field_t              pr_major;
        sl_field_t              pr_minor;
        sl_field_t              pr_rmajor;
        sl_field_t              pr_rminor;
        sl_field_t              pr_ino;
        sl_field_t              pr_offset;
        sl_field_t              pr_size;
        sl_field_t              pr_fileflags;
        sl_field_t              pr_fdflags;
        sl_field_t              pr_path;
} sl_prfdinfo_layout_t;

typedef struct {
        sl_field_t              sizeof_struct;
        sl_field_t              psf_effective;
        sl_field_t              psf_inherit;
} sl_psecflags_layout_t;

#endif /* ! codereview */
/*
 * This type collects all of the layout definitions for
 * a given architecture.
 */
typedef struct {
        const sl_auxv_layout_t          *auxv;          /* auxv_t */
        const sl_fltset_layout_t        *fltset;        /* fltset_t */
        const sl_lwpsinfo_layout_t      *lwpsinfo;      /* lwpsinfo_t */
        const sl_lwpstatus_layout_t     *lwpstatus;     /* lwpstatus_t */
        const sl_prcred_layout_t        *prcred;        /* prcred_t */
        const sl_priv_impl_info_layout_t *priv_impl_info; /* priv_impl_info_t */
        const sl_prpriv_layout_t        *prpriv;        /* prpriv_t */
        const sl_psinfo_layout_t        *psinfo;        /* psinfo_t */
        const sl_pstatus_layout_t       *pstatus;       /* pstatus_t */
        const sl_prgregset_layout_t     *prgregset;     /* prgregset_t */
        const sl_prpsinfo_layout_t      *prpsinfo;      /* prpsinfo_t */
        const sl_prstatus_layout_t      *prstatus;      /* prstatus_t */
        const sl_sigaction_layout_t     *sigaction;     /* struct sigaction */
        const sl_siginfo_layout_t       *siginfo;       /* siginfo_t */
        const sl_sigset_layout_t        *sigset;        /* sigset_t */
        const sl_stack_layout_t         *stack;         /* stack_t */
        const sl_sysset_layout_t        *sysset;        /* sysset_t */
        const sl_timestruc_layout_t     *timestruc;     /* timestruc_t */
        const sl_utsname_layout_t       *utsname;       /* struct utsname */
        const sl_prfdinfo_layout_t      *prfdinfo;      /* prdinfo_t */
        const sl_psecflags_layout_t     *psecflags;     /* psecflags_t */
#endif /* ! codereview */
} sl_arch_layout_t;
```

```
569 extern  void            sl_extract_num_field(const char *data, int do_swap,
570                             const sl_field_t *fdesc, sl_data_t *field_data);
571 extern  Word            sl_extract_as_word(const char *data, int do_swap,
572                             const sl_field_t *fdesc);
573 extern  Lword           sl_extract_as_lword(const char *data, int do_swap,
574                             const sl_field_t *fdesc);
575 extern  Sword           sl_extract_as_sword(const char *data, int do_swap,
576                             const sl_field_t *fdesc);
577 extern  const char      *sl_fmt_num(const char *data, int do_swap,
578                             const sl_field_t *fdesc, sl_fmt_num_t fmt_type,
579                             sl_fmtbuf_t buf);


582 extern  const sl_arch_layout_t  *sl_mach(Half);
583 extern  const sl_arch_layout_t  *struct_layout_i386(void);
584 extern  const sl_arch_layout_t  *struct_layout_amd64(void);
585 extern  const sl_arch_layout_t  *struct_layout_sparc(void);
586 extern  const sl_arch_layout_t  *struct_layout_sparcv9(void);



590 #ifdef  __cplusplus
591 }
592 #endif

594 #endif  /* _STRUCT_LAYOUT_H */
```

```
**********************************************************
   12478 Wed May 27 19:49:00 2015
new/usr/src/cmd/sgs/elfdump/common/struct_layout_amd64.c
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
______unchanged_portion_omitted_


  79 static const sl_pstatus_layout_t pstatus_layout = {
  80         { 0,    1680,   0,      0 },            /* sizeof (pstatus_t) */
  81         { 0,    4,      0,      1 },            /* pr_flags */
  82         { 4,    4,      0,      1 },            /* pr_nlwp */
  83         { 8,    4,      0,      0 },            /* pr_pid */
  84         { 12,   4,      0,      0 },            /* pr_ppid */
  85         { 16,   4,      0,      0 },            /* pr_pgid */
  86         { 20,   4,      0,      0 },            /* pr_sid */
  87         { 24,   4,      0,      1 },            /* pr_aslwpid */
  88         { 28,   4,      0,      1 },            /* pr_agentid */
  89         { 32,   16,     0,      0 },            /* pr_sigpend */
  90         { 48,   8,      0,      0 },            /* pr_brkbase */
  91         { 56,   8,      0,      0 },            /* pr_brksize */
  92         { 64,   8,      0,      0 },            /* pr_stkbase */
  93         { 72,   8,      0,      0 },            /* pr_stksize */
  94         { 80,   16,     0,      0 },            /* pr_utime */
  95         { 96,   16,     0,      0 },            /* pr_stime */
  96         { 112,  16,     0,      0 },            /* pr_cutime */
  97         { 128,  16,     0,      0 },            /* pr_cstime */
  98         { 144,  16,     0,      0 },            /* pr_sigtrace */
  99         { 160,  16,     0,      0 },            /* pr_flttrace */
 100         { 176,  64,     0,      0 },            /* pr_sysentry */
 101         { 240,  64,     0,      0 },            /* pr_sysexit */
 102         { 304,  1,      0,      0 },            /* pr_dmodel */
 103         { 308,  4,      0,      1 },            /* pr_taskid */
 104         { 312,  4,      0,      1 },            /* pr_projid */
 105         { 316,  4,      0,      1 },            /* pr_nzomb */
 106         { 320,  4,      0,      1 },            /* pr_zoneid */
 107         { 324,  8,      0,      0 },            /* pr_secflags */
 108 #endif /* ! codereview */
 109         { 384,  1296,   0,      0 },            /* pr_lwp */
 110 };


 113 static const sl_prstatus_layout_t prstatus_layout = {
 114         { 0,    824,    0,      0 },            /* sizeof (prstatus_t) */
 115         { 0,    4,      0,      1 },            /* pr_flags */
 116         { 4,    2,      0,      1 },            /* pr_why */
 117         { 6,    2,      0,      1 },            /* pr_what */
 118         { 8,    256,    0,      0 },            /* pr_info */
 119         { 264,  2,      0,      1 },            /* pr_cursig */
 120         { 266,  2,      0,      0 },            /* pr_nlwp */
 121         { 268,  16,     0,      0 },            /* pr_sigpend */
 122         { 284,  16,     0,      0 },            /* pr_sighold */
 123         { 304,  24,     0,      0 },            /* pr_altstack */
 124         { 328,  32,     0,      0 },            /* pr_action */
 125         { 360,  4,      0,      0 },            /* pr_pid */
 126         { 364,  4,      0,      0 },            /* pr_ppid */
 127         { 368,  4,      0,      0 },            /* pr_pgrp */
 128         { 372,  4,      0,      0 },            /* pr_sid */
 129         { 376,  16,     0,      0 },            /* pr_utime */
 130         { 392,  16,     0,      0 },            /* pr_stime */
 131         { 408,  16,     0,      0 },            /* pr_cutime */
 132         { 424,  16,     0,      0 },            /* pr_cstime */
 133         { 440,  1,      8,      0 },            /* pr_clname[] */
```



```
 134         { 448,  2,      0,      1 },            /* pr_syscall */
 135         { 450,  2,      0,      1 },            /* pr_nsysarg */
 136         { 456,  8,      8,      1 },            /* pr_sysarg[] */
 137         { 520,  4,      0,      0 },            /* pr_who */
 138         { 524,  16,     0,      0 },            /* pr_lwppend */
 139         { 544,  8,      0,      0 },            /* pr_oldcontext */
 140         { 552,  8,      0,      0 },            /* pr_brkbase */
 141         { 560,  8,      0,      0 },            /* pr_brksize */
 142         { 568,  8,      0,      0 },            /* pr_stkbase */
 143         { 576,  8,      0,      0 },            /* pr_stksize */
 144         { 584,  2,      0,      1 },            /* pr_processor */
 145         { 586,  2,      0,      1 },            /* pr_bind */
 146         { 592,  8,      0,      1 },            /* pr_instr */
 147         { 600,  224,    0,      0 },            /* pr_reg */
 148 };


 151 static const sl_psinfo_layout_t psinfo_layout = {
 152         { 0,    416,    0,      0 },            /* sizeof (psinfo_t) */
 153         { 0,    4,      0,      1 },            /* pr_flag */
 154         { 4,    4,      0,      1 },            /* pr_nlwp */
 155         { 8,    4,      0,      0 },            /* pr_pid */
 156         { 12,   4,      0,      0 },            /* pr_ppid */
 157         { 16,   4,      0,      0 },            /* pr_pgid */
 158         { 20,   4,      0,      0 },            /* pr_sid */
 159         { 24,   4,      0,      0 },            /* pr_uid */
 160         { 28,   4,      0,      0 },            /* pr_euid */
 161         { 32,   4,      0,      0 },            /* pr_gid */
 162         { 36,   4,      0,      0 },            /* pr_egid */
 163         { 40,   8,      0,      0 },            /* pr_addr */
 164         { 48,   8,      0,      0 },            /* pr_size */
 165         { 56,   8,      0,      0 },            /* pr_rssize */
 166         { 72,   8,      0,      0 },            /* pr_ttydev */
 167         { 80,   2,      0,      0 },            /* pr_pctcpu */
 168         { 82,   2,      0,      0 },            /* pr_pctmem */
 169         { 88,   16,     0,      0 },            /* pr_start */
 170         { 104,  16,     0,      0 },            /* pr_time */
 171         { 120,  16,     0,      0 },            /* pr_ctime */
 172         { 136,  1,      16,     0 },            /* pr_fname[] */
 173         { 152,  1,      80,     0 },            /* pr_psargs[] */
 174         { 232,  4,      0,      1 },            /* pr_wstat */
 175         { 236,  4,      0,      1 },            /* pr_argc */
 176         { 240,  8,      0,      0 },            /* pr_argv */
 177         { 248,  8,      0,      0 },            /* pr_envp */
 178         { 256,  1,      0,      0 },            /* pr_dmodel */
 179         { 260,  4,      0,      0 },            /* pr_taskid */
 180         { 264,  4,      0,      0 },            /* pr_projid */
 181         { 268,  4,      0,      1 },            /* pr_nzomb */
 182         { 272,  4,      0,      0 },            /* pr_poolid */
 183         { 276,  4,      0,      0 },            /* pr_zoneid */
 184         { 280,  4,      0,      0 },            /* pr_contract */
 185         { 288,  128,    0,      0 },            /* pr_lwp */
 186 };


 189 static const sl_prpsinfo_layout_t prpsinfo_layout = {
 190         { 0,    328,    0,      0 },            /* sizeof (prpsinfo_t) */
 191         { 0,    1,      0,      0 },            /* pr_state */
 192         { 1,    1,      0,      0 },            /* pr_sname */
 193         { 2,    1,      0,      0 },            /* pr_zomb */
 194         { 3,    1,      0,      0 },            /* pr_nice */
 195         { 4,    4,      0,      0 },            /* pr_flag */
 196         { 8,    4,      0,      0 },            /* pr_uid */
 197         { 12,   4,      0,      0 },            /* pr_gid */
 198         { 16,   4,      0,      0 },            /* pr_pid */
 199         { 20,   4,      0,      0 },            /* pr_ppid */
```

```
200         { 24,   4,      0,      0 },            /* pr_pgrp */
201         { 28,   4,      0,      0 },            /* pr_sid */
202         { 32,   8,      0,      0 },            /* pr_addr */
203         { 40,   8,      0,      0 },            /* pr_size */
204         { 48,   8,      0,      0 },            /* pr_rssize */
205         { 56,   8,      0,      0 },            /* pr_wchan */
206         { 64,   16,     0,      0 },            /* pr_start */
207         { 80,   16,     0,      0 },            /* pr_time */
208         { 96,   4,      0,      1 },            /* pr_pri */
209         { 100,  1,      0,      0 },            /* pr_oldpri */
210         { 101,  1,      0,      0 },            /* pr_cpu */
211         { 102,  2,      0,      0 },            /* pr_ottydev */
212         { 104,  8,      0,      0 },            /* pr_lttydev */
213         { 112,  1,      8,      0 },            /* pr_clname[] */
214         { 120,  1,      16,     0 },            /* pr_fname[] */
215         { 136,  1,      80,     0 },            /* pr_psargs[] */
216         { 216,  2,      0,      1 },            /* pr_syscall */
217         { 224,  16,     0,      0 },            /* pr_ctime */
218         { 240,  8,      0,      0 },            /* pr_bysize */
219         { 248,  8,      0,      0 },            /* pr_byrssize */
220         { 256,  4,      0,      1 },            /* pr_argc */
221         { 264,  8,      0,      0 },            /* pr_argv */
222         { 272,  8,      0,      0 },            /* pr_envp */
223         { 280,  4,      0,      1 },            /* pr_wstat */
224         { 284,  2,      0,      0 },            /* pr_pctcpu */
225         { 286,  2,      0,      0 },            /* pr_pctmem */
226         { 288,  4,      0,      0 },            /* pr_euid */
227         { 292,  4,      0,      0 },            /* pr_egid */
228         { 296,  4,      0,      0 },            /* pr_aslwpid */
229         { 300,  1,      0,      0 },            /* pr_dmodel */
230 };


233 static const sl_lwpsinfo_layout_t lwpsinfo_layout = {
234         { 0,    128,    0,      0 },            /* sizeof (lwpsinfo_t) */
235         { 0,    4,      0,      1 },            /* pr_flag */
236         { 4,    4,      0,      0 },            /* pr_lwpid */
237         { 8,    8,      0,      0 },            /* pr_addr */
238         { 16,   8,      0,      0 },            /* pr_wchan */
239         { 24,   1,      0,      0 },            /* pr_stype */
240         { 25,   1,      0,      0 },            /* pr_state */
241         { 26,   1,      0,      0 },            /* pr_sname */
242         { 27,   1,      0,      0 },            /* pr_nice */
243         { 28,   2,      0,      0 },            /* pr_syscall */
244         { 30,   1,      0,      0 },            /* pr_oldpri */
245         { 31,   1,      0,      0 },            /* pr_cpu */
246         { 32,   4,      0,      1 },            /* pr_pri */
247         { 36,   2,      0,      0 },            /* pr_pctcpu */
248         { 40,   16,     0,      0 },            /* pr_start */
249         { 56,   16,     0,      0 },            /* pr_time */
250         { 72,   1,      8,      0 },            /* pr_clname[] */
251         { 80,   1,      16,     0 },            /* pr_name[] */
252         { 96,   4,      0,      1 },            /* pr_onpro */
253         { 100,  4,      0,      1 },            /* pr_bindpro */
254         { 104,  4,      0,      1 },            /* pr_bindpset */
255         { 108,  4,      0,      1 },            /* pr_lgrp */
256 };


259 static const sl_prcred_layout_t prcred_layout = {
260         { 0,    32,     0,      0 },            /* sizeof (prcred_t) */
261         { 0,    4,      0,      0 },            /* pr_euid */
262         { 4,    4,      0,      0 },            /* pr_ruid */
263         { 8,    4,      0,      0 },            /* pr_suid */
264         { 12,   4,      0,      0 },            /* pr_egid */
265         { 16,   4,      0,      0 },            /* pr_rgid */
```



```
266         { 20,   4,      0,      0 },            /* pr_sgid */
267         { 24,   4,      0,      1 },            /* pr_ngroups */
268         { 28,   4,      1,      0 },            /* pr_groups[] */
269 };


272 static const sl_prpriv_layout_t prpriv_layout = {
273         { 0,    16,     0,      0 },            /* sizeof (prpriv_t) */
274         { 0,    4,      0,      0 },            /* pr_nsets */
275         { 4,    4,      0,      0 },            /* pr_setsize */
276         { 8,    4,      0,      0 },            /* pr_infosize */
277         { 12,   4,      1,      0 },            /* pr_sets[] */
278 };


281 static const sl_priv_impl_info_layout_t priv_impl_info_layout = {
282         { 0,    28,     0,      0 },            /* sizeof (priv_impl_info_t) */
283         { 0,    4,      0,      0 },            /* priv_headersize */
284         { 4,    4,      0,      0 },            /* priv_flags */
285         { 8,    4,      0,      0 },            /* priv_nsets */
286         { 12,   4,      0,      0 },            /* priv_setsize */
287         { 16,   4,      0,      0 },            /* priv_max */
288         { 20,   4,      0,      0 },            /* priv_infosize */
289         { 24,   4,      0,      0 },            /* priv_globalinfosize */
290 };


293 static const sl_fltset_layout_t fltset_layout = {
294         { 0,    16,     0,      0 },            /* sizeof (fltset_t) */
295         { 0,    4,      4,      0 },            /* word[] */
296 };


299 static const sl_siginfo_layout_t siginfo_layout = {
300         { 0,    256,    0,      0 },            /* sizeof (siginfo_t) */
301         { 0,    4,      0,      0 },            /* si_signo */
302         { 8,    4,      0,      0 },            /* si_errno */
303         { 4,    4,      0,      1 },            /* si_code */
304         { 32,   4,      0,      0 },            /* si_value.sival_int */
305         { 32,   8,      0,      0 },            /* si_value.sival_ptr */
306         { 16,   4,      0,      0 },            /* si_pid */
307         { 24,   4,      0,      0 },            /* si_uid */
308         { 48,   4,      0,      0 },            /* si_ctid */
309         { 52,   4,      0,      0 },            /* si_zoneid */
310         { 16,   4,      0,      0 },            /* si_entity */
311         { 16,   8,      0,      0 },            /* si_addr */
312         { 32,   4,      0,      0 },            /* si_status */
313         { 24,   8,      0,      0 },            /* si_band */
314 };


317 static const sl_sigset_layout_t sigset_layout = {
318         { 0,    16,     0,      0 },            /* sizeof (sigset_t) */
319         { 0,    4,      4,      0 },            /* __sigbits[] */
320 };


323 static const sl_sigaction_layout_t sigaction_layout = {
324         { 0,    32,     0,      0 },            /* sizeof (struct sigaction) */
325         { 0,    4,      0,      0 },            /* sa_flags */
326         { 8,    8,      0,      0 },            /* sa_handler */
327         { 8,    8,      0,      0 },            /* sa_sigaction */
328         { 16,   16,     0,      0 },            /* sa_mask */
329 };
```

```
static const sl_stack_layout_t stack_layout = {
        { 0,    24,     0,      0 },            /* sizeof (stack_t) */
        { 0,    8,      0,      0 },            /* ss_sp */
        { 8,    8,      0,      0 },            /* ss_size */
        { 16,   4,      0,      0 },            /* ss_flags */
};


static const sl_sysset_layout_t sysset_layout = {
        { 0,    64,     0,      0 },            /* sizeof (sysset_t) */
        { 0,    4,      16,     0 },            /* word[] */
};


static const sl_timestruc_layout_t timestruc_layout = {
        { 0,    16,     0,      0 },            /* sizeof (timestruc_t) */
        { 0,    8,      0,      0 },            /* tv_sec */
        { 8,    8,      0,      0 },            /* tv_nsec */
};


static const sl_utsname_layout_t utsname_layout = {
        { 0,    1285,   0,      0 },            /* sizeof (struct utsname) */
        { 0,    1,      257,    0 },            /* sysname[] */
        { 257,  1,      257,    0 },            /* nodename[] */
        { 514,  1,      257,    0 },            /* release[] */
        { 771,  1,      257,    0 },            /* version[] */
        { 1028, 1,      257,    0 },            /* machine[] */
};


static const sl_prfdinfo_layout_t prfdinfo_layout = {
        { 0,    1088,   0,      0 },            /* sizeof (prfdinfo_t) */
        { 0,    4,      0,      0 },            /* pr_fd */
        { 4,    4,      0,      0 },            /* pr_mode */
        { 8,    4,      0,      0 },            /* pr_uid */
        { 12,   4,      0,      0 },            /* pr_gid */
        { 16,   4,      0,      0 },            /* pr_major */
        { 20,   4,      0,      0 },            /* pr_minor */
        { 24,   4,      0,      0 },            /* pr_rmajor */
        { 28,   4,      0,      0 },            /* pr_rminor */
        { 32,   8,      0,      0 },            /* pr_ino */
        { 40,   8,      0,      0 },            /* pr_offset */
        { 48,   8,      0,      0 },            /* pr_size */
        { 56,   4,      0,      0 },            /* pr_fileflags */
        { 60,   4,      0,      0 },            /* pr_fdflags */
        { 64,   1,      1024,   0 },            /* pr_path[] */
};


static const sl_psecflags_layout_t psecflags_layout = {
        { 0,    8,      0,      0 },            /* sizeof (psecflags_t) */
        { 0,    4,      0,      0 },            /* psf_effective */
        { 4,    4,      0,      0 },            /* psf_inherit */
};


#endif /* ! codereview */


static const sl_arch_layout_t layout_amd64 = {
        &auxv_layout,
        &fltset_layout,
        &lwpsinfo_layout,
        &lwpstatus_layout,
        &prcred_layout,
```



```
        &priv_impl_info_layout,
        &prpriv_layout,
        &psinfo_layout,
        &pstatus_layout,
        &prgregset_layout,
        &prpsinfo_layout,
        &prstatus_layout,
        &sigaction_layout,
        &siginfo_layout,
        &sigset_layout,
        &stack_layout,
        &sysset_layout,
        &timestruc_layout,
        &utsname_layout,
        &prfdinfo_layout,
        &psecflags_layout,
#endif /* ! codereview */
};


const sl_arch_layout_t *
struct_layout_amd64(void)
{
        return (&layout_amd64);
}
```

```
**********************************************************
   12432 Wed May 27 19:49:00 2015
new/usr/src/cmd/sgs/elfdump/common/struct_layout_i386.c
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
_____unchanged_portion_omitted_


  79 static const sl_pstatus_layout_t pstatus_layout = {
  80         { 0,    1136,   0,      0 },            /* sizeof (pstatus_t) */
  81         { 0,    4,      0,      1 },            /* pr_flags */
  82         { 4,    4,      0,      1 },            /* pr_nlwp */
  83         { 8,    4,      0,      0 },            /* pr_pid */
  84         { 12,   4,      0,      0 },            /* pr_ppid */
  85         { 16,   4,      0,      0 },            /* pr_pgid */
  86         { 20,   4,      0,      0 },            /* pr_sid */
  87         { 24,   4,      0,      1 },            /* pr_aslwpid */
  88         { 28,   4,      0,      1 },            /* pr_agentid */
  89         { 32,   16,     0,      0 },            /* pr_sigpend */
  90         { 48,   4,      0,      0 },            /* pr_brkbase */
  91         { 52,   4,      0,      0 },            /* pr_brksize */
  92         { 56,   4,      0,      0 },            /* pr_stkbase */
  93         { 60,   4,      0,      0 },            /* pr_stksize */
  94         { 64,   8,      0,      0 },            /* pr_utime */
  95         { 72,   8,      0,      0 },            /* pr_stime */
  96         { 80,   8,      0,      0 },            /* pr_cutime */
  97         { 88,   8,      0,      0 },            /* pr_cstime */
  98         { 96,   16,     0,      0 },            /* pr_sigtrace */
  99         { 112,  16,     0,      0 },            /* pr_flttrace */
 100         { 128,  64,     0,      0 },            /* pr_sysentry */
 101         { 192,  64,     0,      0 },            /* pr_sysexit */
 102         { 256,  1,      0,      0 },            /* pr_dmodel */
 103         { 260,  4,      0,      1 },            /* pr_taskid */
 104         { 264,  4,      0,      1 },            /* pr_projid */
 105         { 268,  4,      0,      1 },            /* pr_nzomb */
 106         { 272,  4,      0,      1 },            /* pr_zoneid */
 107         { 276,  8,      0,      0 },            /* pr_secflags */
 108 #endif /* ! codereview */
 109         { 336,  800,    0,      0 },            /* pr_lwp */
 110 };


 113 static const sl_prstatus_layout_t prstatus_layout = {
 114         { 0,    432,    0,      0 },            /* sizeof (prstatus_t) */
 115         { 0,    4,      0,      1 },            /* pr_flags */
 116         { 4,    2,      0,      1 },            /* pr_why */
 117         { 6,    2,      0,      1 },            /* pr_what */
 118         { 8,    128,    0,      0 },            /* pr_info */
 119         { 136,  2,      0,      1 },            /* pr_cursig */
 120         { 138,  2,      0,      0 },            /* pr_nlwp */
 121         { 140,  16,     0,      0 },            /* pr_sigpend */
 122         { 156,  16,     0,      0 },            /* pr_sighold */
 123         { 172,  12,     0,      0 },            /* pr_altstack */
 124         { 184,  32,     0,      0 },            /* pr_action */
 125         { 216,  4,      0,      0 },            /* pr_pid */
 126         { 220,  4,      0,      0 },            /* pr_ppid */
 127         { 224,  4,      0,      0 },            /* pr_pgrp */
 128         { 228,  4,      0,      0 },            /* pr_sid */
 129         { 232,  8,      0,      0 },            /* pr_utime */
 130         { 240,  8,      0,      0 },            /* pr_stime */
 131         { 248,  8,      0,      0 },            /* pr_cutime */
 132         { 256,  8,      0,      0 },            /* pr_cstime */
 133         { 264,  1,      8,      0 },            /* pr_clname[] */
```



```
 134         { 272,  2,      0,      1 },            /* pr_syscall */
 135         { 274,  2,      0,      1 },            /* pr_nsysarg */
 136         { 276,  4,      8,      1 },            /* pr_sysarg[] */
 137         { 308,  4,      0,      0 },            /* pr_who */
 138         { 312,  16,     0,      0 },            /* pr_lwppend */
 139         { 328,  4,      0,      0 },            /* pr_oldcontext */
 140         { 332,  4,      0,      0 },            /* pr_brkbase */
 141         { 336,  4,      0,      0 },            /* pr_brksize */
 142         { 340,  4,      0,      0 },            /* pr_stkbase */
 143         { 344,  4,      0,      0 },            /* pr_stksize */
 144         { 348,  2,      0,      1 },            /* pr_processor */
 145         { 350,  2,      0,      1 },            /* pr_bind */
 146         { 352,  4,      0,      1 },            /* pr_instr */
 147         { 356,  76,     0,      0 },            /* pr_reg */
 148 };


 151 static const sl_psinfo_layout_t psinfo_layout = {
 152         { 0,    336,    0,      0 },            /* sizeof (psinfo_t) */
 153         { 0,    4,      0,      1 },            /* pr_flag */
 154         { 4,    4,      0,      1 },            /* pr_nlwp */
 155         { 8,    4,      0,      0 },            /* pr_pid */
 156         { 12,   4,      0,      0 },            /* pr_ppid */
 157         { 16,   4,      0,      0 },            /* pr_pgid */
 158         { 20,   4,      0,      0 },            /* pr_sid */
 159         { 24,   4,      0,      0 },            /* pr_uid */
 160         { 28,   4,      0,      0 },            /* pr_euid */
 161         { 32,   4,      0,      0 },            /* pr_gid */
 162         { 36,   4,      0,      0 },            /* pr_egid */
 163         { 40,   4,      0,      0 },            /* pr_addr */
 164         { 44,   4,      0,      0 },            /* pr_size */
 165         { 48,   4,      0,      0 },            /* pr_rssize */
 166         { 56,   4,      0,      0 },            /* pr_ttydev */
 167         { 60,   2,      0,      0 },            /* pr_pctcpu */
 168         { 62,   2,      0,      0 },            /* pr_pctmem */
 169         { 64,   8,      0,      0 },            /* pr_start */
 170         { 72,   8,      0,      0 },            /* pr_time */
 171         { 80,   8,      0,      0 },            /* pr_ctime */
 172         { 88,   1,      16,     0 },            /* pr_fname[] */
 173         { 104,  1,      80,     0 },            /* pr_psargs[] */
 174         { 184,  4,      0,      1 },            /* pr_wstat */
 175         { 188,  4,      0,      1 },            /* pr_argc */
 176         { 192,  4,      0,      0 },            /* pr_argv */
 177         { 196,  4,      0,      0 },            /* pr_envp */
 178         { 200,  1,      0,      0 },            /* pr_dmodel */
 179         { 204,  4,      0,      0 },            /* pr_taskid */
 180         { 208,  4,      0,      0 },            /* pr_projid */
 181         { 212,  4,      0,      1 },            /* pr_nzomb */
 182         { 216,  4,      0,      0 },            /* pr_poolid */
 183         { 220,  4,      0,      0 },            /* pr_zoneid */
 184         { 224,  4,      0,      0 },            /* pr_contract */
 185         { 232,  104,    0,      0 },            /* pr_lwp */
 186 };


 189 static const sl_prpsinfo_layout_t prpsinfo_layout = {
 190         { 0,    260,    0,      0 },            /* sizeof (prpsinfo_t) */
 191         { 0,    1,      0,      0 },            /* pr_state */
 192         { 1,    1,      0,      0 },            /* pr_sname */
 193         { 2,    1,      0,      0 },            /* pr_zomb */
 194         { 3,    1,      0,      0 },            /* pr_nice */
 195         { 4,    4,      0,      0 },            /* pr_flag */
 196         { 8,    4,      0,      0 },            /* pr_uid */
 197         { 12,   4,      0,      0 },            /* pr_gid */
 198         { 16,   4,      0,      0 },            /* pr_pid */
 199         { 20,   4,      0,      0 },            /* pr_ppid */
```

```
200         { 24,   4,      0,      0 },            /* pr_pgrp */
201         { 28,   4,      0,      0 },            /* pr_sid */
202         { 32,   4,      0,      0 },            /* pr_addr */
203         { 36,   4,      0,      0 },            /* pr_size */
204         { 40,   4,      0,      0 },            /* pr_rssize */
205         { 44,   4,      0,      0 },            /* pr_wchan */
206         { 48,   8,      0,      0 },            /* pr_start */
207         { 56,   8,      0,      0 },            /* pr_time */
208         { 64,   4,      0,      1 },            /* pr_pri */
209         { 68,   1,      0,      0 },            /* pr_oldpri */
210         { 69,   1,      0,      0 },            /* pr_cpu */
211         { 70,   2,      0,      0 },            /* pr_ottydev */
212         { 72,   4,      0,      0 },            /* pr_lttydev */
213         { 76,   1,      8,      0 },            /* pr_clname[] */
214         { 84,   1,      16,     0 },            /* pr_fname[] */
215         { 100,  1,      80,     0 },            /* pr_psargs[] */
216         { 180,  2,      0,      1 },            /* pr_syscall */
217         { 184,  8,      0,      0 },            /* pr_ctime */
218         { 192,  4,      0,      0 },            /* pr_bysize */
219         { 196,  4,      0,      0 },            /* pr_byrssize */
220         { 200,  4,      0,      1 },            /* pr_argc */
221         { 204,  4,      0,      0 },            /* pr_argv */
222         { 208,  4,      0,      0 },            /* pr_envp */
223         { 212,  4,      0,      1 },            /* pr_wstat */
224         { 216,  2,      0,      0 },            /* pr_pctcpu */
225         { 218,  2,      0,      0 },            /* pr_pctmem */
226         { 220,  4,      0,      0 },            /* pr_euid */
227         { 224,  4,      0,      0 },            /* pr_egid */
228         { 228,  4,      0,      0 },            /* pr_aslwpid */
229         { 232,  1,      0,      0 },            /* pr_dmodel */
230 };


233 static const sl_lwpsinfo_layout_t lwpsinfo_layout = {
234         { 0,    104,    0,      0 },            /* sizeof (lwpsinfo_t) */
235         { 0,    4,      0,      1 },            /* pr_flag */
236         { 4,    4,      0,      0 },            /* pr_lwpid */
237         { 8,    4,      0,      0 },            /* pr_addr */
238         { 12,   4,      0,      0 },            /* pr_wchan */
239         { 16,   1,      0,      0 },            /* pr_stype */
240         { 17,   1,      0,      0 },            /* pr_state */
241         { 18,   1,      0,      0 },            /* pr_sname */
242         { 19,   1,      0,      0 },            /* pr_nice */
243         { 20,   2,      0,      0 },            /* pr_syscall */
244         { 22,   1,      0,      0 },            /* pr_oldpri */
245         { 23,   1,      0,      0 },            /* pr_cpu */
246         { 24,   4,      0,      1 },            /* pr_pri */
247         { 28,   2,      0,      0 },            /* pr_pctcpu */
248         { 32,   8,      0,      0 },            /* pr_start */
249         { 40,   8,      0,      0 },            /* pr_time */
250         { 48,   1,      8,      0 },            /* pr_clname[] */
251         { 56,   1,      16,     0 },            /* pr_name[] */
252         { 72,   4,      0,      1 },            /* pr_onpro */
253         { 76,   4,      0,      1 },            /* pr_bindpro */
254         { 80,   4,      0,      1 },            /* pr_bindpset */
255         { 84,   4,      0,      1 },            /* pr_lgrp */
256 };


259 static const sl_prcred_layout_t prcred_layout = {
260         { 0,    32,     0,      0 },            /* sizeof (prcred_t) */
261         { 0,    4,      0,      0 },            /* pr_euid */
262         { 4,    4,      0,      0 },            /* pr_ruid */
263         { 8,    4,      0,      0 },            /* pr_suid */
264         { 12,   4,      0,      0 },            /* pr_egid */
265         { 16,   4,      0,      0 },            /* pr_rgid */
```



```
266         { 20,   4,      0,      0 },            /* pr_sgid */
267         { 24,   4,      0,      1 },            /* pr_ngroups */
268         { 28,   4,      1,      0 },            /* pr_groups[] */
269 };


272 static const sl_prpriv_layout_t prpriv_layout = {
273         { 0,    16,     0,      0 },            /* sizeof (prpriv_t) */
274         { 0,    4,      0,      0 },            /* pr_nsets */
275         { 4,    4,      0,      0 },            /* pr_setsize */
276         { 8,    4,      0,      0 },            /* pr_infosize */
277         { 12,   4,      1,      0 },            /* pr_sets[] */
278 };


281 static const sl_priv_impl_info_layout_t priv_impl_info_layout = {
282         { 0,    28,     0,      0 },            /* sizeof (priv_impl_info_t) */
283         { 0,    4,      0,      0 },            /* priv_headersize */
284         { 4,    4,      0,      0 },            /* priv_flags */
285         { 8,    4,      0,      0 },            /* priv_nsets */
286         { 12,   4,      0,      0 },            /* priv_setsize */
287         { 16,   4,      0,      0 },            /* priv_max */
288         { 20,   4,      0,      0 },            /* priv_infosize */
289         { 24,   4,      0,      0 },            /* priv_globalinfosize */
290 };


293 static const sl_fltset_layout_t fltset_layout = {
294         { 0,    16,     0,      0 },            /* sizeof (fltset_t) */
295         { 0,    4,      4,      0 },            /* word[] */
296 };


299 static const sl_siginfo_layout_t siginfo_layout = {
300         { 0,    128,    0,      0 },            /* sizeof (siginfo_t) */
301         { 0,    4,      0,      0 },            /* si_signo */
302         { 8,    4,      0,      0 },            /* si_errno */
303         { 4,    4,      0,      1 },            /* si_code */
304         { 20,   4,      0,      0 },            /* si_value.sival_int */
305         { 20,   4,      0,      0 },            /* si_value.sival_ptr */
306         { 12,   4,      0,      0 },            /* si_pid */
307         { 16,   4,      0,      0 },            /* si_uid */
308         { 28,   4,      0,      0 },            /* si_ctid */
309         { 32,   4,      0,      0 },            /* si_zoneid */
310         { 12,   4,      0,      0 },            /* si_entity */
311         { 12,   4,      0,      0 },            /* si_addr */
312         { 20,   4,      0,      0 },            /* si_status */
313         { 16,   4,      0,      0 },            /* si_band */
314 };


317 static const sl_sigset_layout_t sigset_layout = {
318         { 0,    16,     0,      0 },            /* sizeof (sigset_t) */
319         { 0,    4,      4,      0 },            /* __sigbits[] */
320 };


323 static const sl_sigaction_layout_t sigaction_layout = {
324         { 0,    32,     0,      0 },            /* sizeof (struct sigaction) */
325         { 0,    4,      0,      0 },            /* sa_flags */
326         { 4,    4,      0,      0 },            /* sa_handler */
327         { 4,    4,      0,      0 },            /* sa_sigaction */
328         { 8,    16,     0,      0 },            /* sa_mask */
329 };
```

```
static const sl_stack_layout_t stack_layout = {
        { 0,    12,     0,      0 },            /* sizeof (stack_t) */
        { 0,    4,      0,      0 },            /* ss_sp */
        { 4,    4,      0,      0 },            /* ss_size */
        { 8,    4,      0,      0 },            /* ss_flags */
};


static const sl_sysset_layout_t sysset_layout = {
        { 0,    64,     0,      0 },            /* sizeof (sysset_t) */
        { 0,    4,      16,     0 },            /* word[] */
};


static const sl_timestruc_layout_t timestruc_layout = {
        { 0,    8,      0,      0 },            /* sizeof (timestruc_t) */
        { 0,    4,      0,      0 },            /* tv_sec */
        { 4,    4,      0,      0 },            /* tv_nsec */
};


static const sl_utsname_layout_t utsname_layout = {
        { 0,    1285,   0,      0 },            /* sizeof (struct utsname) */
        { 0,    1,      257,    0 },            /* sysname[] */
        { 257,  1,      257,    0 },            /* nodename[] */
        { 514,  1,      257,    0 },            /* release[] */
        { 771,  1,      257,    0 },            /* version[] */
        { 1028, 1,      257,    0 },            /* machine[] */
};


static const sl_prfdinfo_layout_t prfdinfo_layout = {
        { 0,    1088,   0,      0 },            /* sizeof (prfdinfo_t) */
        { 0,    4,      0,      0 },            /* pr_fd */
        { 4,    4,      0,      0 },            /* pr_mode */
        { 8,    4,      0,      0 },            /* pr_uid */
        { 12,   4,      0,      0 },            /* pr_gid */
        { 16,   4,      0,      0 },            /* pr_major */
        { 20,   4,      0,      0 },            /* pr_minor */
        { 24,   4,      0,      0 },            /* pr_rmajor */
        { 28,   4,      0,      0 },            /* pr_rminor */
        { 32,   8,      0,      0 },            /* pr_ino */
        { 40,   8,      0,      0 },            /* pr_offset */
        { 48,   8,      0,      0 },            /* pr_size */
        { 56,   4,      0,      0 },            /* pr_fileflags */
        { 60,   4,      0,      0 },            /* pr_fdflags */
        { 64,   1,      1024,   0 },            /* pr_path[] */
};


static const sl_psecflags_layout_t psecflags_layout = {
        { 0,    8,      0,      0 },            /* sizeof (psecflags_t) */
        { 0,    4,      0,      0 },            /* psf_effective */
        { 4,    4,      0,      0 },            /* psf_inherit */
};


#endif /* ! codereview */


static const sl_arch_layout_t layout_i386 = {
        &auxv_layout,
        &fltset_layout,
        &lwpsinfo_layout,
        &lwpstatus_layout,
        &prcred_layout,
```



```
        &priv_impl_info_layout,
        &prpriv_layout,
        &psinfo_layout,
        &pstatus_layout,
        &prgregset_layout,
        &prpsinfo_layout,
        &prstatus_layout,
        &sigaction_layout,
        &siginfo_layout,
        &sigset_layout,
        &stack_layout,
        &sysset_layout,
        &timestruc_layout,
        &utsname_layout,
        &prfdinfo_layout,
        &psecflags_layout,
#endif /* ! codereview */
};


const sl_arch_layout_t *
struct_layout_i386(void)
{
        return (&layout_i386);
}
```

```
**********************************************************
   41161 Wed May 27 19:49:01 2015
new/usr/src/cmd/sgs/include/conv.h
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
______unchanged_portion_omitted_
```

```
328 /* conv_psecflags() */
329 #define CONV_PSECFLAGS_BUFSIZE          31
330 typedef union {
331         Conv_inv_buf_t                  inv_buf;
332         char                            buf[CONV_PSECFLAGS_BUFSIZE];
333 } Conv_secflags_buf_t;
334 #endif /* ! codereview */

336 /* conv_cnote_sigset() */
337 #define CONV_CNOTE_SIGSET_BUFSIZE       639
338 typedef union {
339         Conv_inv_buf_t                  inv_buf;
340         char                            buf[CONV_CNOTE_SIGSET_BUFSIZE];
341 } Conv_cnote_sigset_buf_t;

343 /* conv_cnote_fltset() */
344 #define CONV_CNOTE_FLTSET_BUFSIZE       511
345 typedef union {
346         Conv_inv_buf_t                  inv_buf;
347         char                            buf[CONV_CNOTE_FLTSET_BUFSIZE];
348 } Conv_cnote_fltset_buf_t;

350 /* conv_cnote_sysset() */
351 #define CONV_CNOTE_SYSSET_BUFSIZE       3195
352 typedef union {
353         Conv_inv_buf_t                  inv_buf;
354         char                            buf[CONV_CNOTE_SYSSET_BUFSIZE];
355 } Conv_cnote_sysset_buf_t;

357 /* conv_cnote_sa_flags() */
358 #define CONV_CNOTE_SA_FLAGS_BUFSIZE     109
359 typedef union {
360         Conv_inv_buf_t                  inv_buf;
361         char                            buf[CONV_CNOTE_SA_FLAGS_BUFSIZE];
362 } Conv_cnote_sa_flags_buf_t;

364 /* conv_cnote_ss_flags() */
365 #define CONV_CNOTE_SS_FLAGS_BUFSIZE     48
366 typedef union {
367         Conv_inv_buf_t                  inv_buf;
368         char                            buf[CONV_CNOTE_SS_FLAGS_BUFSIZE];
369 } Conv_cnote_ss_flags_buf_t;

371 /* conv_cnote_cc_content() */
372 #define CONV_CNOTE_CC_CONTENT_BUFSIZE   97
373 typedef union {
374         Conv_inv_buf_t                  inv_buf;
375         char                            buf[CONV_CNOTE_CC_CONTENT_BUFSIZE];
376 } Conv_cnote_cc_content_buf_t;

378 /* conv_cnote_auxv_af() */
379 #define CONV_CNOTE_AUXV_AF_BUFSIZE      73
380 typedef union {
381         Conv_inv_buf_t                  inv_buf;
382         char                            buf[CONV_CNOTE_AUXV_AF_BUFSIZE];
383 } Conv_cnote_auxv_af_buf_t;
```



```
385 /* conv_ver_flags() */
386 #define CONV_VER_FLAGS_BUFSIZE          41
387 typedef union {
388         Conv_inv_buf_t                  inv_buf;
389         char                            buf[CONV_VER_FLAGS_BUFSIZE];
390 } Conv_ver_flags_buf_t;

392 /* conv_ent_flags() */
393 #define CONV_ENT_FLAGS_BUFSIZE          69
394 typedef union {
395         Conv_inv_buf_t                  inv_buf;
396         char                            buf[CONV_ENT_FLAGS_BUFSIZE];
397 } Conv_ent_flags_buf_t;

399 /* conv_ent_files_flags() */
400 #define CONV_ENT_FILES_FLAGS_BUFSIZE    89
401 typedef union {
402         Conv_inv_buf_t                  inv_buf;
403         char                            buf[CONV_ENT_FILES_FLAGS_BUFSIZE];
404 } Conv_ent_files_flags_buf_t;

406 /*
407  * conv_time()
408  *
409  * This size is based on the maximum "hour.min.sec.fraction: " time that
410  * would be expected of ld().
411  */
412 #define CONV_TIME_BUFSIZE               18
413 typedef union {
414         char                            buf[CONV_TIME_BUFSIZE];
415 } Conv_time_buf_t;

417 /*
418  * Many conversion routines accept a fmt_flags argument of this type
419  * to allow the caller to modify the output. There are two parts to
420  * this value:
421  *
422  *      (1) Format requests (decimal vs hex, etc...)
423  *      (2) The low order bits specified by CONV_MASK_FMT_ALT
424  *              and retrieved by CONV_TYPE_FMT_ALT are integer
425  *              values that specify that an alternate set of
426  *              strings should be used.
427  *
428  * The fmt_flags value is designed such that a caller can always
429  * supply a 0 in order to receive default behavior.
430  */
431 typedef int Conv_fmt_flags_t;

433 /*
434  * Type used to represent ELF constants within libconv. This relies on
435  * the fact that there are no ELF constants that need more than 32-bits,
436  * nor are there any signed values.
437  */
438 typedef uint32_t Conv_elfvalue_t;

440 /*
441  * Most conversion routines are able to provide strings in one of
442  * several alternative styles. The bottom 8 bits of Conv_fmt_flags_t
443  * are used to specify which strings should be used for a given call
444  * to a conversion routine:
445  *
446  *   DEFAULT
447  *      The default string style used by a given conversion routine is
448  *      an independent choice made by that routine. Different routines
449  *      make different choices, based largely on historical usage and
```

```
 *      the perceived common case. It may be an alias for one of the
 *      specific styles listed below, or it may be unique.
 *
 *   DUMP
 *      Style of strings used by dump(1).
 *
 *   FILE
 *      Style of strings used by file(1).
 *
 *   CRLE
 *      Style of strings used by crle(1).
 *
 *   CF
 *      Canonical Form: The string is exactly the same as the name
 *      of the #define macro that defines it in the public header files.
 *      (e.g. STB_LOCAL, not LOCL, LOCAL, LOC, or any other variation).
 *
 *   CFNP
 *      No Prefix Canonical Form: The same strings supplied by CF,
 *      but without their standard prefix. (e.g. LOCAL, instead of STT_LOCAL).
 *
 *   NF
 *      Natural Form: The form of the strings that might typically be entered
 *      via a keyboard by an interactive user. These are usually the strings
 *      from CFNP, converted to lowercase, although in some cases they may
 *      take some other "natural" form. In command completion applications,
 *      lowercase strings appear less formal, and are easier on the eye.
 *
 * Every routine is required to have a default style. The others are optional,
 * and may not be provided if not needed. If a given conversion routine does
 * not support alternative strings for a given CONV_FMT_ALT type, it silently
 * ignores the request and supplies the default set. This means that a utility
 * like dump(1) is free to specify a style like DUMP to every conversion
 * routine. It will receive its special strings if there are any, and
 * the defaults otherwise.
 */
#define CONV_MASK_FMT_ALT               0xff
#define CONV_TYPE_FMT_ALT(fmt_flags)    (fmt_flags & CONV_MASK_FMT_ALT)

#define CONV_FMT_ALT_DEFAULT    0       /* "Standard" strings */
#define CONV_FMT_ALT_DUMP       1       /* dump(1) */
#define CONV_FMT_ALT_FILE       2       /* file(1) */
#define CONV_FMT_ALT_CRLE       3       /* crle(1) */
#define CONV_FMT_ALT_CF         4       /* Canonical Form */
#define CONV_FMT_ALT_CFNP       5       /* No Prefix Canonical Form */
#define CONV_FMT_ALT_NF         6       /* Natural Form */

/*
 * Flags that alter standard formatting for conversion routines.
 * These bits start after the range occupied by CONV_MASK_FMT_ALT.
 */
#define CONV_FMT_DECIMAL        0x0100  /* conv_invalid_val() should print */
                                        /*    integer print as decimal */
                                        /*    (default is hex) */
#define CONV_FMT_SPACE          0x0200  /* conv_invalid_val() should append */
                                        /*    a space after the number.  */
#define CONV_FMT_NOBKT          0x0400  /* conv_expn_field() should omit */
                                        /*    prefix and suffix strings */

/*
 * A Val_desc structure is used to associate an ELF constant and
 * the message code (Msg) for the string that corresponds to it.
 *
 * Val_desc2 adds v_osabi and v_mach fields to Val_desc, which allows
 * for non-generic mappings that apply only to a specific OSABI/machine.
 * Setting v_osabi to 0 (ELFOSABI_NONE) specifies that any OSABI matches.
```



```
 * Similarly, setting v_mach to 0 (EM_MACH) matches any machine. Hence,
 * setting v_osabi and v_mach to 0 in a Val_desc2 results in a generic item,
 * and is equivalent to simply using a Val_desc.
 *
 * These structs are used in two different contexts:
 *
 * 1)   To expand bit-field data items, using conv_expn_field() to
 *      process a NULL terminated array of Val_desc, or conv_expn_field2()
 *      to process a null terminated array of Val_desc2.
 *
 * 2)   To represent sparse ranges of non-bitfield values, referenced via
 *      conv_ds_vd_t or conv_ds_vd2_t descriptors, as described below.
 */
typedef struct {
        Conv_elfvalue_t v_val;          /* expansion value */
        Msg             v_msg;          /* associated message string code */
} Val_desc;
typedef struct {
        Conv_elfvalue_t v_val;          /* expansion value */
        uchar_t         v_osabi;        /* OSABI to which entry applies */
        Half            v_mach;         /* Machine to which entry applies */
        Msg             v_msg;          /* associated message string code */
} Val_desc2;

/*
 * The conv_ds_XXX_t structs are used to pull together the information used
 * to map non-bitfield values to strings. They are a variant family, sharing
 * the same initial fields, with a generic "header" definition that can be
 * used to read those common fields and determine which subcase is being
 * seen. We do this instead of using a single struct containing a type code
 * and a union in order to allow for static compile-time initialization.
 *
 * conv_ds_t is the base type, containing the initial fields common to all
 * the variants. Variables of type conv_ds_t are never instantiated. This
 * type exists only to provide a common pointer type that can reference
 * any of the variants safely. In C++, it would be a virtual base class.
 * The fields common to all the variants are:
 *
 *      ds_type: Identifies the variant
 *      ds_baseval/ds_topval: The lower and upper bound of the range
 *              of values represented by this conv_ds_XXX_t descriptor.
 *
 * There are three different variants:
 *    conv_ds_msg_t (ds_type == CONV_DS_MSGARR)
 *      This structure references an array of message codes corresponding
 *      to consecutive ELF values. The first item in the array is the Msg
 *      code for the value given by ds_baseval. Consecutive strings follow
 *      in consecutive order. The final item corresponds to the value given
 *      by ds_topval. Zero (0) Msg values can be used to represent missing
 *      values. Entries with a 0 are quietly ignored.
 *
 *    conv_ds_vd_t (ds_type == CONV_DS_VD)
 *      This structure employs a NULL terminated array of Val_desc structs.
 *      Each Val_desc supplies a mapping from a value in the range
 *      (ds_baseval <= value <= ds_topval). The values described need not
 *      be consecutive, and can be sparse. ds_baseval does not need to
 *      correspond to the first item, and ds_topval need not correspond to
 *      the final item.
 *
 *    conv_ds_vd2_t (ds_type == CONV_DS_VD2)
 *      This structure employs a NULL terminated array of Val_desc2 structs,
 *      rather than Val_desc, adding the ability to specify OSABI and machine
 *      as part of the value/string mapping. It is otherwise the same thing
 *      as CONV_DS_VD.
 */
typedef enum {
```

```
582         CONV_DS_MSGARR = 0,             /* Array of Msg */
583         CONV_DS_VD = 1,                 /* Null terminated array of Val_desc */
584         CONV_DS_VD2 = 2,                /* Null terminated array of Val_desc2 */
585 } conv_ds_type_t;

587 #define CONV_DS_COMMON_FIELDS \
588         conv_ds_type_t  ds_type;        /* Type of data structure used */ \
589         uint32_t        ds_baseval;     /* Value of first item */       \
590         uint32_t        ds_topval       /* Value of last item */

592 typedef struct {                /* Virtual base type --- do not instantiate */
593         CONV_DS_COMMON_FIELDS;
594 } conv_ds_t;
595 typedef struct {
596         CONV_DS_COMMON_FIELDS;
597         const Msg               *ds_msg;
598 } conv_ds_msg_t;
599 typedef struct {
600         CONV_DS_COMMON_FIELDS;
601         const Val_desc          *ds_vd;
602 } conv_ds_vd_t;
603 typedef struct {
604         CONV_DS_COMMON_FIELDS;
605         const Val_desc2         *ds_vd2;
606 } conv_ds_vd2_t;

608 /*
609  * The initialization of conv_ds_msg_t can be completely derived from
610  * its base value and the array of Msg codes. CONV_DS_MSG_INIT() is used
611  * to do that.
612  */
613 #define CONV_DS_MSG_INIT(_baseval, _arr) \
614         CONV_DS_MSGARR, _baseval, \
615         _baseval + (sizeof (_arr) / sizeof (_arr[0])) - 1, _arr

617 /*
618  * Null terminated arrays of pointers to conv_ds_XXX_t structs are processed
619  * by conv_map_ds() to convert ELF constants to their symbolic names, and by
620  * conv_iter_ds() to iterate over all the available value/name combinations.
621  *
622  * These pointers are formed by casting the address of the specific
623  * variant types (described above) to generic base type pointer.
624  * CONV_DS_ADDR() is a convenience macro to take the address of
625  * one of these variants and turn it into a generic pointer.
626  */
627 #define CONV_DS_ADDR(_item) ((conv_ds_t *)&(_item))

629 /*
630  * Type used by libconv to represent osabi values passed to iteration
631  * functions. The type in the ELF header is uchar_t. However, every possible
632  * value 0-255 has a valid meaning, leaving us no extra value to assign
633  * to mean "ALL". Using Half for osabi leaves us the top byte to use for
634  * out of bound values.
635  *
636  * Non-iteration functions, and any code that does not need to use
637  * CONV_OSABI_ALL, should use uchar_t for osabi.
638  */
639 typedef Half conv_iter_osabi_t;

641 /*
642  * Many of the iteration functions accept an osabi or mach argument,
643  * used to specify the type of object being processed. The following
644  * values can be used to specify a wildcard that matches any item. Their
645  * values are carefully chosen to ensure that they cannot be interpreted
646  * as an otherwise valid osabi or machine.
647  */
```



```
648 #define CONV_OSABI_ALL  1024    /* Larger than can be represented by uchar_t */
649 #define CONV_MACH_ALL   EM_NUM  /* Never a valid machine type */

651 /*
652  * We compare Val_Desc2 descriptors with a specified osabi and machine
653  * to determine whether to use it or not. This macro encapsulates that logic.
654  *
655  * We consider an osabi to match when any of the following things hold:
656  *
657  * -    The descriptor osabi is ELFOSABI_NONE.
658  * -    The supplied osabi and the descriptor osabi match
659  * -    The supplied osabi is ELFOSABI_NONE, and the descriptor osabi is
660  *      ELFOSABI_SOLARIS. Many operating systems, Solaris included,
661  *      produce or have produced ELFOSABI_NONE native objects, if only
662  *      because OSABI ranges are not an original ELF feature. We
663  *      give our own objects the home field advantage.
664  * -    Iteration Only: An osabi value of CONV_OSABI_ALL is specified.
665  *
666  * We consider a machine to match when any of the following things hold:
667  *
668  * -    The descriptor mach is EM_NONE.
669  * -    The supplied mach and the descriptor mach match
670  * -    Iteration Only: A mach value of CONV_MACH_ALL is specified.
671  *
672  * The special extra _ALL case for iteration is handled by defining a separate
673  * macro with the extra CONV_xxx_ALL tests.
674  */
675 #define CONV_VD2_SKIP_OSABI(_osabi, _vdp) \
676         ((_vdp->v_osabi != ELFOSABI_NONE) && (_vdp->v_osabi != osabi) && \
677         ((_osabi != ELFOSABI_NONE) || (_vdp->v_osabi != ELFOSABI_SOLARIS)))

679 #define CONV_VD2_SKIP_MACH(_mach, _vdp) \
680         ((_vdp->v_mach != EM_NONE) && (_vdp->v_mach != _mach))

682 #define CONV_VD2_SKIP(_osabi, _mach, _vdp) \
683         (CONV_VD2_SKIP_OSABI(_osabi, _vdp) || CONV_VD2_SKIP_MACH(_mach, _vdp))

685 #define CONV_ITER_VD2_SKIP(_osabi, _mach, _vdp)                 \
686         ((CONV_VD2_SKIP_OSABI(_osabi, _vdp) && (_osabi != CONV_OSABI_ALL)) || \
687         (CONV_VD2_SKIP_MACH(_mach, _vdp) && (_mach != CONV_MACH_ALL)))


690 /*
691  * Possible return values from iteration functions.
692  */
693 typedef enum {
694         CONV_ITER_DONE,         /* Stop: No more iterations are desired */
695         CONV_ITER_CONT          /* Continue with following iterations */
696 } conv_iter_ret_t;

698 /*
699  * Prototype for caller supplied callback function to iteration functions.
700  */
701 typedef conv_iter_ret_t (* conv_iter_cb_t)(const char *str,
702     Conv_elfvalue_t value, void *uvalue);

704 /*
705  * User value block employed by conv_iter_strtol()
706  */
707 typedef struct {
708         const char      *csl_str;       /* String to search for */
709         size_t          csl_strlen;     /* # chars in csl_str to examine */
710         int             csl_found;      /* Init to 0, set to 1 if item found */
711         Conv_elfvalue_t csl_value;      /* If csl_found, resulting value */
712 } conv_strtol_uvalue_t;
```

```
714 /*
715  * conv_expn_field() is willing to supply default strings for the
716  * prefix, separator, and suffix arguments, if they are passed as NULL.
717  * The caller needs to know how much room to allow for these items.
718  * These values supply those sizes.
719  */
720 #define CONV_EXPN_FIELD_DEF_PREFIX_SIZE 2       /* Default is "[ " */
721 #define CONV_EXPN_FIELD_DEF_SEP_SIZE    1       /* Default is " " */
722 #define CONV_EXPN_FIELD_DEF_SUFFIX_SIZE 2       /* Default is " ]" */

724 /*
725  * conv_expn_field() requires a large number of inputs, many of which
726  * can be NULL to accept default behavior. An argument of the following
727  * type is used to supply them.
728  */
729 typedef struct {
730         char *buf;              /* Buffer to receive generated string */
731         size_t bufsize;         /* sizeof(buf) */
732         const char **lead_str;  /* NULL, or array of pointers to strings to */
733                                 /*      be output at the head of the list. */
734                                 /*      Last entry must be NULL. */
735         Xword oflags;           /* Bits for which output strings are desired */
736         Xword rflags;           /* Bits for which a numeric value should be */
737                                 /*      output if vdp does not provide str. */
738                                 /*      Must be a proper subset of oflags */
739         const char *prefix;     /* NULL, or string to prefix output with */
740                                 /*      If NULL, "[ " is used. */
741         const char *sep;        /* NULL, or string to separate output items */
742                                 /*      with. If NULL, " " is used. */
743         const char *suffix;     /* NULL, or string to suffix output with */
744                                 /*      If NULL, " ]" is used. */
745 } CONV_EXPN_FIELD_ARG;

747 /*
748  * Callback function for conv_str_to_c_literal(). A user supplied function
749  * of this type is called by conv_str_to_c_literal() in order to dispatch
750  * the translated output characters.
751  *
752  *      buf - Pointer to output text
753  *      n - # of characters to output
754  *      uvalue - User value argument to conv_str_to_c_literal(),
755  *               passed through without interpretation.
756  */
757 typedef void            Conv_str_to_c_literal_func_t(const void *ptr,
758                             size_t size, void *uvalue);

760 /*
761  * Generic miscellaneous interfaces
762  */
763 extern  uchar_t         conv_check_native(char **, char **);
764 extern  const char      *conv_lddstub(int);
765 extern  int             conv_strproc_isspace(int);
766 extern  char            *conv_strproc_trim(char *);
767 extern  Boolean         conv_strproc_extract_value(char *, size_t, int,
768                             const char **);
769 extern  int             conv_sys_eclass(void);
770 extern  int             conv_translate_c_esc(char **);

772 /*
773  * Generic core formatting and iteration functionality
774  */
775 extern  conv_iter_ret_t _conv_iter_ds(conv_iter_osabi_t, Half,
776                             const conv_ds_t **, conv_iter_cb_t, void *,
777                             const char *);
778 extern  conv_iter_ret_t _conv_iter_ds_msg(const conv_ds_msg_t *,
779                             conv_iter_cb_t, void *, const char *);
```



```
780 extern  conv_iter_ret_t _conv_iter_vd(const Val_desc *, conv_iter_cb_t,
781                             void *, const char *);
782 extern  conv_iter_ret_t _conv_iter_vd2(conv_iter_osabi_t, Half,
783                             const Val_desc2 *, conv_iter_cb_t, void *,
784                             const char *);
785 extern  int             conv_iter_strtol_init(const char *,
786                             conv_strtol_uvalue_t *);
787 extern  conv_iter_ret_t conv_iter_strtol(const char *, Conv_elfvalue_t, void *);
788 extern  const char      *_conv_map_ds(uchar_t, Half, Conv_elfvalue_t,
789                             const conv_ds_t **, Conv_fmt_flags_t,
790                             Conv_inv_buf_t *, const char *);


793 /*
794  * Generic formatting interfaces.
795  */
796 extern  const char      *conv_bnd_obj(uint_t, Conv_bnd_obj_buf_t *);
797 extern  const char      *conv_bnd_type(uint_t, Conv_bnd_type_buf_t *);
798 extern  const char      *conv_config_feat(int, Conv_config_feat_buf_t *);
799 extern  const char      *conv_config_obj(ushort_t, Conv_config_obj_buf_t *);
800 extern  const char      *conv_config_upm(const char *, const char *,
801                             const char *, size_t);
802 extern  const char      *conv_cnote_auxv_af(Word, Conv_fmt_flags_t,
803                             Conv_cnote_auxv_af_buf_t *);
804 extern  const char      *conv_cnote_auxv_type(Word, Conv_fmt_flags_t,
805                             Conv_inv_buf_t *);
806 extern  const char      *conv_cnote_cc_content(Lword, Conv_fmt_flags_t,
807                             Conv_cnote_cc_content_buf_t *);
808 extern  const char      *conv_cnote_errno(int, Conv_fmt_flags_t,
809                             Conv_inv_buf_t *);
810 extern  const char      *conv_cnote_fault(Word, Conv_fmt_flags_t,
811                             Conv_inv_buf_t *);
812 extern  const char      *conv_cnote_fltset(uint32_t *, int,
813                             Conv_fmt_flags_t, Conv_cnote_fltset_buf_t *);
814 extern  const char      *conv_cnote_old_pr_flags(int, Conv_fmt_flags_t,
815                             Conv_cnote_old_pr_flags_buf_t *);
816 extern  const char      *conv_cnote_pr_dmodel(Word, Conv_fmt_flags_t,
817                             Conv_inv_buf_t *);
818 extern  const char      *conv_cnote_pr_flags(int, Conv_fmt_flags_t,
819                             Conv_cnote_pr_flags_buf_t *);
820 extern  const char      *conv_cnote_proc_flag(int, Conv_fmt_flags_t,
821                             Conv_cnote_proc_flag_buf_t *);
822 extern  const char      *conv_cnote_pr_regname(Half, int, Conv_fmt_flags_t,
823                             Conv_inv_buf_t *inv_buf);
824 extern  const char      *conv_cnote_pr_stype(Word, Conv_fmt_flags_t,
825                             Conv_inv_buf_t *);
826 extern  const char      *conv_cnote_pr_what(short, short, Conv_fmt_flags_t,
827                             Conv_inv_buf_t *);
828 extern  const char      *conv_cnote_pr_why(short, Conv_fmt_flags_t,
829                             Conv_inv_buf_t *);
830 extern  const char      *conv_cnote_priv(int, Conv_fmt_flags_t,
831                             Conv_inv_buf_t *);
832 extern  const char      *conv_psecflags(int, Conv_fmt_flags_t,
833                             Conv_secflags_buf_t *);
834 #endif /* ! codereview */
835 extern  const char      *conv_cnote_psetid(int, Conv_fmt_flags_t,
836                             Conv_inv_buf_t *);
837 extern  const char      *conv_cnote_sa_flags(int, Conv_fmt_flags_t,
838                             Conv_cnote_sa_flags_buf_t *);
839 extern  const char      *conv_cnote_signal(Word, Conv_fmt_flags_t,
840                             Conv_inv_buf_t *);
841 extern  const char      *conv_cnote_si_code(Half, int, int, Conv_fmt_flags_t,
842                             Conv_inv_buf_t *);
843 extern  const char      *conv_cnote_sigset(uint32_t *, int,
844                             Conv_fmt_flags_t, Conv_cnote_sigset_buf_t *);
845 extern  const char      *conv_cnote_ss_flags(int, Conv_fmt_flags_t,
```

```
                                Conv_cnote_ss_flags_buf_t *);
extern  const char      *conv_cnote_syscall(Word, Conv_fmt_flags_t,
                                Conv_inv_buf_t *);
extern  const char      *conv_cnote_sysset(uint32_t *, int,
                                Conv_fmt_flags_t, Conv_cnote_sysset_buf_t *);
extern  const char      *conv_cnote_fileflags(uint32_t, Conv_fmt_flags_t,
                                char *, size_t);
extern  const char      *conv_cnote_filemode(uint32_t, Conv_fmt_flags_t,
                                char *, size_t);
extern  const char      *conv_cnote_type(Word, Conv_fmt_flags_t,
                                Conv_inv_buf_t *);
extern  const char      *conv_def_tag(Symref, Conv_inv_buf_t *);
extern  const char      *conv_demangle_name(const char *);
extern  const char      *conv_dl_flag(int, Conv_fmt_flags_t,
                                Conv_dl_flag_buf_t *);
extern  const char      *conv_dl_info(int);
extern  const char      *conv_dl_mode(int, int, Conv_dl_mode_buf_t *);
extern  const char      *conv_dwarf_cfa(uchar_t, Conv_fmt_flags_t,
                                Conv_inv_buf_t *);
extern  const char      *conv_dwarf_ehe(uint_t, Conv_dwarf_ehe_buf_t *);
extern  const char      *conv_dwarf_regname(Half, Word, Conv_fmt_flags_t,
                                int *, Conv_inv_buf_t *);
extern  const char      *conv_ehdr_abivers(uchar_t, Word, Conv_fmt_flags_t,
                                Conv_inv_buf_t *);
extern  const char      *conv_ehdr_class(uchar_t, Conv_fmt_flags_t,
                                Conv_inv_buf_t *);
extern  const char      *conv_ehdr_data(uchar_t, Conv_fmt_flags_t,
                                Conv_inv_buf_t *);
extern  const char      *conv_ehdr_flags(Half, Word, Conv_fmt_flags_t,
                                Conv_ehdr_flags_buf_t *);
extern  const char      *conv_ehdr_mach(Half, Conv_fmt_flags_t,
                                Conv_inv_buf_t *);
extern  const char      *conv_ehdr_osabi(uchar_t, Conv_fmt_flags_t,
                                Conv_inv_buf_t *);
extern  const char      *conv_ehdr_type(uchar_t, Half, Conv_fmt_flags_t,
                                Conv_inv_buf_t *);
extern  const char      *conv_ehdr_vers(Word, Conv_fmt_flags_t,
                                Conv_inv_buf_t *);
extern  const char      *conv_elfdata_type(Elf_Type, Conv_inv_buf_t *);
extern  const char      *conv_ent_flags(ec_flags_t, Conv_ent_flags_buf_t *);
extern  const char      *conv_ent_files_flags(Word,  Conv_fmt_flags_t fmt_flags,
                                Conv_ent_files_flags_buf_t *);
extern  const char      *conv_la_activity(uint_t, Conv_fmt_flags_t,
                                Conv_inv_buf_t *);
extern  const char      *conv_la_bind(uint_t, Conv_la_bind_buf_t *);
extern  const char      *conv_la_search(uint_t, Conv_la_search_buf_t *);
extern  const char      *conv_la_symbind(uint_t, Conv_la_symbind_buf_t *);
extern  const char      *conv_grphdl_flags(uint_t, Conv_grphdl_flags_buf_t *);
extern  const char      *conv_grpdesc_flags(uint_t, Conv_grpdesc_flags_buf_t *);
extern  Isa_desc        *conv_isalist(void);
extern  const char      *conv_mapfile_version(Word, Conv_fmt_flags_t,
                                Conv_inv_buf_t *);
extern  const char      *conv_phdr_flags(uchar_t, Word, Conv_fmt_flags_t,
                                Conv_phdr_flags_buf_t *);
extern  const char      *conv_phdr_type(uchar_t, Half, Word, Conv_fmt_flags_t,
                                Conv_inv_buf_t *);
extern  const char      *conv_reject_desc(Rej_desc *, Conv_reject_desc_buf_t *,
                                Half mach);
extern  const char      *conv_reloc_type(Half, Word, Conv_fmt_flags_t,
                                Conv_inv_buf_t *);
extern  const char      *conv_reloc_type_static(Half, Word, Conv_fmt_flags_t);
extern  const char      *conv_reloc_386_type(Word, Conv_fmt_flags_t,
                                Conv_inv_buf_t *);
extern  const char      *conv_reloc_amd64_type(Word, Conv_fmt_flags_t,
                                Conv_inv_buf_t *);
extern  const char      *conv_reloc_SPARC_type(Word, Conv_fmt_flags_t,
```



```
                                Conv_inv_buf_t *);
extern  const char      *conv_sec_type(uchar_t, Half, Word, Conv_fmt_flags_t,
                                Conv_inv_buf_t *);
extern  const char      *conv_seg_flags(sg_flags_t, Conv_seg_flags_buf_t *);
extern  void            conv_str_to_c_literal(const char *buf, size_t n,
                                Conv_str_to_c_literal_func_t *cb_func,
                                void *uvalue);
extern  const char      *conv_sym_info_bind(uchar_t, Conv_fmt_flags_t,
                                Conv_inv_buf_t *);
extern  const char      *conv_sym_info_type(Half, uchar_t, Conv_fmt_flags_t,
                                Conv_inv_buf_t *);
extern  const char      *conv_sym_shndx(uchar_t, Half, Half, Conv_fmt_flags_t,
                                Conv_inv_buf_t *);
extern  const char      *conv_sym_other(uchar_t, Conv_inv_buf_t *);
extern  const char      *conv_sym_other_vis(uchar_t, Conv_fmt_flags_t,
                                Conv_inv_buf_t *);
extern  const char      *conv_syminfo_boundto(Half, Conv_fmt_flags_t,
                                Conv_inv_buf_t *);
extern  const char      *conv_syminfo_flags(Half, Conv_fmt_flags_t,
                                Conv_syminfo_flags_buf_t *);
extern  const char      *conv_time(struct timeval *, struct timeval *,
                                Conv_time_buf_t *);
extern  Uts_desc        *conv_uts(void);
extern  const char      *conv_ver_flags(Half, Conv_fmt_flags_t,
                                Conv_ver_flags_buf_t *);
extern  const char      *conv_ver_index(Versym, int, Conv_inv_buf_t *);


/*
 * Generic iteration interfaces.
 */
extern  conv_iter_ret_t conv_iter_cap_tags(Conv_fmt_flags_t, conv_iter_cb_t,
                                void *);
extern  conv_iter_ret_t conv_iter_cap_val_hw1(Half, Conv_fmt_flags_t,
                                conv_iter_cb_t, void *);
extern  conv_iter_ret_t conv_iter_cap_val_hw2(Half, Conv_fmt_flags_t,
                                conv_iter_cb_t, void *);
extern  conv_iter_ret_t conv_iter_cap_val_sf1(Conv_fmt_flags_t, conv_iter_cb_t,
                                void *);

extern  conv_iter_ret_t conv_iter_dyn_feature1(Conv_fmt_flags_t, conv_iter_cb_t,
                                void *);
extern  conv_iter_ret_t conv_iter_dyn_flag(Conv_fmt_flags_t, conv_iter_cb_t,
                                void *);
extern  conv_iter_ret_t conv_iter_dyn_flag1(Conv_fmt_flags_t, conv_iter_cb_t,
                                void *);
extern  conv_iter_ret_t conv_iter_dyn_posflag1(Conv_fmt_flags_t, conv_iter_cb_t,
                                void *);
extern  conv_iter_ret_t conv_iter_dyn_tag(conv_iter_osabi_t, Half,
                                Conv_fmt_flags_t, conv_iter_cb_t, void *);

extern  conv_iter_ret_t conv_iter_ehdr_abivers(conv_iter_osabi_t,
                                Conv_fmt_flags_t, conv_iter_cb_t, void *);
extern  conv_iter_ret_t conv_iter_ehdr_class(Conv_fmt_flags_t, conv_iter_cb_t,
                                void *);
extern  conv_iter_ret_t conv_iter_ehdr_data(Conv_fmt_flags_t, conv_iter_cb_t,
                                void *);
extern  conv_iter_ret_t conv_iter_ehdr_eident(Conv_fmt_flags_t, conv_iter_cb_t,
                                void *);
extern  conv_iter_ret_t conv_iter_ehdr_flags(Half, Conv_fmt_flags_t,
                                conv_iter_cb_t, void *);
extern  conv_iter_ret_t conv_iter_ehdr_mach(Conv_fmt_flags_t, conv_iter_cb_t,
                                void *);
extern  conv_iter_ret_t conv_iter_ehdr_osabi(Conv_fmt_flags_t, conv_iter_cb_t,
                                void *);
extern  conv_iter_ret_t conv_iter_ehdr_type(conv_iter_osabi_t, Conv_fmt_flags_t,
```

```
 978                             conv_iter_cb_t, void *);
 979 extern  conv_iter_ret_t conv_iter_ehdr_vers(Conv_fmt_flags_t, conv_iter_cb_t,
 980                             void *);

 982 extern  conv_iter_ret_t conv_iter_phdr_flags(conv_iter_osabi_t,
 983                             Conv_fmt_flags_t, conv_iter_cb_t, void *);
 984 extern  conv_iter_ret_t conv_iter_phdr_type(conv_iter_osabi_t, Conv_fmt_flags_t,
 985                             conv_iter_cb_t, void *);

 987 extern  conv_iter_ret_t conv_iter_sec_flags(conv_iter_osabi_t, Half,
 988                             Conv_fmt_flags_t, conv_iter_cb_t, void *);
 989 extern  conv_iter_ret_t conv_iter_sec_symtab(conv_iter_osabi_t,
 990                             Conv_fmt_flags_t, conv_iter_cb_t, void *);
 991 extern  conv_iter_ret_t conv_iter_sec_type(conv_iter_osabi_t, Half,
 992                             Conv_fmt_flags_t, conv_iter_cb_t, void *);

 994 extern  conv_iter_ret_t conv_iter_sym_info_bind(Conv_fmt_flags_t,
 995                             conv_iter_cb_t, void *);
 996 extern  conv_iter_ret_t conv_iter_sym_other_vis(Conv_fmt_flags_t,
 997                             conv_iter_cb_t, void *);
 998 extern  conv_iter_ret_t conv_iter_sym_shndx(conv_iter_osabi_t, Half,
 999                             Conv_fmt_flags_t, conv_iter_cb_t, void *);
1000 extern  conv_iter_ret_t conv_iter_sym_info_type(Half, Conv_fmt_flags_t,
1001                             conv_iter_cb_t, void *);

1003 extern  conv_iter_ret_t conv_iter_syminfo_boundto(Conv_fmt_flags_t,
1004                             conv_iter_cb_t, void *);
1005 extern  conv_iter_ret_t conv_iter_syminfo_flags(Conv_fmt_flags_t,
1006                             conv_iter_cb_t, void *);

1008 /*
1009  * Define all class specific routines.
1010  */
1011 #if     defined(_ELF64)
1012 #define conv_cap_tag            conv64_cap_tag
1013 #define conv_cap_val            conv64_cap_val
1014 #define conv_cap_val_hw1        conv64_cap_val_hw1
1015 #define conv_cap_val_hw2        conv64_cap_val_hw2
1016 #define conv_cap_val_sf1        conv64_cap_val_sf1
1017 #define conv_dyn_feature1       conv64_dyn_feature1
1018 #define conv_dyn_flag1          conv64_dyn_flag1
1019 #define conv_dyn_flag           conv64_dyn_flag
1020 #define conv_dyn_posflag1       conv64_dyn_posflag1
1021 #define conv_dyn_tag            conv64_dyn_tag
1022 #define _conv_expn_field        _conv64_expn_field
1023 #define _conv_expn_field2       _conv64_expn_field2
1024 #define conv_invalid_val        conv64_invalid_val
1025 #define conv_sec_flags          conv64_sec_flags
1026 #define conv_sec_linkinfo       conv64_sec_linkinfo
1027 #define conv_sym_value          conv64_sym_value
1028 #define conv_sym_SPARC_value    conv64_sym_SPARC_value
1029 #else
1030 #define conv_cap_tag            conv32_cap_tag
1031 #define conv_cap_val            conv32_cap_val
1032 #define conv_cap_val_hw1        conv32_cap_val_hw1
1033 #define conv_cap_val_hw2        conv32_cap_val_hw2
1034 #define conv_cap_val_sf1        conv32_cap_val_sf1
1035 #define conv_dyn_feature1       conv32_dyn_feature1
1036 #define conv_dyn_flag1          conv32_dyn_flag1
1037 #define conv_dyn_flag           conv32_dyn_flag
1038 #define conv_dyn_posflag1       conv32_dyn_posflag1
1039 #define conv_dyn_tag            conv32_dyn_tag
1040 #define _conv_expn_field        _conv32_expn_field
1041 #define _conv_expn_field2       _conv32_expn_field2
1042 #define conv_invalid_val        conv32_invalid_val
1043 #define conv_sec_flags          conv32_sec_flags
```



```
1044 #define conv_sec_linkinfo       conv32_sec_linkinfo
1045 #define conv_sym_value          conv32_sym_value
1046 #define conv_sym_SPARC_value    conv32_sym_SPARC_value
1047 #endif

1049 /*
1050  * ELFCLASS-specific core formatting functionality
1051  */
1052 extern  int             _conv_expn_field(CONV_EXPN_FIELD_ARG *,
1053                             const Val_desc *, Conv_fmt_flags_t, const char *);
1054 extern  int             _conv_expn_field2(CONV_EXPN_FIELD_ARG *, uchar_t,
1055                             Half, const Val_desc2 *, Conv_fmt_flags_t,
1056                             const char *);
1057 extern  const char      *conv_invalid_val(Conv_inv_buf_t *, Xword,
1058                             Conv_fmt_flags_t);

1060 /*
1061  * ELFCLASS-specific formatting interfaces.
1062  */
1063 extern  const char      *conv_cap_tag(Xword, Conv_fmt_flags_t,
1064                             Conv_inv_buf_t *);
1065 extern  const char      *conv_cap_val(Xword, Xword, Half, Conv_fmt_flags_t,
1066                             Conv_cap_val_buf_t *);
1067 extern  const char      *conv_cap_val_hw1(Xword, Half, Conv_fmt_flags_t,
1068                             Conv_cap_val_hw1_buf_t *);
1069 extern  const char      *conv_cap_val_hw2(Xword, Half, Conv_fmt_flags_t,
1070                             Conv_cap_val_hw2_buf_t *);
1071 extern  const char      *conv_cap_val_sf1(Xword, Half, Conv_fmt_flags_t,
1072                             Conv_cap_val_sf1_buf_t *);
1073 extern  const char      *conv_dyn_flag1(Xword, Conv_fmt_flags_t,
1074                             Conv_dyn_flag1_buf_t *);
1075 extern  const char      *conv_dyn_flag(Xword, Conv_fmt_flags_t,
1076                             Conv_dyn_flag_buf_t *);
1077 extern  const char      *conv_dyn_posflag1(Xword, Conv_fmt_flags_t,
1078                             Conv_dyn_posflag1_buf_t *);
1079 extern  const char      *conv_dyn_tag(Xword, uchar_t, Half, Conv_fmt_flags_t,
1080                             Conv_inv_buf_t *);
1081 extern  const char      *conv_dyn_feature1(Xword, Conv_fmt_flags_t,
1082                             Conv_dyn_feature1_buf_t *);
1083 extern  const char      *conv_sec_flags(uchar_t osabi, Half mach, Xword,
1084                             Conv_fmt_flags_t, Conv_sec_flags_buf_t *);
1085 extern  const char      *conv_sec_linkinfo(Word, Xword, Conv_inv_buf_t *);
1086 extern  const char      *conv_sym_value(Half, uchar_t, Addr, Conv_inv_buf_t *);
1087 extern  const char      *conv_sym_SPARC_value(Addr, Conv_fmt_flags_t,
1088                             Conv_inv_buf_t *);

1090 /*
1091  * Define macros for _conv_XXX() routines that accept local_sgs_msg as the
1092  * final argument. The macros hide that argument from the caller's view and
1093  * supply the SGS message array for the file from which the macro is used
1094  * in its place. This trick is used to allow these functions to access the
1095  * message strings from any source file they are called from.
1096  */
1097 #define conv_expn_field(_arg, _vdp, _fmt_flags) \
1098     _conv_expn_field(_arg, _vdp, _fmt_flags, MSG_SGS_LOCAL_ARRAY)

1100 #define conv_expn_field2(_arg, _osabi, _mach, _vdp, _fmt_flags) \
1101     _conv_expn_field2(_arg, _osabi, _mach, _vdp, _fmt_flags, \
1102     MSG_SGS_LOCAL_ARRAY)

1104 #define conv_iter_ds(_osabi, _mach, _dsp, _func, _uvalue) \
1105     _conv_iter_ds(_osabi, _mach, _dsp, _func, _uvalue, MSG_SGS_LOCAL_ARRAY)

1107 #define conv_iter_vd(_vdp, _func, _uvalue)      \
1108     _conv_iter_vd(_vdp, _func, _uvalue, MSG_SGS_LOCAL_ARRAY)
```

```
1110 #define conv_iter_vd2(_osabi, _mach, _vdp, _func, _uvalue)              \
1111     _conv_iter_vd2(_osabi, _mach, _vdp, _func, _uvalue, MSG_SGS_LOCAL_ARRAY)

1113 #define conv_map_ds(_osabi, _mach, _value, _dsp, _fmt_flags, _inv_buf) \
1114     _conv_map_ds(_osabi, _mach, _value, _dsp, _fmt_flags, _inv_buf, \
1115     MSG_SGS_LOCAL_ARRAY)


1118 #ifdef  __cplusplus
1119 }
1120 #endif

1122 #endif /* _CONV_H */
```

```
**********************************************************
   66739 Wed May 27 19:49:01 2015
new/usr/src/cmd/sgs/include/libld.h
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
______unchanged_portion_omitted_


 241 /*
 242  * Output file processing structure
 243  */
 244 typedef Lword   ofl_flag_t;
 245 typedef Word    ofl_guideflag_t;
 246 struct ofl_desc {
 247         char            *ofl_sgsid;     /* link-editor identification */
 248         const char      *ofl_name;      /* full file name */
 249         Elf             *ofl_elf;       /* elf_memory() elf descriptor */
 250         Elf             *ofl_welf;      /* ELF_C_WRITE elf descriptor */
 251         Ehdr            *ofl_dehdr;     /* default elf header, and new elf */
 252         Ehdr            *ofl_nehdr;     /*      header describing this file */
 253         Phdr            *ofl_phdr;      /* program header descriptor */
 254         Phdr            *ofl_tlsphdr;   /* TLS phdr */
 255         int             ofl_fd;         /* file descriptor */
 256         size_t          ofl_size;       /* image size */
 257         APlist          *ofl_maps;      /* list of input mapfiles */
 258         APlist          *ofl_segs;      /* list of segments */
 259         APlist          *ofl_segs_order; /* SEGMENT_ORDER segments */
 260         avl_tree_t      ofl_segs_avl;   /* O(log N) access to named segments */
 261         APlist          *ofl_ents;      /* list of entrance descriptors */
 262         avl_tree_t      ofl_ents_avl;   /* O(log N) access to named ent. desc */
 263         APlist          *ofl_objs;      /* relocatable object file list */
 264         Word            ofl_objscnt;    /*      and count */
 265         APlist          *ofl_ars;       /* archive library list */
 266         Word            ofl_arscnt;     /*      and count */
 267         int             ofl_ars_gsandx; /* archive group argv index. 0 means */
 268                                         /*      no current group, < 0 means */
 269                                         /*      error reported. >0 is cur ndx */
 270         Word            ofl_ars_gsndx;  /* current -zrescan-start ofl_ars ndx */
 271         APlist          *ofl_sos;       /* shared object list */
 272         Word            ofl_soscnt;     /*      and count */
 273         APlist          *ofl_soneed;    /* list of implicitly required .so's */
 274         APlist          *ofl_socntl;    /* list of .so control definitions */
 275         Rel_cache       ofl_outrels;    /* list of output relocations */
 276         Rel_cache       ofl_actrels;    /* list of relocations to perform */
 277         APlist          *ofl_relaux;    /* Rel_aux cache for outrels/actrels */
 278         Word            ofl_entrelscnt; /* no of relocations entered */
 279         Alist           *ofl_copyrels;  /* list of copy relocations */
 280         APlist          *ofl_ordered;   /* list of shf_ordered sections */
 281         APlist          *ofl_symdtent;  /* list of syminfo symbols that need */
 282                                         /*      to reference .dynamic entries */
 283         APlist          *ofl_ismove;    /* list of .SUNW_move sections */
 284         APlist          *ofl_ismoverel; /* list of relocation input section */
 285                                         /* targeting to expanded area */
 286         APlist          *ofl_parsyms;   /* list of partially initialized */
 287                                         /*      symbols (ie. move symbols) */
 288         APlist          *ofl_extrarels; /* relocation sections which have */
 289                                         /*    a NULL sh_info */
 290         avl_tree_t      *ofl_groups;    /* pointer to head of Groups AVL tree */
 291         APlist          *ofl_initarray; /* list of init array func names */
 292         APlist          *ofl_finiarray; /* list of fini array func names */
 293         APlist          *ofl_preiarray; /* list of preinit array func names */
 294         APlist          *ofl_rtldinfo;  /* list of rtldinfo syms */
 295         APlist          *ofl_osgroups;  /* list of output GROUP sections */
```



```
 296         APlist          *ofl_ostlsseg;  /* pointer to sections in TLS segment */
 297         APlist          *ofl_unwind;    /* list of unwind output sections */
 298         Os_desc         *ofl_unwindhdr; /* Unwind hdr */
 299         avl_tree_t      ofl_symavl;     /* pointer to head of Syms AVL tree */
 300         Sym_desc        **ofl_regsyms;  /* array of potential register */
 301         Word            ofl_regsymsno;  /*    symbols and array count */
 302         Word            ofl_regsymcnt;  /* no. of output register symbols */
 303         Word            ofl_lregsymcnt; /* no. of local register symbols */
 304         Sym_desc        *ofl_dtracesym; /* ld -zdtrace= */
 305         ofl_flag_t      ofl_flags;      /* various state bits, args etc. */
 306         ofl_flag_t      ofl_flags1;     /*      more flags */
 307         void            *ofl_entry;     /* entry point (-e and Sym_desc *) */
 308         char            *ofl_filtees;   /* shared objects we are a filter for */
 309         const char      *ofl_soname;    /* (-h option) output file name for */
 310                                         /*      dynamic structure */
 311         const char      *ofl_interp;    /* interpreter name used by exec() */
 312         char            *ofl_rpath;     /* run path to store in .dynamic */
 313         char            *ofl_config;    /* config path to store in .dynamic */
 314         APlist          *ofl_ulibdirs;  /* user supplied library search list */
 315         APlist          *ofl_dlibdirs;  /* default library search list */
 316         Word            ofl_vercnt;     /* number of versions to generate */
 317         APlist          *ofl_verdesc;   /* list of version descriptors */
 318         size_t          ofl_verdefsz;   /* size of version definition section */
 319         size_t          ofl_verneedsz;  /* size of version needed section */
 320         Word            ofl_entercnt;   /* no. of global symbols entered */
 321         Word            ofl_globcnt;    /* no. of global symbols to output */
 322         Word            ofl_scopecnt;   /* no. of scoped symbols to output */
 323         Word            ofl_dynscopecnt; /* no. scoped syms in .SUNW_ldynsym */
 324         Word            ofl_elimcnt;    /* no. of eliminated symbols */
 325         Word            ofl_locscnt;    /* no. of local symbols in .symtab */
 326         Word            ofl_dynlocscnt; /* no. local symbols in .SUNW_ldynsym */
 327         Word            ofl_dynsymsortcnt; /* no. ndx in .SUNW_dynsymsort */
 328         Word            ofl_dyntlssortcnt; /* no. ndx in .SUNW_dyntlssort */
 329         Word            ofl_dynshdrcnt; /* no. of output section in .dynsym */
 330         Word            ofl_shdrcnt;    /* no. of output sections */
 331         Word            ofl_caploclcnt; /* no. of local capabilities symbols */
 332         Word            ofl_capsymcnt;  /* no. of symbol capabilities entries */
 333                                         /*      required */
 334         Word            ofl_capchaincnt; /* no. of Capchain symbols */
 335         APlist          *ofl_capgroups; /* list of capabilities groups */
 336         avl_tree_t      *ofl_capfamilies; /* capability family AVL tree */
 337         Str_tbl         *ofl_shdrsttab; /* Str_tbl for shdr strtab */
 338         Str_tbl         *ofl_strtab;    /* Str_tbl for symtab strtab */
 339         Str_tbl         *ofl_dynstrtab; /* Str_tbl for dymsym strtab */
 340         Gotndx          *ofl_tlsldgotndx; /* index to LD TLS_index structure */
 341         Xword           ofl_relocsz;    /* size of output relocations */
 342         Xword           ofl_relocgotsz; /* size of .got relocations */
 343         Xword           ofl_relocpltsz; /* size of .plt relocations */
 344         Xword           ofl_relocbsssz; /* size of .bss (copy) relocations */
 345         Xword           ofl_relocrelsz; /* size of .rel[a] relocations */
 346         Word            ofl_relocincnt; /* no. of input relocations */
 347         Word            ofl_relocent;   /* tot number of output relocations */
 348         Word            ofl_relocentsub; /* tot numb of output relocations to */
 349                                         /*      skip (-zignore) */
 350         Word            ofl_relocrelcnt; /* tot number of relative */
 351                                         /*      relocations */
 352         Word            ofl_gotcnt;     /* no. of .got entries */
 353         Word            ofl_pltcnt;     /* no. of .plt entries */
 354         Word            ofl_pltpad;     /* no. of .plt padd entries */
 355         Word            ofl_hashbkts;   /* no. of hash buckets required */
 356         Is_desc         *ofl_isbss;     /* .bss input section (globals) */
 357         Is_desc         *ofl_islbss;    /* .lbss input section (globals) */
 358         Is_desc         *ofl_istlsbss;  /* .tlsbss input section (globals) */
 359         Is_desc         *ofl_isparexpn; /* -z nopartial .data input section */
 360         Os_desc         *ofl_osdynamic; /* .dynamic output section */
 361         Os_desc         *ofl_osdynsym;  /* .dynsym output section */
```

```
362         Os_desc         *ofl_osldynsym; /* .SUNW_ldynsym output section */
363         Os_desc         *ofl_osdynstr;  /* .dynstr output section */
364         Os_desc         *ofl_osdynsymsort; /* .SUNW_dynsymsort output section */
365         Os_desc         *ofl_osdyntlssort; /* .SUNW_dyntlssort output section */
366         Os_desc         *ofl_osgot;     /* .got output section */
367         Os_desc         *ofl_oshash;    /* .hash output section */
368         Os_desc         *ofl_osinitarray; /* .init_array output section */
369         Os_desc         *ofl_osfiniarray; /* .fini_array output section */
370         Os_desc         *ofl_ospreinitarray; /* .preinit_array output section */
371         Os_desc         *ofl_osinterp;  /* .interp output section */
372         Os_desc         *ofl_oscap;     /* .SUNW_cap output section */
373         Os_desc         *ofl_oscapinfo; /* .SUNW_capinfo output section */
374         Os_desc         *ofl_oscapchain; /* .SUNW_capchain output section */
375         Os_desc         *ofl_osplt;     /* .plt output section */
376         Os_desc         *ofl_osmove;    /* .SUNW_move output section */
377         Os_desc         *ofl_osrelhead; /* first relocation section */
378         Os_desc         *ofl_osrel;     /* .rel[a] relocation section */
379         Os_desc         *ofl_osshstrtab; /* .shstrtab output section */
380         Os_desc         *ofl_osstrtab;  /* .strtab output section */
381         Os_desc         *ofl_ossymtab;  /* .symtab output section */
382         Os_desc         *ofl_ossymshndx; /* .symtab_shndx output section */
383         Os_desc         *ofl_osdynshndx; /* .dynsym_shndx output section */
384         Os_desc         *ofl_osldynshndx; /* .SUNW_ldynsym_shndx output sec */
385         Os_desc         *ofl_osverdef;  /* .version definition output section */
386         Os_desc         *ofl_osverneed; /* .version needed output section */
387         Os_desc         *ofl_osversym;  /* .version symbol ndx output section */
388         Word            ofl_dtflags_1;  /* DT_FLAGS_1 entries */
389         Word            ofl_dtflags;    /* DT_FLAGS entries */
390         Os_desc         *ofl_ossyminfo; /* .SUNW_syminfo output section */
391         Half            ofl_parexpnndx; /* -z nopartial section index */
392                                         /* Ref. at perform_outreloc() in */
393                                         /* libld/{mach}/machrel.c */
394         Xword           *ofl_checksum;  /* DT_CHECKSUM value address */
395         char            *ofl_depaudit;  /* dependency auditing required (-P) */
396         char            *ofl_audit;     /* object auditing required (-p) */
397         Alist           *ofl_symfltrs;  /* per-symbol filtees and their */
398         Alist           *ofl_dtsfltrs;  /*      associated .dynamic/.dynstrs */
399         Objcapset       ofl_ocapset;    /* object capabilities */
400         Lm_list         *ofl_lml;       /* runtime link-map list */
401         Gottable        *ofl_gottable;  /* debugging got information */
402         Rlxrel_cache    ofl_sr_cache;   /* Cache last result from */
403                                         /*      sloppy_comdat_reloc() */
404         APlist          *ofl_maptext;   /* mapfile added text sections */
405         APlist          *ofl_mapdata;   /* mapfile added data sections */
406         avl_tree_t      *ofl_wrap;      /* -z wrap symbols */
407         ofl_guideflag_t ofl_guideflags; /* -z guide flags */
408         APlist          *ofl_assdeflib; /* -z assert-deflib exceptions */
409         int             ofl_aslr;       /* -z aslr, -1 is disable, 1 is enable *
410         /*
411          * XXX: I'd rather this be ofl_secflags, -z secflags, and
412          * DT_ILL_SECFLAGS, with ASLR and -z aslr for compat.  I think? maybe?
413          */
414 #endif /* ! codereview */
415 };

417 #define FLG_OF_DYNAMIC  0x00000001      /* generate dynamic output module */
418 #define FLG_OF_STATIC   0x00000002      /* generate static output module */
419 #define FLG_OF_EXEC     0x00000004      /* generate an executable */
420 #define FLG_OF_RELOBJ   0x00000008      /* generate a relocatable object */
421 #define FLG_OF_SHAROBJ  0x00000010      /* generate a shared object */
422 #define FLG_OF_BFLAG    0x00000020      /* do no special plt building: -b */
423 #define FLG_OF_IGNENV   0x00000040      /* ignore LD_LIBRARY_PATH: -i */
424 #define FLG_OF_STRIP    0x00000080      /* strip output: -s */
425 #define FLG_OF_NOWARN   0x00000100      /* disable symbol warnings: -t */
426 #define FLG_OF_NOUNDEF  0x00000200      /* allow no undefined symbols: -zdefs */
427 #define FLG_OF_PURETXT  0x00000400      /* allow no text relocations: -ztext */
```



```
428 #define FLG_OF_GENMAP   0x00000800      /* generate a memory map: -m */
429 #define FLG_OF_DYNLIBS  0x00001000      /* dynamic input allowed: -Bdynamic */
430 #define FLG_OF_SYMBOLIC 0x00002000      /* bind global symbols: -Bsymbolic */
431 #define FLG_OF_ADDVERS  0x00004000      /* add version stamp: -Qy */
432 #define FLG_OF_NOLDYNSYM 0x00008000     /* -znoldynsym set */
433 #define FLG_OF_IS_ORDER 0x00010000      /* input section ordering within a */
434                                         /*      segment is required */
435 #define FLG_OF_EC_FILES 0x00020000      /* Ent_desc exist w/non-NULL ec_files */
436 #define FLG_OF_TEXTREL  0x00040000      /* text relocations have been found */
437 #define FLG_OF_MULDEFS  0x00080000      /* multiple symbols are allowed */
438 #define FLG_OF_TLSPHDR  0x00100000      /* a TLS program header is required */
439 #define FLG_OF_BLDGOT   0x00200000      /* build GOT table */
440 #define FLG_OF_VERDEF   0x00400000      /* record version definitions */
441 #define FLG_OF_VERNEED  0x00800000      /* record version dependencies */
442 #define FLG_OF_NOVERSEC 0x01000000      /* don't record version sections */
443 #define FLG_OF_KEY      0x02000000      /* file requires sort keys */
444 #define FLG_OF_PROCRED  0x04000000      /* process any symbol reductions by */
445                                         /*      effecting the symbol table */
446                                         /*      output and relocations */
447 #define FLG_OF_SYMINFO  0x08000000      /* create a syminfo section */
448 #define FLG_OF_AUX      0x10000000      /* ofl_filter is an auxiliary filter */
449 #define FLG_OF_FATAL    0x20000000      /* fatal error during input */
450 #define FLG_OF_WARN     0x40000000      /* warning during input processing. */
451 #define FLG_OF_VERBOSE  0x80000000      /* -z verbose flag set */

453 #define FLG_OF_MAPSYMB  0x000100000000  /* symbolic scope definition seen */
454 #define FLG_OF_MAPGLOB  0x000200000000  /* global scope definition seen */
455 #define FLG_OF_COMREL   0x000400000000  /* -z combreloc set, which enables */
456                                         /*      DT_RELACNT tracking, */
457 #define FLG_OF_NOCOMREL 0x000800000000  /* -z nocombreloc set */
458 #define FLG_OF_AUTOLCL  0x001000000000  /* automatically reduce unspecified */
459                                         /*      global symbols to locals */
460 #define FLG_OF_AUTOELM  0x002000000000  /* automatically eliminate */
461                                         /*      unspecified global symbols */
462 #define FLG_OF_REDLSYM  0x004000000000  /* reduce local symbols */
463 #define FLG_OF_OS_ORDER 0x008000000000  /* output section ordering required */
464 #define FLG_OF_OSABI    0x010000000000  /* tag object as ELFOSABI_SOLARIS */
465 #define FLG_OF_ADJOSCNT 0x020000000000  /* adjust ofl_shdrcnt to accommodate */
466                                         /*      discarded sections */
467 #define FLG_OF_OTOSCAP  0x040000000000  /* convert object capabilities to */
468                                         /*      symbol capabilities */
469 #define FLG_OF_PTCAP    0x080000000000  /* PT_SUNWCAP required */
470 #define FLG_OF_CAPSTRS  0x100000000000  /* capability strings are required */
471 #define FLG_OF_EHFRAME  0x200000000000  /* output contains .eh_frame section */
472 #define FLG_OF_FATWARN  0x400000000000  /* make warnings fatal */
473 #define FLG_OF_ADEFLIB  0x800000000000  /* no libraries in default path */

475 /*
476  * In the flags1 arena, establish any options that are applicable to archive
477  * extraction first, and associate a mask.  These values are recorded with any
478  * archive descriptor so that they may be reset should the archive require a
479  * rescan to try and resolve undefined symbols.
480  */
481 #define FLG_OF1_ALLEXRT 0x0000000001    /* extract all members from an */
482                                         /*      archive file */
483 #define FLG_OF1_WEAKEXT 0x0000000002    /* allow archive extraction to */
484                                         /*      resolve weak references */
485 #define MSK_OF1_ARCHIVE 0x0000000003    /* archive flags mask */

487 #define FLG_OF1_NOINTRP 0x0000000008    /* -z nointerp flag set */
488 #define FLG_OF1_ZDIRECT 0x0000000010    /* -z direct flag set */
489 #define FLG_OF1_NDIRECT 0x0000000020    /* no-direct bindings specified */
490 #define FLG_OF1_DEFERRED 0x0000000040   /* deferred dependency recording */

492 #define FLG_OF1_RELDYN  0x0000000100    /* process .dynamic in rel obj */
493 #define FLG_OF1_NRLXREL 0x0000000200    /* -z norelaxreloc flag set */
```

```
#define FLG_OF1_RLXREL  0x0000000400    /* -z relaxreloc flag set */
#define FLG_OF1_IGNORE  0x0000000800    /* ignore unused dependencies */
#define FLG_OF1_NOSGHND 0x0000001000    /* -z nosighandler flag set */
#define FLG_OF1_TEXTOFF 0x0000002000    /* text relocations are ok */
#define FLG_OF1_ABSEXEC 0x0000004000    /* -zabsexec set */
#define FLG_OF1_LAZYLD  0x0000008000    /* lazy loading of objects enabled */
#define FLG_OF1_GRPPRM  0x0000010000    /* dependencies are to have */
                                        /*      GROUPPERM enabled */

#define FLG_OF1_NOPARTI 0x0000040000    /* -znopartial set */
#define FLG_OF1_BSSOREL 0x0000080000    /* output relocation against bss */
                                        /*      section */
#define FLG_OF1_TLSOREL 0x0000100000    /* output relocation against .tlsbss */
                                        /*      section */
#define FLG_OF1_MEMORY  0x0000200000    /* produce a memory model */
#define FLG_OF1_NGLBDIR 0x0000400000    /* no DT_1_DIRECT flag allowed */
#define FLG_OF1_ENCDIFF 0x0000800000    /* host running linker has different */
                                        /*      byte order than output object */
#define FLG_OF1_VADDR   0x0001000000    /* a segment defines explicit vaddr */
#define FLG_OF1_EXTRACT 0x0002000000    /* archive member has been extracted */
#define FLG_OF1_RESCAN  0x0004000000    /* any archives should be rescanned */
#define FLG_OF1_IGNPRC  0x0008000000    /* ignore processing required */
#define FLG_OF1_NCSTTAB 0x0010000000    /* -znocompstrtab set */
#define FLG_OF1_DONE    0x0020000000    /* link-editor processing complete */
#define FLG_OF1_NONREG  0x0040000000    /* non-regular file specified as */
                                        /*      the output file */
#define FLG_OF1_ALNODIR 0x0080000000    /* establish NODIRECT for all */
                                        /*      exported interfaces. */
#define FLG_OF1_OVHWCAP1 0x0100000000   /* override CA_SUNW_HW_1 capabilities */
#define FLG_OF1_OVSFCAP1 0x0200000000   /* override CA_SUNW_SF_1 capabilities */
#define FLG_OF1_OVHWCAP2 0x0400000000   /* override CA_SUNW_HW_2 capabilities */
#define FLG_OF1_OVMACHCAP 0x0800000000  /* override CA_SUNW_MACH capability */
#define FLG_OF1_OVPLATCAP 0x1000000000  /* override CA_SUNW_PLAT capability */
#define FLG_OF1_OVIDCAP 0x2000000000    /* override CA_SUNW_ID capability */

/*
 * Guidance flags. The flags with the FLG_OFG_NO_ prefix are used to suppress
 * messages for a given category, and use the lower 28 bits of the word,
 * The upper nibble is reserved for other guidance status.
 */
#define FLG_OFG_ENABLE          0x10000000      /* -z guidance option active */
#define FLG_OFG_ISSUED          0x20000000      /* -z guidance message issued */

#define FLG_OFG_NO_ALL          0x0fffffff      /* disable all guidance */
#define FLG_OFG_NO_DEFS         0x00000001      /* specify all dependencies */
#define FLG_OFG_NO_DB           0x00000002      /* use direct bindings */
#define FLG_OFG_NO_LAZY         0x00000004      /* be explicit about lazyload */
#define FLG_OFG_NO_MF           0x00000008      /* use v2 mapfile syntax */
#define FLG_OFG_NO_TEXT         0x00000010      /* verify pure text segment */
#define FLG_OFG_NO_UNUSED       0x00000020      /* remove unused dependency */

/*
 * Test to see if a guidance should be given for a given category
 * or not. _no_flag is one of the FLG_OFG_NO_xxx flags. Returns TRUE
 * if the guidance should be issued, and FALSE to remain silent.
 */
#define OFL_GUIDANCE(_ofl, _no_flag) (((_ofl)->ofl_guideflags & \
        (FLG_OFG_ENABLE | (_no_flag))) == FLG_OFG_ENABLE)

/*
 * Test to see if the output file would allow the presence of
 * a .dynsym section.
 */
#define OFL_ALLOW_DYNSYM(_ofl) (((_ofl)->ofl_flags & \
        (FLG_OF_DYNAMIC | FLG_OF_RELOBJ)) == FLG_OF_DYNAMIC)
```



```
/*
 * Test to see if the output file would allow the presence of
 * a .SUNW_ldynsym section. The requirements are that a .dynsym
 * is allowed, and -znoldynsym has not been specified. Note that
 * even if the answer is True (1), we will only generate one if there
 * are local symbols that require it.
 */
#define OFL_ALLOW_LDYNSYM(_ofl) (((_ofl)->ofl_flags & \
        (FLG_OF_DYNAMIC | FLG_OF_RELOBJ | FLG_OF_NOLDYNSYM)) == FLG_OF_DYNAMIC)

/*
 * Test to see if relocation processing should be done. This is normally
 * true, but can be disabled via the '-z noreloc' option. Note that
 * relocatable objects are still relocated even if '-z noreloc' is present.
 */
#define OFL_DO_RELOC(_ofl) (((_ofl)->ofl_flags & FLG_OF_RELOBJ) || \
        !((_ofl)->ofl_dtflags_1 & DF_1_NORELOC))

/*
 * Determine whether a static executable is being built.
 */
#define OFL_IS_STATIC_EXEC(_ofl) (((_ofl)->ofl_flags & \
        (FLG_OF_STATIC | FLG_OF_EXEC)) == (FLG_OF_STATIC | FLG_OF_EXEC))

/*
 * Determine whether a static object is being built.  This macro is used
 * to select the appropriate string table, and symbol table that other
 * sections need to reference.
 */
#define OFL_IS_STATIC_OBJ(_ofl) ((_ofl)->ofl_flags & \
        (FLG_OF_RELOBJ | FLG_OF_STATIC))

/*
 * Macros for counting symbol table entries.  These are used to size symbol
 * tables and associated sections (.syminfo, SUNW_capinfo, .hash, etc.) and
 * set required sh_info entries (the offset to the first global symbol).
 */
#define SYMTAB_LOC_CNT(_ofl)            /* local .symtab entries */     \
        (2 +                            /*    NULL and STT_FILE */      \
        (_ofl)->ofl_shdrcnt +           /*    section symbol */         \
        (_ofl)->ofl_caploclcnt +        /*    local capabilities */     \
        (_ofl)->ofl_scopecnt +          /*    scoped symbols */         \
        (_ofl)->ofl_locscnt)            /*    standard locals */
#define SYMTAB_ALL_CNT(_ofl)            /* all .symtab entries */       \
        (SYMTAB_LOC_CNT(_ofl) +         /*    .symtab locals */         \
        (_ofl)->ofl_globcnt)            /*    standard globals */

#define DYNSYM_LOC_CNT(_ofl)            /* local .dynsym entries */     \
        (1 +                            /*    NULL */                   \
        (_ofl)->ofl_dynshdrcnt +        /*    section symbols */        \
        (_ofl)->ofl_caploclcnt +        /*    local capabilities */     \
        (_ofl)->ofl_lregsymcnt)         /*    local register symbols */
#define DYNSYM_ALL_CNT(_ofl)            /* all .dynsym entries */       \
        (DYNSYM_LOC_CNT(_ofl) +         /*    .dynsym locals */         \
        (_ofl)->ofl_globcnt)            /*    standard globals */

/*
 * Define a move descriptor used within relocation structures.
 */
typedef struct {
        Move            *mr_move;
        Sym_desc        *mr_sym;
} Mv_reloc;

/*
 * Relocation (active & output) processing structure - transparent to common
```

```
 * code. There can be millions of these structures in a large link, so it
 * is important to keep it small. You should only add new items to Rel_desc
 * if they are critical, apply to most relocations, and cannot be easily
 * computed from the other information.
 *
 * Items that can be derived should be implemented as a function that accepts
 * a Rel_desc argument, and returns the desired data. ld_reloc_sym_name() is
 * an example of this.
 *
 * Lesser used relocation data is kept in an auxiliary block, Rel_aux,
 * that is only allocated as necessary. In exchange for adding one pointer
 * of overhead to Rel_desc (rel_aux), most relocations are reduced in size
 * by the size of Rel_aux. This strategy relies on the data in Rel_aux
 * being rarely needed --- otherwise it will backfire badly.
 *
 * Note that rel_raddend is primarily only of interest to RELA relocations,
 * and is set to 0 for REL. However, there is an exception: If FLG_REL_NADDEND
 * is set, then rel_raddend contains a replacement value for the implicit
 * addend found in the relocation target.
 *
 * Fields should be ordered from largest to smallest, to minimize packing
 * holes in the struct layout.
 */
struct rel_desc {
        Is_desc         *rel_isdesc;    /* input section reloc is against */
        Sym_desc        *rel_sym;       /* sym relocation is against */
        Rel_aux         *rel_aux;       /* NULL, or auxiliary data */
        Xword           rel_roffset;    /* relocation offset */
        Sxword          rel_raddend;    /* addend from input relocation */
        Word            rel_flags;      /* misc. flags for relocations */
        Word            rel_rtype;      /* relocation type */
};

/*
 * Data that would be kept in Rel_desc if the size of that structure was
 * not an issue. This auxiliary block is only allocated as needed,
 * and must only contain rarely needed items. The goal is for the vast
 * majority of Rel_desc structs to not have an auxiliary block.
 *
 * When a Rel_desc does not have an auxiliary block, a default value
 * is assumed for each auxiliary item:
 *
 * -    ra_osdesc:
 *      Output section to which relocation applies. The default
 *      value for this is the output section associated with the
 *      input section (rel_isdesc->is_osdesc), or NULL if there
 *      is no associated input section.
 *
 * -    ra_usym:
 *      If the symbol associated with a relocation is part of a weak/strong
 *      pair, then ra_usym contains the strong symbol and rel_sym the weak.
 *      Otherwise, the default value is the same value as rel_sym.
 *
 * -    ra_move:
 *      Move table data. The default value is NULL.
 *
 * -    ra_typedata:
 *      ELF_R_TYPE_DATA(info). This value applies only to a small
 *      subset of 64-bit sparc relocations, and is otherwise 0. The
 *      default value is 0.
 *
 * If any value in Rel_aux is non-default, then an auxiliary block is
 * necessary, and each field contains its actual value. If all the auxiliary
 * values are default, no Rel_aux is needed, and the RELAUX_GET_xxx()
 * macros below are able to supply the proper default.
 *
```



```
 * To set a Rel_aux value, use the ld_reloc_set_aux_XXX() functions.
 * These functions are written to avoid unnecessary auxiliary allocations,
 * and know the rules for each item.
 */
struct rel_aux {
        Os_desc         *ra_osdesc;     /* output section reloc is against */
        Sym_desc        *ra_usym;       /* strong sym if this is a weak pair */
        Mv_reloc        *ra_move;       /* move table information */
        Word            ra_typedata;    /* ELF_R_TYPE_DATA(info) */
};

/*
 * Test a given auxiliary value to determine if it has the default value
 * for that item, as described above. If all the auxiliary items have
 * their default values, no auxiliary place is necessary to represent them.
 * If any one of them is non-default, the auxiliary block is needed.
 */
#define RELAUX_ISDEFAULT_MOVE(_rdesc, _mv) (_mv == NULL)
#define RELAUX_ISDEFAULT_USYM(_rdesc, _usym) ((_rdesc)->rel_sym == _usym)
#define RELAUX_ISDEFAULT_OSDESC(_rdesc, _osdesc) \
        ((((_rdesc)->rel_isdesc == NULL) && (_osdesc == NULL)) || \
        ((_rdesc)->rel_isdesc && ((_rdesc)->rel_isdesc->is_osdesc == _osdesc)))
#define RELAUX_ISDEFAULT_TYPEDATA(_rdesc, _typedata) (_typedata == 0)

/*
 * Retrieve the value of an auxiliary relocation item, preserving the illusion
 * that every relocation descriptor has an auxiliary block attached. The
 * real implementation is that an auxiliary block is only present if one or
 * more auxiliary items have non-default values. These macros return the true
 * value if an auxiliary block is present, and the default value for the
 * item otherwise.
 */
#define RELAUX_GET_MOVE(_rdesc) \
        ((_rdesc)->rel_aux ? (_rdesc)->rel_aux->ra_move : NULL)
#define RELAUX_GET_USYM(_rdesc) \
        ((_rdesc)->rel_aux ? (_rdesc)->rel_aux->ra_usym : (_rdesc)->rel_sym)
#define RELAUX_GET_OSDESC(_rdesc) \
        ((_rdesc)->rel_aux ? (_rdesc)->rel_aux->ra_osdesc : \
        ((_rdesc)->rel_isdesc ? (_rdesc)->rel_isdesc->is_osdesc : NULL))
#define RELAUX_GET_TYPEDATA(_rdesc) \
        ((_rdesc)->rel_aux ? (_rdesc)->rel_aux->ra_typedata : 0)

/*
 * common flags used on the Rel_desc structure (defined in machrel.h).
 */
#define FLG_REL_GOT     0x00000001      /* relocation against GOT */
#define FLG_REL_PLT     0x00000002      /* relocation against PLT */
#define FLG_REL_BSS     0x00000004      /* relocation against BSS */
#define FLG_REL_LOAD    0x00000008      /* section loadable */
#define FLG_REL_SCNNDX  0x00000010      /* use section index for symbol ndx */
#define FLG_REL_CLVAL   0x00000020      /* clear VALUE for active relocation */
#define FLG_REL_ADVAL   0x00000040      /* add VALUE for output relocation, */
                                        /*      only relevant to SPARC and */
                                        /*      R_SPARC_RELATIVE */
#define FLG_REL_GOTCL   0x00000080      /* clear the GOT entry.  This is */
                                        /* relevant to RELA relocations, */
                                        /* not REL (i386) relocations */
#define FLG_REL_MOVETAB 0x00000100      /* Relocation against .SUNW_move */
                                        /*      adjustments required before */
                                        /*      actual relocation */
#define FLG_REL_NOINFO  0x00000200      /* Relocation comes from a section */
                                        /*      with a null sh_info field */
#define FLG_REL_REG     0x00000400      /* Relocation target is reg sym */
#define FLG_REL_FPTR    0x00000800      /* relocation against func. desc. */
#define FLG_REL_RFPTR1  0x00001000      /* Relative relocation against */
                                        /*   1st part of FD */
```

```
758 #define FLG_REL_RFPTR2  0x00002000      /* Relative relocation against */
759                                         /*   2nd part of FD */
760 #define FLG_REL_DISP    0x00004000      /* *disp* relocation */
761 #define FLG_REL_STLS    0x00008000      /* IE TLS reference to */
762                                         /*      static TLS GOT index */
763 #define FLG_REL_DTLS    0x00010000      /* GD TLS reference relative to */
764                                         /*      dynamic TLS GOT index */
765 #define FLG_REL_MTLS    0x00020000      /* LD TLS reference against GOT */
766 #define FLG_REL_STTLS   0x00040000      /* LE TLS reference directly */
767                                         /*      to static tls index */
768 #define FLG_REL_TLSFIX  0x00080000      /* relocation points to TLS instr. */
769                                         /*      which needs updating */
770 #define FLG_REL_RELA    0x00100000      /* descriptor captures a Rela */
771 #define FLG_REL_GOTFIX  0x00200000      /* relocation points to GOTOP instr. */
772                                         /*      which needs updating */
773 #define FLG_REL_NADDEND 0x00400000      /* Replace implicit addend in dest */
774                                         /*      with value in rel_raddend */
775                                         /*      Relevant to REL (i386) */
776                                         /*      relocations, not to RELA. */

778 /*
779  * We often need the name of the symbol contained in a relocation descriptor
780  * for diagnostic or error output. This is usually the symbol name, but
781  * we substitute a constructed name in some cases. Hence, the name is
782  * generated on the fly by a private function within libld. This is the
783  * prototype for that function.
784  */
785 typedef const char *(* rel_desc_sname_func_t)(Rel_desc *);

787 /*
788  * Header for a relocation descriptor cache buffer.
789  */
790 struct rel_cachebuf {
791         Rel_desc        *rc_end;
792         Rel_desc        *rc_free;
793         Rel_desc        rc_arr[1];
794 };

796 /*
797  * Header for a relocation auxiliary descriptor cache buffer.
798  */
799 struct rel_aux_cachebuf {
800         Rel_aux         *rac_end;
801         Rel_aux         *rac_free;
802         Rel_aux         rac_arr[1];
803 };

805 /*
806  * Convenience macro for traversing every relocation descriptor found within
807  * a given relocation cache, transparently handling the cache buffers and
808  * skipping any unallocated descriptors within the buffers.
809  *
810  * entry:
811  *      _rel_cache - Relocate descriptor cache (Rel_cache) to traverse
812  *      _idx - Aliste index variable for use by the macro
813  *      _rcbp - Cache buffer pointer, for use by the macro
814  *      _orsp - Rel_desc pointer, which will take on the value of a different
815  *              relocation descriptor in the cache in each iteration.
816  *
817  * The caller must not assign new values to _idx, _rcbp, or _orsp within
818  * the scope of REL_CACHE_TRAVERSE.
819  */
820 #define REL_CACHE_TRAVERSE(_rel_cache, _idx, _rcbp, _orsp) \
821         for (APLIST_TRAVERSE((_rel_cache)->rc_list, _idx, _rcbp)) \
822                 for (_orsp = _rcbp->rc_arr; _orsp < _rcbp->rc_free; _orsp++)
```



```
824 /*
825  * Symbol value descriptor.  For relocatable objects, each symbols value is
826  * its offset within its associated section.  Therefore, to uniquely define
827  * each symbol within a relocatable object, record and sort the sh_offset and
828  * symbol value.  This information is used to search for displacement
829  * relocations as part of copy relocation validation.
830  */
831 typedef struct {
832         Addr            ssv_value;
833         Sym_desc        *ssv_sdp;
834 } Ssv_desc;

836 /*
837  * Input file processing structures.
838  */
839 struct ifl_desc {                       /* input file descriptor */
840         const char      *ifl_name;      /* full file name */
841         const char      *ifl_soname;    /* shared object name */
842         dev_t           ifl_stdev;      /* device id and inode number for .so */
843         ino_t           ifl_stino;      /*      multiple inclusion checks */
844         Ehdr            *ifl_ehdr;      /* elf header describing this file */
845         Elf             *ifl_elf;       /* elf descriptor for this file */
846         Sym_desc        **ifl_oldndx;   /* original symbol table indices */
847         Sym_desc        *ifl_locs;      /* symbol desc version of locals */
848         Ssv_desc        *ifl_sortsyms;  /* sorted list of symbols by value */
849         Word            ifl_locscnt;    /* no. of local symbols to process */
850         Word            ifl_symscnt;    /* total no. of symbols to process */
851         Word            ifl_sortcnt;    /* no. of sorted symbols to process */
852         Word            ifl_shnum;      /* number of sections in file */
853         Word            ifl_shstrndx;   /* index to .shstrtab */
854         Word            ifl_vercnt;     /* number of versions in file */
855         Half            ifl_neededndx;  /* index to NEEDED in .dyn section */
856         Word            ifl_flags;      /* explicit/implicit reference */
857         Is_desc         **ifl_isdesc;   /* isdesc[scn ndx] = Is_desc ptr */
858         Sdf_desc        *ifl_sdfdesc;   /* control definition */
859         Versym          *ifl_versym;    /* version symbol table array */
860         Ver_index       *ifl_verndx;    /* verndx[ver ndx] = Ver_index */
861         APlist          *ifl_verdesc;   /* version descriptor list */
862         APlist          *ifl_relsect;   /* relocation section list */
863         Alist           *ifl_groups;    /* SHT_GROUP section list */
864         Cap_desc        *ifl_caps;      /* capabilities descriptor */
865 };

867 #define FLG_IF_CMDLINE  0x00000001      /* full filename specified from the */
868                                         /*      command line (no -l) */
869 #define FLG_IF_NEEDED   0x00000002      /* shared object should be recorded */
870 #define FLG_IF_DIRECT   0x00000004      /* establish direct bindings to this */
871                                         /*      object */
872 #define FLG_IF_EXTRACT  0x00000008      /* file extracted from an archive */
873 #define FLG_IF_VERNEED  0x00000010      /* version dependency information is */
874                                         /*      required */
875 #define FLG_IF_DEPREQD  0x00000020      /* dependency is required to satisfy */
876                                         /*      symbol references */
877 #define FLG_IF_NEEDSTR  0x00000040      /* dependency specified by -Nn */
878                                         /*      flag */
879 #define FLG_IF_IGNORE   0x00000080      /* ignore unused dependencies */
880 #define FLG_IF_NODIRECT 0x00000100      /* object contains symbols that */
881                                         /*      cannot be directly bound to */
882 #define FLG_IF_LAZYLD   0x00000200      /* dependency should be lazy loaded */
883 #define FLG_IF_GRPPRM   0x00000400      /* dependency establishes a group */
884 #define FLG_IF_DISPPEND 0x00000800      /* displacement relocation done */
885                                         /*      in the ld time. */
886 #define FLG_IF_DISPDONE 0x00001000      /* displacement relocation done */
887                                         /*      at the run time */
888 #define FLG_IF_MAPFILE  0x00002000      /* file is a mapfile */
889 #define FLG_IF_HSTRTAB  0x00004000      /* file has a string section */
```

```
890 #define FLG_IF_FILEREF  0x00008000      /* file contains a section which */
891                                         /*      is included in the output */
892                                         /*      allocatable image */
893 #define FLG_IF_GNUVER   0x00010000      /* file used GNU-style versioning */
894 #define FLG_IF_ORDERED  0x00020000      /* ordered section processing */
895                                         /*      required */
896 #define FLG_IF_OTOSCAP  0x00040000      /* convert object capabilities to */
897                                         /*      symbol capabilities */
898 #define FLG_IF_DEFERRED 0x00080000      /* dependency is deferred */
899 #define FLG_IF_RTLDINF  0x00100000      /* dependency has DT_SUNW_RTLTINF set */
900 #define FLG_IF_GROUPS   0x00200000      /* input file has groups to process */

902 /*
903  * Symbol states that require the generation of a DT_POSFLAG_1 .dynamic entry.
904  */
905 #define MSK_IF_POSFLAG1 (FLG_IF_LAZYLD | FLG_IF_GRPPRM | FLG_IF_DEFERRED)

907 /*
908  * Symbol states that require an associated Syminfo entry.
909  */
910 #define MSK_IF_SYMINFO  (FLG_IF_LAZYLD | FLG_IF_DIRECT | FLG_IF_DEFERRED)


913 struct is_desc {                        /* input section descriptor */
914         const char      *is_name;       /* original section name */
915         const char      *is_sym_name;   /* NULL, or name string to use for */
916                                         /*      related STT_SECTION symbols */
917         Shdr            *is_shdr;       /* the elf section header */
918         Ifl_desc        *is_file;       /* infile desc for this section */
919         Os_desc         *is_osdesc;     /* new output section for this */
920                                         /*      input section */
921         Elf_Data        *is_indata;     /* input sections raw data */
922         Is_desc         *is_symshndx;   /* related SHT_SYM_SHNDX section */
923         Is_desc         *is_comdatkeep; /* If COMDAT section is discarded, */
924                                         /*      this is section that was kept */
925         Word            is_scnndx;      /* original section index in file */
926         Word            is_ordndx;      /* index for section.  Used to decide */
927                                         /*      where to insert section when */
928                                         /*      reordering sections */
929         Word            is_keyident;    /* key for SHF_{ORDERED|LINK_ORDER} */
930                                         /*      processing and ident used for */
931                                         /*       placing/ordering sections */
932         Word            is_flags;       /* Various flags */
933 };

935 #define FLG_IS_ORDERED  0x0001          /* this is a SHF_ORDERED section */
936 #define FLG_IS_KEY      0x0002          /* section requires sort keys */
937 #define FLG_IS_DISCARD  0x0004          /* section is to be discarded */
938 #define FLG_IS_RELUPD   0x0008          /* symbol defined here may have moved */
939 #define FLG_IS_SECTREF  0x0010          /* section has been referenced */
940 #define FLG_IS_GDATADEF 0x0020          /* section contains global data sym */
941 #define FLG_IS_EXTERNAL 0x0040          /* isp from a user file */
942 #define FLG_IS_INSTRMRG 0x0080          /* Usable SHF_MERGE|SHF_STRINGS sec */
943 #define FLG_IS_GNSTRMRG 0x0100          /* Generated mergeable string section */

945 #define FLG_IS_PLACE    0x0400          /* section requires to be placed */
946 #define FLG_IS_COMDAT   0x0800          /* section is COMDAT */
947 #define FLG_IS_EHFRAME  0x1000          /* section is .eh_frame */

949 /*
950  * Output sections contain lists of input sections that are assigned to them.
951  * These items fall into 4 categories:
952  *      BEFORE - Ordered sections that specify SHN_BEFORE, in input order.
953  *      ORDERED - Ordered sections that are sorted using unsorted sections
954  *              as the sort key.
955  *      DEFAULT - Sections that are placed into the output section
```



```
956  *              in input order.
957  *      AFTER - Ordered sections that specify SHN_AFTER, in input order.
958  */
959 #define OS_ISD_BEFORE   0
960 #define OS_ISD_ORDERED  1
961 #define OS_ISD_DEFAULT  2
962 #define OS_ISD_AFTER    3
963 #define OS_ISD_NUM      4
964 typedef APlist *os_isdecs_arr[OS_ISD_NUM];

966 /*
967  * Convenience macro for traversing every input section associated
968  * with a given output section. The primary benefit of this macro
969  * is that it preserves a precious level of code indentation in the
970  * code that uses it.
971  */
972 #define OS_ISDESCS_TRAVERSE(_list_idx, _osp, _idx, _isp) \
973         for (_list_idx = 0; _list_idx < OS_ISD_NUM; _list_idx++) \
974                 for (APLIST_TRAVERSE(_osp->os_isdescs[_list_idx], _idx, _isp))


977 /*
978  * Map file and output file processing structures
979  */
980 struct os_desc {                        /* Output section descriptor */
981         const char      *os_name;       /* the section name */
982         Elf_Scn         *os_scn;        /* the elf section descriptor */
983         Shdr            *os_shdr;       /* the elf section header */
984         Os_desc         *os_relosdesc;  /* the output relocation section */
985         APlist          *os_relisdescs; /* reloc input section descriptors */
986                                         /*      for this output section */
987         os_isdecs_arr   os_isdescs;     /* lists of input sections in output */
988         APlist          *os_mstrisdescs; /* FLG_IS_INSTRMRG input sections */
989         Sg_desc         *os_sgdesc;     /* segment os_desc is placed on */
990         Elf_Data        *os_outdata;    /* output sections raw data */
991         avl_tree_t      *os_comdats;    /* AVL tree of COMDAT input sections */
992                                         /*      associated to output section */
993         Word            os_identndx;    /* section identifier for input */
994                                         /*      section processing, followed */
995                                         /*      by section symbol index */
996         Word            os_ordndx;      /* index for section.  Used to decide */
997                                         /*      where to insert section when */
998                                         /*      reordering sections */
999         Xword           os_szoutrels;   /* size of output relocation section */
1000        uint_t          os_namehash;    /* hash on section name */
1001        uchar_t         os_flags;       /* various flags */
1002 };

1004 #define FLG_OS_KEY              0x01    /* section requires sort keys */
1005 #define FLG_OS_OUTREL           0x02    /* output rel against this section */
1006 #define FLG_OS_SECTREF          0x04    /* isps are not affected by -zignore */
1007 #define FLG_OS_EHFRAME          0x08    /* section is .eh_frame */

1009 /*
1010  * The sg_id field of the segment descriptor is used to establish the default
1011  * order for program headers and segments in the output object. Segments are
1012  * ordered according to the following SGID values that classify them based on
1013  * their attributes. The initial set of built in segments are in this order,
1014  * and new mapfile defined segments are inserted into these groups. Within a
1015  * given SGID group, the position of new segments depends on the syntax
1016  * version of the mapfile that creates them. Version 1 (original sysv)
1017  * mapfiles place the new segment at the head of their group (reverse creation
1018  * order). The newer syntax places them at the end, following the others
1019  * (creation order).
1020  *
1021  * Note that any new segments must always be added after PT_PHDR and
```

```
 * PT_INTERP (refer Generic ABI, Page 5-4).
 */
#define SGID_PHDR       0       /* PT_PHDR */
#define SGID_INTERP     1       /* PT_INTERP */
#define SGID_SUNWCAP    2       /* PT_SUNWCAP */
#define SGID_TEXT       3       /* PT_LOAD */
#define SGID_DATA       4       /* PT_LOAD */
#define SGID_BSS        5       /* PT_LOAD */
#if     defined(_ELF64)
#define SGID_LRODATA    6       /* PT_LOAD (amd64-only) */
#define SGID_LDATA      7       /* PT_LOAD (amd64-only) */
#endif
#define SGID_TEXT_EMPTY 8       /* PT_LOAD, reserved (?E in version 1 syntax) */
#define SGID_NULL_EMPTY 9       /* PT_NULL, reserved (?E in version 1 syntax) */
#define SGID_DYN        10      /* PT_DYNAMIC */
#define SGID_DTRACE     11      /* PT_SUNWDTRACE */
#define SGID_TLS        12      /* PT_TLS */
#define SGID_UNWIND     13      /* PT_SUNW_UNWIND */
#define SGID_SUNWSTACK  14      /* PT_SUNWSTACK */
#define SGID_NOTE       15      /* PT_NOTE */
#define SGID_NULL       16      /* PT_NULL,  mapfile defined empty phdr slots */
                                /*      for use by post processors */
#define SGID_EXTRA      17      /* PT_NULL (final catchall) */

typedef Half sg_flags_t;
struct sg_desc {                        /* output segment descriptor */
        Word            sg_id;          /* segment identifier (for sorting) */
        Phdr            sg_phdr;        /* segment header for output file */
        const char      *sg_name;       /* segment name for PT_LOAD, PT_NOTE, */
                                        /*      and PT_NULL, otherwise NULL */
        Xword           sg_round;       /* data rounding required (mapfile) */
        Xword           sg_length;      /* maximum segment length; if 0 */
                                        /*      segment is not specified */
        APlist          *sg_osdescs;    /* list of output section descriptors */
        APlist          *sg_is_order;   /* list of entry criteria */
                                        /*      giving input section order */
        Alist           *sg_os_order;   /* list specifying output section */
                                        /*      ordering for the segment */
        sg_flags_t      sg_flags;
        APlist          *sg_sizesym;    /* size symbols for this segment */
        Xword           sg_align;       /* LCM of sh_addralign */
        Elf_Scn         *sg_fscn;       /* the SCN of the first section. */
        avl_node_t      sg_avlnode;     /* AVL book-keeping */
};

#define FLG_SG_P_VADDR          0x0001  /* p_vaddr segment attribute set */
#define FLG_SG_P_PADDR          0x0002  /* p_paddr segment attribute set */
#define FLG_SG_LENGTH           0x0004  /* length segment attribute set */
#define FLG_SG_P_ALIGN          0x0008  /* p_align segment attribute set */
#define FLG_SG_ROUND            0x0010  /* round segment attribute set */
#define FLG_SG_P_FLAGS          0x0020  /* p_flags segment attribute set */
#define FLG_SG_P_TYPE           0x0040  /* p_type segment attribute set */
#define FLG_SG_IS_ORDER         0x0080  /* input section ordering is required */
                                        /*      for this segment. */
#define FLG_SG_NOHDR            0x0100  /* don't map ELF or phdrs into */
                                        /*      this segment */
#define FLG_SG_EMPTY            0x0200  /* an empty segment specification */
                                        /*      no input sections will be */
                                        /*      associated to this section */
#define FLG_SG_KEY              0x0400  /* segment requires sort keys */
#define FLG_SG_NODISABLE        0x0800  /* FLG_SG_DISABLED is not allowed on */
                                        /*      this segment */
#define FLG_SG_DISABLED         0x1000  /* this segment is disabled */
#define FLG_SG_PHREQ            0x2000  /* this segment requires a program */
                                        /* header */
#define FLG_SG_ORDERED          0x4000  /* SEGMENT_ORDER segment */
```



```
struct sec_order {
        const char      *sco_secname;   /* section name to be ordered */
        Half            sco_flags;
};

#define FLG_SGO_USED    0x0001          /* was ordering used? */

typedef Half ec_flags_t;
struct ent_desc {                       /* input section entrance criteria */
        const char      *ec_name;       /* entrace criteria name, or NULL */
        Alist           *ec_files;      /* files from which to accept */
                                        /*      sections */
        const char      *ec_is_name;    /* input section name to match */
                                        /*      (NULL if none) */
        Word            ec_type;        /* section type */
        Word            ec_attrmask;    /* section attribute mask (AWX) */
        Word            ec_attrbits;    /* sections attribute bits */
        Sg_desc         *ec_segment;    /* output segment to enter if matched */
        Word            ec_ordndx;      /* index to determine where section */
                                        /*      meeting this criteria should */
                                        /*      inserted. Used for reordering */
                                        /*      of sections. */
        ec_flags_t      ec_flags;
        avl_node_t      ec_avlnode;     /* AVL book-keeping */
};

#define FLG_EC_BUILTIN  0x0001          /* built in descriptor */
#define FLG_EC_USED     0x0002          /* entrance criteria met? */
#define FLG_EC_CATCHALL 0x0004          /* Catches any section */

/*
 * Ent_desc_file is the type of element maintained in the ec_files Alist
 * of an entrance criteria descriptor. Each item maintains one file
 * path, and a set of flags that specify the type of comparison it implies,
 * and other information about it. The comparison type is maintained in
 * the bottom byte of the flags.
 */
#define TYP_ECF_MASK            0x00ff  /* Comparison type mask */
#define TYP_ECF_PATH            0       /* Compare to file path */
#define TYP_ECF_BASENAME        1       /* Compare to file basename */
#define TYP_ECF_OBJNAME         2       /* Compare to regular file basename, */
                                        /*       or to archive member name */
#define TYP_ECF_NUM             3

#define FLG_ECF_ARMEMBER        0x0100  /* name includes archive member */

typedef struct {
        Word            edf_flags;      /* Type of comparison */
        const char      *edf_name;      /* String to compare to */
        size_t          edf_name_len;   /* strlen(edf_name) */
} Ent_desc_file;

/*
 * One structure is allocated for a move entry, and associated to the symbol
 * against which a move is targeted.
 */
typedef struct {
        Move            *md_move;       /* original Move entry */
        Xword           md_start;       /* start position */
        Xword           md_len;         /* length of initialization */
        Word            md_oidx;        /* output Move entry index */
} Mv_desc;

/*
 * Symbol descriptor.
```

```
 */
typedef Lword           sd_flag_t;
struct sym_desc {
        Alist           *sd_GOTndxs;    /* list of associated GOT entries */
        Sym             *sd_sym;        /* pointer to symbol table entry */
        Sym             *sd_osym;       /* copy of the original symbol entry */
                                        /*      used only for local partial */
        Alist           *sd_move;       /* move information associated with a */
                                        /*      partially initialized symbol */
        const char      *sd_name;       /* symbols name */
        Ifl_desc        *sd_file;       /* file where symbol is taken */
        Is_desc         *sd_isc;        /* input section of symbol definition */
        Sym_aux         *sd_aux;        /* auxiliary global symbol info. */
        Word            sd_symndx;      /* index in output symbol table */
        Word            sd_shndx;       /* sect. index sym is associated w/ */
        sd_flag_t       sd_flags;       /* state flags */
        Half            sd_ref;         /* reference definition of symbol */
};

/*
 * The auxiliary symbol descriptor contains the additional information (beyond
 * the symbol descriptor) required to process global symbols.  These symbols are
 * accessed via an internal symbol hash table where locality of reference is
 * important for performance.
 */
struct sym_aux {
        APlist          *sa_dfiles;     /* files where symbol is defined */
        Sym             sa_sym;         /* copy of symtab entry */
        const char      *sa_vfile;      /* first unavailable definition */
        const char      *sa_rfile;      /* file with first symbol referenced */
        Word            sa_hash;        /* the pure hash value of symbol */
        Word            sa_PLTndx;      /* index into PLT for symbol */
        Word            sa_PLTGOTndx;   /* GOT entry indx for PLT indirection */
        Word            sa_linkndx;     /* index of associated symbol from */
                                        /*      ET_DYN file */
        Half            sa_symspec;     /* special symbol ids */
        Half            sa_overndx;     /* output file versioning index */
        Half            sa_dverndx;     /* dependency versioning index */
};

/*
 * Nodes used to track symbols in the global AVL symbol dictionary.
 */
struct sym_avlnode {
        avl_node_t      sav_node;       /* AVL node */
        Word            sav_hash;       /* symbol hash value */
        const char      *sav_name;      /* symbol name */
        Sym_desc        *sav_sdp;       /* symbol descriptor */
};

/*
 * These are the ids for processing of 'Special symbols'.  They are used
 * to set the sym->sd_aux->sa_symspec field.
 */
#define SDAUX_ID_ETEXT  1               /* etext && _etext symbol */
#define SDAUX_ID_EDATA  2               /* edata && _edata symbol */
#define SDAUX_ID_END    3               /* end, _end, && _END_ symbol */
#define SDAUX_ID_DYN    4               /* DYNAMIC && _DYNAMIC symbol */
#define SDAUX_ID_PLT    5               /* _PROCEDURE_LINKAGE_TABLE_ symbol */
#define SDAUX_ID_GOT    6               /* _GLOBAL_OFFSET_TABLE_ symbol */
#define SDAUX_ID_START  7               /* START_ && _START_ symbol */

/*
 * Flags for sym_desc.sd_flags
 */
#define FLG_SY_MVTOCOMM 0x00000001      /* assign symbol to common (.bss) */
```



```
                                        /*      this is a result of a */
                                        /*      copy reloc against sym */
#define FLG_SY_GLOBREF  0x00000002      /* a global reference has been seen */
#define FLG_SY_WEAKDEF  0x00000004      /* a weak definition has been used */
#define FLG_SY_CLEAN    0x00000008      /* 'Sym' entry points to original */
                                        /*      input file (read-only). */
#define FLG_SY_UPREQD   0x00000010      /* symbol value update is required, */
                                        /*      either it's used as an entry */
                                        /*      point or for relocation, but */
                                        /*      it must be updated even if */
                                        /*      the -s flag is in effect */
#define FLG_SY_NOTAVAIL 0x00000020      /* symbol is not available to the */
                                        /*      application either because it */
                                        /*      originates from an implicitly */
                                        /*      referenced shared object, or */
                                        /*      because it is not part of a */
                                        /*      specified version. */
#define FLG_SY_REDUCED  0x00000040      /* a global is reduced to local */
#define FLG_SY_VERSPROM 0x00000080      /* version definition has been */
                                        /*      promoted to output file */
#define FLG_SY_PROT     0x00000100      /* stv_protected visibility seen */
#define FLG_SY_MAPREF   0x00000200      /* symbol reference generated by user */
                                        /*      from mapfile */
#define FLG_SY_REFRSD   0x00000400      /* symbols sd_ref has been raised */
                                        /*      due to a copy-relocs */
                                        /*      weak-strong pairing */
#define FLG_SY_INTPOSE  0x00000800      /* symbol defines an interposer */
#define FLG_SY_INVALID  0x00001000      /* unwanted/erroneous symbol */
#define FLG_SY_SMGOT    0x00002000      /* small got index assigned to symbol */
                                        /*      sparc only */
#define FLG_SY_PARENT   0x00004000      /* symbol to be found in parent */
                                        /*    only used with direct bindings */
#define FLG_SY_LAZYLD   0x00008000      /* symbol to cause lazyloading of */
                                        /*      parent object */
#define FLG_SY_ISDISC   0x00010000      /* symbol is a member of a DISCARDED */
                                        /*      section (COMDAT) */
#define FLG_SY_PAREXPN  0x00020000      /* partially init. symbol to be */
                                        /*      expanded */
#define FLG_SY_PLTPAD   0x00040000      /* pltpadding has been allocated for */
                                        /*      this symbol */
#define FLG_SY_REGSYM   0x00080000      /* REGISTER symbol (sparc only) */
#define FLG_SY_SOFOUND  0x00100000      /* compared against an SO definition */
#define FLG_SY_EXTERN   0x00200000      /* symbol is external, allows -zdefs */
                                        /*    error suppression */
#define FLG_SY_MAPUSED  0x00400000      /* mapfile symbol used (occurred */
                                        /*    within a relocatable object) */
#define FLG_SY_COMMEXP  0x00800000      /* COMMON symbol which has been */
                                        /*      allocated */
#define FLG_SY_CMDREF   0x01000000      /* symbol was referenced from the */
                                        /*      command line.  (ld -u <>, */
                                        /*      ld -zrtldinfo=<>, ...) */
#define FLG_SY_SPECSEC  0x02000000      /* section index is reserved value */
                                        /*      ABS, COMMON, ... */
#define FLG_SY_TENTSYM  0x04000000      /* tentative symbol */
#define FLG_SY_VISIBLE  0x08000000      /* symbols visibility determined */
#define FLG_SY_STDFLTR  0x10000000      /* symbol is a standard filter */
#define FLG_SY_AUXFLTR  0x20000000      /* symbol is an auxiliary filter */
#define FLG_SY_DYNSORT  0x40000000      /* req. in dyn[sym|tls]sort section */
#define FLG_SY_NODYNSORT 0x80000000     /* excluded from dyn[sym_tls]sort sec */

#define FLG_SY_DEFAULT  0x0000100000000 /* global symbol, default */
#define FLG_SY_SINGLE   0x0000200000000 /* global symbol, singleton defined */
#define FLG_SY_PROTECT  0x0000400000000 /* global symbol, protected defined */
#define FLG_SY_EXPORT   0x0000800000000 /* global symbol, exported defined */

#define MSK_SY_GLOBAL \
```

```
1286         (FLG_SY_DEFAULT | FLG_SY_SINGLE | FLG_SY_PROTECT | FLG_SY_EXPORT)
1287                                         /* this mask indicates that the */
1288                                         /*    symbol has been explicitly */
1289                                         /*    defined within a mapfile */
1290                                         /*    definition, and is a candidate */
1291                                         /*    for versioning */

1293 #define FLG_SY_HIDDEN   0x0001000000000 /* global symbol, reduce to local */
1294 #define FLG_SY_ELIM     0x0002000000000 /* global symbol, eliminate */
1295 #define FLG_SY_IGNORE   0x0004000000000 /* global symbol, ignored */

1297 #define MSK_SY_LOCAL    (FLG_SY_HIDDEN | FLG_SY_ELIM | FLG_SY_IGNORE)
1298                                         /* this mask allows all local state */
1299                                         /*    flags to be removed when the */
1300                                         /*    symbol is copy relocated */

1302 #define FLG_SY_EXPDEF   0x0008000000000 /* symbol visibility defined */
1303                                         /*    explicitly */

1305 #define MSK_SY_NOAUTO   (FLG_SY_SINGLE | FLG_SY_EXPORT | FLG_SY_EXPDEF)
1306                                         /* this mask indicates that the */
1307                                         /*    symbol is not a candidate for */
1308                                         /*    auto-reduction/elimination */

1310 #define FLG_SY_MAPFILE  0x0010000000000 /* symbol attribute defined in a */
1311                                         /*    mapfile */
1312 #define FLG_SY_DIR      0x0020000000000 /* global symbol, direct bindings */
1313 #define FLG_SY_NDIR     0x0040000000000 /* global symbol, nondirect bindings */
1314 #define FLG_SY_OVERLAP  0x0080000000000 /* move entry overlap detected */
1315 #define FLG_SY_CAP      0x0100000000000 /* symbol is associated with */
1316                                         /*    capabilities */
1317 #define FLG_SY_DEFERRED 0x0200000000000 /* symbol should not be bound to */
1318                                         /*      during BIND_NOW relocations */

1320 /*
1321  * A symbol can only be truly hidden if it is not a capabilities symbol.
1322  */
1323 #define SYM_IS_HIDDEN(_sdp) \
1324         (((_sdp)->sd_flags & (FLG_SY_HIDDEN | FLG_SY_CAP)) == FLG_SY_HIDDEN)

1326 /*
1327  * Create a mask for (sym.st_other & visibility) since the gABI does not yet
1328  * define a ELF*_ST_OTHER macro.
1329  */
1330 #define MSK_SYM_VISIBILITY      0x7

1332 /*
1333  * Structure to manage the shared object definition lists.  There are two lists
1334  * that use this structure:
1335  *
1336  *  -   ofl_soneed; maintain the list of implicitly required dependencies
1337  *      (ie. shared objects needed by other shared objects).  These definitions
1338  *      may include RPATH's required to locate the dependencies, and any
1339  *      version requirements.
1340  *
1341  *  -   ofl_socntl; maintains the shared object control definitions.  These are
1342  *      provided by the user (via a mapfile) and are used to indicate any
1343  *      version control requirements.
1344  */
1345 struct  sdf_desc {
1346         const char      *sdf_name;      /* the shared objects file name */
1347         char            *sdf_rpath;     /* library search path DT_RPATH */
1348         const char      *sdf_rfile;     /* referencing file for diagnostics */
1349         Ifl_desc        *sdf_file;      /* the final input file descriptor */
1350         Alist           *sdf_vers;      /* list of versions that are required */
1351                                         /*      from this object */
```



```
1352         Alist           *sdf_verneed;   /* list of VERNEEDS to create for */
1353                                         /*      object via mapfile ADDVERS */
1354         Word            sdf_flags;
1355 };

1357 #define FLG_SDF_SELECT  0x01            /* version control selection required */
1358 #define FLG_SDF_VERIFY  0x02            /* version definition verification */
1359                                         /*      required */
1360 #define FLG_SDF_ADDVER  0x04            /* add VERNEED references */

1362 /*
1363  * Structure to manage shared object version usage requirements.
1364  */
1365 struct  sdv_desc {
1366         const char      *sdv_name;      /* version name */
1367         const char      *sdv_ref;       /* versions reference */
1368         Word            sdv_flags;      /* flags */
1369 };

1371 #define FLG_SDV_MATCHED 0x01            /* VERDEF found and matched */

1373 /*
1374  * Structures to manage versioning information.  Two versioning structures are
1375  * defined:
1376  *
1377  *   -  a version descriptor maintains a linked list of versions and their
1378  *      associated dependencies.  This is used to build the version definitions
1379  *      for an image being created (see map_symbol), and to determine the
1380  *      version dependency graph for any input files that are versioned.
1381  *
1382  *   -  a version index array contains each version of an input file that is
1383  *      being processed.  It informs us which versions are available for
1384  *      binding, and is used to generate any version dependency information.
1385  */
1386 struct  ver_desc {
1387         const char      *vd_name;       /* version name */
1388         Ifl_desc        *vd_file;       /* file that defined version */
1389         Word            vd_hash;        /* hash value of name */
1390         Half            vd_ndx;         /* coordinates with symbol index */
1391         Half            vd_flags;       /* version information */
1392         APlist          *vd_deps;       /* version dependencies */
1393         Ver_desc        *vd_ref;        /* dependency's first reference */
1394 };

1396 struct  ver_index {
1397         const char      *vi_name;       /* dependency version name */
1398         Half            vi_flags;       /* communicates availability */
1399         Half            vi_overndx;     /* index assigned to this version in */
1400                                         /*      output object Verneed section */
1401         Ver_desc        *vi_desc;       /* cross reference to descriptor */
1402 };

1404 /*
1405  * Define any internal version descriptor flags ([vd|vi]_flags).  Note that the
1406  * first byte is reserved for user visible flags (refer VER_FLG's in link.h).
1407  */
1408 #define MSK_VER_USER    0x0f            /* mask for user visible flags */

1410 #define FLG_VER_AVAIL   0x10            /* version is available for binding */
1411 #define FLG_VER_REFER   0x20            /* version has been referenced */
1412 #define FLG_VER_CYCLIC  0x40            /* a member of cyclic dependency */

1414 /*
1415  * isalist(1) descriptor - used to break an isalist string into its component
1416  * options.
1417  */
```

```
struct  isa_opt {
        char            *isa_name;      /* individual isa option name */
        size_t          isa_namesz;     /*      and associated size */
};

struct  isa_desc {
        char            *isa_list;      /* sysinfo(SI_ISALIST) list */
        size_t          isa_listsz;     /*      and associated size */
        Isa_opt         *isa_opt;       /* table of individual isa options */
        size_t          isa_optno;      /*      and associated number */
};

/*
 * uname(2) descriptor - used to break a utsname structure into its component
 * options (at least those that we're interested in).
 */
struct  uts_desc {
        char            *uts_osname;    /* operating system name */
        size_t          uts_osnamesz;   /*      and associated size */
        char            *uts_osrel;     /* operating system release */
        size_t          uts_osrelsz;    /*      and associated size */
};

/*
 * SHT_GROUP descriptor - used to track group sections at the global
 * level to resolve conflicts and determine which to keep.
 */
struct group_desc {
        Is_desc         *gd_isc;        /* input section descriptor */
        Is_desc         *gd_oisc;       /* overriding input section */
                                        /*      descriptor when discarded */
        const char      *gd_name;       /* group name (signature symbol) */
        Word            *gd_data;       /* data for group section */
        size_t          gd_cnt;         /* number of entries in group data */
};

/*
 * Indexes into the ld_support_funcs[] table.
 */
typedef enum {
        LDS_VERSION = 0,        /* Must be first and have value 0 */
        LDS_INPUT_DONE,
        LDS_START,
        LDS_ATEXIT,
        LDS_OPEN,
        LDS_FILE,
        LDS_INSEC,
        LDS_SEC,
        LDS_NUM
} Support_ndx;

/*
 * Structure to manage archive member caching.  Each archive has an archive
 * descriptor (Ar_desc) associated with it.  This contains pointers to the
 * archive symbol table (obtained by elf_getarsyms(3e)) and an auxiliary
 * structure (Ar_uax[]) that parallels this symbol table.  The member element
 * of this auxiliary table indicates whether the archive member associated with
 * the symbol offset has already been extracted (AREXTRACTED) or partially
 * processed (refer process_member()).
 */
typedef struct ar_mem {
        Elf             *am_elf;        /* elf descriptor for this member */
        const char      *am_name;       /* members name */
        const char      *am_path;       /* path (ie. lib(foo.o)) */
        Sym             *am_syms;       /* start of global symbols */
        char            *am_strs;       /* associated string table start */
```



```
        Xword           am_symn;        /* no. of global symbols */
} Ar_mem;

typedef struct ar_aux {
        Sym_desc        *au_syms;       /* internal symbol descriptor */
        Ar_mem          *au_mem;        /* associated member */
} Ar_aux;

#define FLG_ARMEM_PROC  (Ar_mem *)-1

typedef struct ar_desc {
        const char      *ad_name;       /* archive file name */
        Elf             *ad_elf;        /* elf descriptor for the archive */
        Elf_Arsym       *ad_start;      /* archive symbol table start */
        Ar_aux          *ad_aux;        /* auxiliary symbol information */
        dev_t           ad_stdev;       /* device id and inode number for */
        ino_t           ad_stino;       /*      multiple inclusion checks */
        ofl_flag_t      ad_flags;       /* archive specific cmd line flags */
} Ar_desc;

/*
 * Define any archive descriptor flags.  NOTE, make sure they do not clash with
 * any output file descriptor archive extraction flags, as these are saved in
 * the same entry (see MSK_OF1_ARCHIVE).
 */
#define FLG_ARD_EXTRACT 0x00010000      /* archive member has been extracted */

/* Mapfile versions supported by libld */
#define MFV_NONE        0       /* Not a valid version */
#define MFV_SYSV        1       /* Original System V syntax */
#define MFV_SOLARIS     2       /* Solaris mapfile syntax */
#define MFV_NUM         3       /* # of mapfile versions */


/*
 * Function Declarations.
 */
#if     defined(_ELF64)

#define ld_create_outfile       ld64_create_outfile
#define ld_ent_setup            ld64_ent_setup
#define ld_init_strings         ld64_init_strings
#define ld_init_target          ld64_init_target
#define ld_make_sections        ld64_make_sections
#define ld_main                 ld64_main
#define ld_ofl_cleanup          ld64_ofl_cleanup
#define ld_process_mem          ld64_process_mem
#define ld_reloc_init           ld64_reloc_init
#define ld_reloc_process        ld64_reloc_process
#define ld_sym_validate         ld64_sym_validate
#define ld_update_outfile       ld64_update_outfile

#else

#define ld_create_outfile       ld32_create_outfile
#define ld_ent_setup            ld32_ent_setup
#define ld_init_strings         ld32_init_strings
#define ld_init_target          ld32_init_target
#define ld_make_sections        ld32_make_sections
#define ld_main                 ld32_main
#define ld_ofl_cleanup          ld32_ofl_cleanup
#define ld_process_mem          ld32_process_mem
#define ld_reloc_init           ld32_reloc_init
#define ld_reloc_process        ld32_reloc_process
#define ld_sym_validate         ld32_sym_validate
#define ld_update_outfile       ld32_update_outfile
```

```
#endif

extern int              ld_getopt(Lm_list *, int, int, char **);

extern int              ld32_main(int, char **, Half);
extern int              ld64_main(int, char **, Half);

extern uintptr_t        ld_create_outfile(Ofl_desc *);
extern uintptr_t        ld_ent_setup(Ofl_desc *, Xword);
extern uintptr_t        ld_init_strings(Ofl_desc *);
extern int              ld_init_target(Lm_list *, Half mach);
extern uintptr_t        ld_make_sections(Ofl_desc *);
extern void             ld_ofl_cleanup(Ofl_desc *);
extern Ifl_desc         *ld_process_mem(const char *, const char *, char *,
                            size_t, Ofl_desc *, Rej_desc *);
extern uintptr_t        ld_reloc_init(Ofl_desc *);
extern uintptr_t        ld_reloc_process(Ofl_desc *);
extern uintptr_t        ld_sym_validate(Ofl_desc *);
extern uintptr_t        ld_update_outfile(Ofl_desc *);

#ifdef  __cplusplus
}
#endif

#endif  /* _LIBLD_H */
```

```
**********************************************************
   87677 Wed May 27 19:49:02 2015
new/usr/src/cmd/sgs/libconv/common/corenote.c
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
______unchanged_portion_omitted_


  75 const char *
  76 conv_cnote_auxv_type(Word type, Conv_fmt_flags_t fmt_flags,
  77     Conv_inv_buf_t *inv_buf)
  78 {
  79         static const Msg        types_0_22[] = {
  80                 MSG_AUXV_AT_NULL,               MSG_AUXV_AT_IGNORE,
  81                 MSG_AUXV_AT_EXECFD,             MSG_AUXV_AT_PHDR,
  82                 MSG_AUXV_AT_PHENT,              MSG_AUXV_AT_PHNUM,
  83                 MSG_AUXV_AT_PAGESZ,             MSG_AUXV_AT_BASE,
  84                 MSG_AUXV_AT_FLAGS,              MSG_AUXV_AT_ENTRY,
  85                 MSG_AUXV_AT_NOTELF,             MSG_AUXV_AT_UID,
  86                 MSG_AUXV_AT_EUID,               MSG_AUXV_AT_GID,
  87                 MSG_AUXV_AT_EGID,               MSG_AUXV_AT_PLATFORM,
  88                 MSG_AUXV_AT_HWCAP,              MSG_AUXV_AT_CLKTCK,
  89                 MSG_AUXV_AT_FPUCW,              MSG_AUXV_AT_DCACHEBSIZE,
  90                 MSG_AUXV_AT_ICACHEBSIZE,        MSG_AUXV_AT_UCACHEBSIZE,
  91                 MSG_AUXV_AT_IGNOREPPC
  92         };
  93         static const conv_ds_msg_t ds_types_0_22 = {
  94             CONV_DS_MSG_INIT(0, types_0_22) };

  96         static const Msg        types_2000_2011[] = {
  97                 MSG_AUXV_AT_SUN_UID,            MSG_AUXV_AT_SUN_RUID,
  98                 MSG_AUXV_AT_SUN_GID,            MSG_AUXV_AT_SUN_RGID,
  99                 MSG_AUXV_AT_SUN_LDELF,          MSG_AUXV_AT_SUN_LDSHDR,
 100                 MSG_AUXV_AT_SUN_LDNAME,         MSG_AUXV_AT_SUN_LPAGESZ,
 101                 MSG_AUXV_AT_SUN_PLATFORM,       MSG_AUXV_AT_SUN_HWCAP,
 102                 MSG_AUXV_AT_SUN_IFLUSH,         MSG_AUXV_AT_SUN_CPU
 103         };
 104         static const conv_ds_msg_t ds_types_2000_2011 = {
 105             CONV_DS_MSG_INIT(2000, types_2000_2011) };

 107         static const Msg        types_2014_2024[] = {
 107         static const Msg        types_2014_2023[] = {
 108                 MSG_AUXV_AT_SUN_EXECNAME,       MSG_AUXV_AT_SUN_MMU,
 109                 MSG_AUXV_AT_SUN_LDDATA,         MSG_AUXV_AT_SUN_AUXFLAGS,
 110                 MSG_AUXV_AT_SUN_EMULATOR,       MSG_AUXV_AT_SUN_BRANDNAME,
 111                 MSG_AUXV_AT_SUN_BRAND_AUX1,     MSG_AUXV_AT_SUN_BRAND_AUX2,
 112                 MSG_AUXV_AT_SUN_BRAND_AUX3,     MSG_AUXV_AT_SUN_HWCAP2,
 113                 MSG_AUXV_AT_SUN_SECFLAGS
 112                 MSG_AUXV_AT_SUN_BRAND_AUX3,     MSG_AUXV_AT_SUN_HWCAP2
 114         };
 115         static const conv_ds_msg_t ds_types_2014_2024 = {
 116             CONV_DS_MSG_INIT(2014, types_2014_2024) };
 114         static const conv_ds_msg_t ds_types_2014_2023 = {
 115             CONV_DS_MSG_INIT(2014, types_2014_2023) };

 118         static const conv_ds_t  *ds[] = {
 119                 CONV_DS_ADDR(ds_types_0_22), CONV_DS_ADDR(ds_types_2000_2011),
 120                 CONV_DS_ADDR(ds_types_2014_2024), NULL };
 119                 CONV_DS_ADDR(ds_types_2014_2023), NULL };

 122         return (conv_map_ds(ELFOSABI_NONE, EM_NONE, type, ds, fmt_flags,
 123             inv_buf));
 124 }
______unchanged_portion_omitted_
```



```
2588 #define PROCSECFLGSZ    CONV_EXPN_FIELD_DEF_PREFIX_SIZE + \
2589         MSG_PROC_SEC_ASLR_SIZE          + CONV_EXPN_FIELD_DEF_SEP_SIZE + \
2590         CONV_INV_BUFSIZE                + CONV_EXPN_FIELD_DEF_SUFFIX_SIZE

2592 /*
2593  * Ensure that Conv_cnote_pr_secflags_buf_t is large enough:
2594  *
2595  * PROCSECFLGSZ is the real minimum size of the buffer required by
2596  * conv_cnote_psecflags(). However, Conv_cnote_pr_secflags_buf_t uses
2597  * CONV_CNOTE_PSECFLAGS_FLAG_BUFSIZE to set the buffer size. We do things this
2598  * way because the definition of PROCSECFLGSZ uses information that is not
2599  * available in the environment of other programs that include the conv.h
2600  * header file.
2601  */
2602 #if (CONV_PSECFLAGS_BUFSIZE != PROCSECFLGSZ) && !defined(__lint)
2603 #define REPORT_BUFSIZE PROCSECFLGSZ
2604 #include "report_bufsize.h"
2605 #error "CONV_PSECFLAGS_BUFSIZE does not match PROCSECFLGSZ"
2606 #endif

2608 const char *
2609 conv_psecflags(int flags, Conv_fmt_flags_t fmt_flags,
2610     Conv_secflags_buf_t *secflags_buf)
2611 {
2612         static const Val_desc vda[] = {
2613                 { 0x0001,               MSG_PROC_SEC_ASLR },
2614                 { 0,                    0 }
2615         };
2616         static CONV_EXPN_FIELD_ARG conv_arg = {
2617             NULL, sizeof (secflags_buf->buf) };

2619         if (flags == 0)
2620                 return (MSG_ORIG(MSG_GBL_ZERO));

2622         conv_arg.buf = secflags_buf->buf;
2623         conv_arg.oflags = conv_arg.rflags = flags;
2624         (void) conv_expn_field(&conv_arg, vda, fmt_flags);

2626         return ((const char *)secflags_buf->buf);
2627 }
2628 #endif /* ! codereview */
```

```
**********************************************************
   39674 Wed May 27 19:49:02 2015
new/usr/src/cmd/sgs/libconv/common/corenote.msg
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
```

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#

#
# Copyright 2010 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#
# Copyright 2012 DEY Storage Systems, Inc.  All rights reserved.
# Copyright (c) 2013, Joyent, Inc. All rights reserved.
#

@ MSG_NT_PRSTATUS               "[ NT_PRSTATUS ]"
@ MSG_NT_PRFPREG                "[ NT_PRFPREG ]"
@ MSG_NT_PRPSINFO               "[ NT_PRPSINFO ]"
@ MSG_NT_PRXREG                 "[ NT_PRXREG ]"
@ MSG_NT_PLATFORM               "[ NT_PLATFORM ]"
@ MSG_NT_AUXV                   "[ NT_AUXV ]"
@ MSG_NT_GWINDOWS               "[ NT_GWINDOWS ]"
@ MSG_NT_ASRS                   "[ NT_ASRS ]"
@ MSG_NT_LDT                    "[ NT_LDT ]"
@ MSG_NT_PSTATUS                "[ NT_PSTATUS ]"
@ MSG_NT_PSINFO                 "[ NT_PSINFO ]"
@ MSG_NT_PRCRED                 "[ NT_PRCRED ]"
@ MSG_NT_UTSNAME                "[ NT_UTSNAME ]"
@ MSG_NT_LWPSTATUS              "[ NT_LWPSTATUS ]"
@ MSG_NT_LWPSINFO               "[ NT_LWPSINFO ]"
@ MSG_NT_PRPRIV                 "[ NT_PRPRIV ]"
@ MSG_NT_PRPRIVINFO             "[ NT_PRPRIVINFO ]"
@ MSG_NT_CONTENT                "[ NT_CONTENT ]"
@ MSG_NT_ZONENAME               "[ NT_ZONENAME ]"
@ MSG_NT_FDINFO                 "[ NT_FDINFO ]"
@ MSG_NT_SPYMASTER              "[ NT_SPYMASTER ]"


@ MSG_AUXV_AF_SUN_SETUGID               "AF_SUN_SETUGID"
@ MSG_AUXV_AF_SUN_HWCAPVERIFY           "AF_SUN_HWCAPVERIFY"
@ MSG_AUXV_AF_SUN_NOPLM                 "AF_SUN_NOPLM"


@ MSG_AUXV_AT_NULL                      "NULL"
```



```
@ MSG_AUXV_AT_IGNORE                    "IGNORE"
@ MSG_AUXV_AT_EXECFD                    "EXECFD"
@ MSG_AUXV_AT_PHDR                      "PHDR"
@ MSG_AUXV_AT_PHENT                     "PHENT"
@ MSG_AUXV_AT_PHNUM                     "PHNUM"
@ MSG_AUXV_AT_PAGESZ                    "PAGESZ"
@ MSG_AUXV_AT_BASE                      "BASE"
@ MSG_AUXV_AT_FLAGS                     "FLAGS"
@ MSG_AUXV_AT_ENTRY                     "ENTRY"
@ MSG_AUXV_AT_NOTELF                    "NOTELF"
@ MSG_AUXV_AT_UID                       "UID"
@ MSG_AUXV_AT_EUID                      "EUID"
@ MSG_AUXV_AT_GID                       "GID"
@ MSG_AUXV_AT_EGID                      "EGID"
@ MSG_AUXV_AT_PLATFORM                  "PLATFORM"
@ MSG_AUXV_AT_HWCAP                     "HWCAP"
@ MSG_AUXV_AT_CLKTCK                    "CLKTCK"
@ MSG_AUXV_AT_FPUCW                     "FPUCW"
@ MSG_AUXV_AT_DCACHEBSIZE               "DCACHEBSIZE"
@ MSG_AUXV_AT_ICACHEBSIZE               "ICACHEBSIZE"
@ MSG_AUXV_AT_UCACHEBSIZE               "UCACHEBSIZE"
@ MSG_AUXV_AT_IGNOREPPC                 "IGNOREPPC"
@ MSG_AUXV_AT_SUN_UID                   "SUN_UID"
@ MSG_AUXV_AT_SUN_RUID                  "SUN_RUID"
@ MSG_AUXV_AT_SUN_GID                   "SUN_GID"
@ MSG_AUXV_AT_SUN_RGID                  "SUN_RGID"
@ MSG_AUXV_AT_SUN_LDELF                 "SUN_LDELF"
@ MSG_AUXV_AT_SUN_LDSHDR                "SUN_LDSHDR"
@ MSG_AUXV_AT_SUN_LDNAME                "SUN_LDNAME"
@ MSG_AUXV_AT_SUN_LPAGESZ               "SUN_LPAGESZ"
@ MSG_AUXV_AT_SUN_PLATFORM              "SUN_PLATFORM"
@ MSG_AUXV_AT_SUN_HWCAP                 "SUN_HWCAP"
@ MSG_AUXV_AT_SUN_IFLUSH                "SUN_IFLUSH"
@ MSG_AUXV_AT_SUN_CPU                   "SUN_CPU"
@ MSG_AUXV_AT_SUN_EXECNAME              "SUN_EXECNAME"
@ MSG_AUXV_AT_SUN_MMU                   "SUN_MMU"
@ MSG_AUXV_AT_SUN_LDDATA                "SUN_LDDATA"
@ MSG_AUXV_AT_SUN_AUXFLAGS              "SUN_AUXFLAGS"
@ MSG_AUXV_AT_SUN_EMULATOR              "SUN_EMULATOR"
@ MSG_AUXV_AT_SUN_BRANDNAME             "SUN_BRANDNAME"
@ MSG_AUXV_AT_SUN_BRAND_AUX1            "SUN_BRAND_AUX1"
@ MSG_AUXV_AT_SUN_BRAND_AUX2            "SUN_BRAND_AUX2"
@ MSG_AUXV_AT_SUN_BRAND_AUX3            "SUN_BRAND_AUX3"
@ MSG_AUXV_AT_SUN_HWCAP2                "SUN_HWCAP2"
@ MSG_AUXV_AT_SUN_SECFLAGS              "SUN_SECFLAGS"
#endif /* ! codereview */


@ MSG_CC_CONTENT_STACK          "STACK"
@ MSG_CC_CONTENT_HEAP           "HEAP"
@ MSG_CC_CONTENT_SHFILE         "SHFILE"
@ MSG_CC_CONTENT_SHANON         "SHANON"
@ MSG_CC_CONTENT_TEXT           "TEXT"
@ MSG_CC_CONTENT_DATA           "DATA"
@ MSG_CC_CONTENT_RODATA         "RODATA"
@ MSG_CC_CONTENT_ANON           "ANON"
@ MSG_CC_CONTENT_SHM            "SHM"
@ MSG_CC_CONTENT_ISM            "ISM"
@ MSG_CC_CONTENT_DISM           "DISM"
@ MSG_CC_CONTENT_CTF            "CTF"
@ MSG_CC_CONTENT_SYMTAB         "SYMTAB"


@ MSG_ERRNO_EPERM               "[ EPERM ]"                     # 1
@ MSG_ERRNO_ENOENT              "[ ENOENT ]"            # 2
@ MSG_ERRNO_ESRCH               "[ ESRCH ]"                     # 3
```

```
125 @ MSG_ERRNO_EINTR               "[ EINTR ]"                 # 4
126 @ MSG_ERRNO_EIO                 "[ EIO ]"                   # 5
127 @ MSG_ERRNO_ENXIO               "[ ENXIO ]"                 # 6
128 @ MSG_ERRNO_E2BIG               "[ E2BIG ]"                 # 7
129 @ MSG_ERRNO_ENOEXEC             "[ ENOEXEC ]"           # 8
130 @ MSG_ERRNO_EBADF               "[ EBADF ]"                 # 9
131 @ MSG_ERRNO_ECHILD              "[ ECHILD ]"            # 10
132 @ MSG_ERRNO_EAGAIN              "[ EAGAIN ]"            # 11
133 @ MSG_ERRNO_ENOMEM              "[ ENOMEM ]"            # 12
134 @ MSG_ERRNO_EACCES              "[ EACCES ]"            # 13
135 @ MSG_ERRNO_EFAULT              "[ EFAULT ]"            # 14
136 @ MSG_ERRNO_ENOTBLK             "[ ENOTBLK ]"           # 15
137 @ MSG_ERRNO_EBUSY               "[ EBUSY ]"                 # 16
138 @ MSG_ERRNO_EEXIST              "[ EEXIST ]"            # 17
139 @ MSG_ERRNO_EXDEV               "[ EXDEV ]"                 # 18
140 @ MSG_ERRNO_ENODEV              "[ ENODEV ]"            # 19
141 @ MSG_ERRNO_ENOTDIR             "[ ENOTDIR ]"           # 20
142 @ MSG_ERRNO_EISDIR             "[ EISDIR ]"            # 21
143 @ MSG_ERRNO_EINVAL             "[ EINVAL ]"            # 22
144 @ MSG_ERRNO_ENFILE             "[ ENFILE ]"            # 23
145 @ MSG_ERRNO_EMFILE             "[ EMFILE ]"            # 24
146 @ MSG_ERRNO_ENOTTY             "[ ENOTTY ]"            # 25
147 @ MSG_ERRNO_ETXTBSY            "[ ETXTBSY ]"           # 26
148 @ MSG_ERRNO_EFBIG               "[ EFBIG ]"                 # 27
149 @ MSG_ERRNO_ENOSPC              "[ ENOSPC ]"            # 28
150 @ MSG_ERRNO_ESPIPE             "[ ESPIPE ]"            # 29
151 @ MSG_ERRNO_EROFS               "[ EROFS ]"                 # 30
152 @ MSG_ERRNO_EMLINK             "[ EMLINK ]"            # 31
153 @ MSG_ERRNO_EPIPE               "[ EPIPE ]"                 # 32
154 @ MSG_ERRNO_EDOM                "[ EDOM ]"                  # 33
155 @ MSG_ERRNO_ERANGE              "[ ERANGE ]"            # 34
156 @ MSG_ERRNO_ENOMSG             "[ ENOMSG ]"            # 35
157 @ MSG_ERRNO_EIDRM               "[ EIDRM ]"                 # 36
158 @ MSG_ERRNO_ECHRNG             "[ ECHRNG ]"            # 37
159 @ MSG_ERRNO_EL2NSYNC           "[ EL2NSYNC ]"          # 38
160 @ MSG_ERRNO_EL3HLT             "[ EL3HLT ]"            # 39
161 @ MSG_ERRNO_EL3RST             "[ EL3RST ]"            # 40
162 @ MSG_ERRNO_ELNRNG             "[ ELNRNG ]"            # 41
163 @ MSG_ERRNO_EUNATCH            "[ EUNATCH ]"           # 42
164 @ MSG_ERRNO_ENOCSI             "[ ENOCSI ]"            # 43
165 @ MSG_ERRNO_EL2HLT             "[ EL2HLT ]"            # 44
166 @ MSG_ERRNO_EDEADLK            "[ EDEADLK ]"           # 45
167 @ MSG_ERRNO_ENOLCK             "[ ENOLCK ]"            # 46
168 @ MSG_ERRNO_ECANCELED          "[ ECANCELED ]"         # 47
169 @ MSG_ERRNO_ENOTSUP            "[ ENOTSUP ]"           # 48
170 @ MSG_ERRNO_EDQUOT             "[ EDQUOT ]"            # 49
171 @ MSG_ERRNO_EBADE               "[ EBADE ]"                 # 50
172 @ MSG_ERRNO_EBADR               "[ EBADR ]"                 # 51
173 @ MSG_ERRNO_EXFULL             "[ EXFULL ]"            # 52
174 @ MSG_ERRNO_ENOANO             "[ ENOANO ]"            # 53
175 @ MSG_ERRNO_EBADRQC            "[ EBADRQC ]"           # 54
176 @ MSG_ERRNO_EBADSLT            "[ EBADSLT ]"           # 55
177 @ MSG_ERRNO_EDEADLOCK          "[ EDEADLOCK ]"         # 56
178 @ MSG_ERRNO_EBFONT             "[ EBFONT ]"            # 57
179 @ MSG_ERRNO_EOWNERDEAD         "[ EOWNERDEAD ]"                # 58
180 @ MSG_ERRNO_ENOTRECOVERABLE    "[ ENOTRECOVERABLE ]"   # 59
181 @ MSG_ERRNO_ENOSTR             "[ ENOSTR ]"            # 60
182 @ MSG_ERRNO_ENODATA            "[ ENODATA ]"           # 61
183 @ MSG_ERRNO_ETIME               "[ ETIME ]"                 # 62
184 @ MSG_ERRNO_ENOSR               "[ ENOSR ]"                 # 63
185 @ MSG_ERRNO_ENONET             "[ ENONET ]"            # 64
186 @ MSG_ERRNO_ENOPKG             "[ ENOPKG ]"            # 65
187 @ MSG_ERRNO_EREMOTE            "[ EREMOTE ]"           # 66
188 @ MSG_ERRNO_ENOLINK            "[ ENOLINK ]"           # 67
189 @ MSG_ERRNO_EADV                "[ EADV ]"                  # 68
190 @ MSG_ERRNO_ESRMNT             "[ ESRMNT ]"            # 69
```



```
191 @ MSG_ERRNO_ECOMM               "[ ECOMM ]"                 # 70
192 @ MSG_ERRNO_EPROTO              "[ EPROTO ]"            # 71
193 @ MSG_ERRNO_ELOCKUNMAPPED      "[ ELOCKUNMAPPED ]"         # 72
194 @ MSG_ERRNO_ENOTACTIVE         "[ ENOTACTIVE ]"            # 73
195 @ MSG_ERRNO_EMULTIHOP          "[ EMULTIHOP ]"         # 74
196 # errno 75 and 76 are not defined
197 @ MSG_ERRNO_EBADMSG             "[ EBADMSG ]"           # 77
198 @ MSG_ERRNO_ENAMETOOLONG       "[ ENAMETOOLONG ]"          # 78
199 @ MSG_ERRNO_EOVERFLOW          "[ EOVERFLOW ]"         # 79
200 @ MSG_ERRNO_ENOTUNIQ          "[ ENOTUNIQ ]"          # 80
201 @ MSG_ERRNO_EBADFD            "[ EBADFD ]"            # 81
202 @ MSG_ERRNO_EREMCHG           "[ EREMCHG ]"           # 82
203 @ MSG_ERRNO_ELIBACC           "[ ELIBACC ]"           # 83
204 @ MSG_ERRNO_ELIBBAD           "[ ELIBBAD ]"           # 84
205 @ MSG_ERRNO_ELIBSCN           "[ ELIBSCN ]"           # 85
206 @ MSG_ERRNO_ELIBMAX           "[ ELIBMAX ]"           # 86
207 @ MSG_ERRNO_ELIBEXEC          "[ ELIBEXEC ]"          # 87
208 @ MSG_ERRNO_EILSEQ            "[ EILSEQ ]"            # 88
209 @ MSG_ERRNO_ENOSYS            "[ ENOSYS ]"            # 89
210 @ MSG_ERRNO_ELOOP             "[ ELOOP ]"                 # 90
211 @ MSG_ERRNO_ERESTART          "[ ERESTART ]"          # 91
212 @ MSG_ERRNO_ESTRPIPE          "[ ESTRPIPE ]"          # 92
213 @ MSG_ERRNO_ENOTEMPTY         "[ ENOTEMPTY ]"         # 93
214 @ MSG_ERRNO_EUSERS            "[ EUSERS ]"            # 94
215 @ MSG_ERRNO_ENOTSOCK          "[ ENOTSOCK ]"          # 95
216 @ MSG_ERRNO_EDESTADDRREQ      "[ EDESTADDRREQ ]"          # 96
217 @ MSG_ERRNO_EMSGSIZE          "[ EMSGSIZE ]"          # 97
218 @ MSG_ERRNO_EPROTOTYPE        "[ EPROTOTYPE ]"            # 98
219 @ MSG_ERRNO_ENOPROTOOPT       "[ ENOPROTOOPT ]"           # 99
220 # Gap in ERRNO values 100-119
221 @ MSG_ERRNO_EPROTONOSUPPORT   "[ EPROTONOSUPPORT ]"   # 120
222 @ MSG_ERRNO_ESOCKTNOSUPPORT  "[ ESOCKTNOSUPPORT ]"  # 121
223 @ MSG_ERRNO_EOPNOTSUPP       "[ EOPNOTSUPP ]"            # 122
224 @ MSG_ERRNO_EPFNOSUPPORT     "[ EPFNOSUPPORT ]"          # 123
225 @ MSG_ERRNO_EAFNOSUPPORT     "[ EAFNOSUPPORT ]"          # 124
226 @ MSG_ERRNO_EADDRINUSE       "[ EADDRINUSE ]"            # 125
227 @ MSG_ERRNO_EADDRNOTAVAIL    "[ EADDRNOTAVAIL ]"         # 126
228 @ MSG_ERRNO_ENETDOWN         "[ ENETDOWN ]"          # 127
229 @ MSG_ERRNO_ENETUNREACH      "[ ENETUNREACH ]"           # 128
230 @ MSG_ERRNO_ENETRESET        "[ ENETRESET ]"         # 129
231 @ MSG_ERRNO_ECONNABORTED     "[ ECONNABORTED ]"          # 130
232 @ MSG_ERRNO_ECONNRESET      "[ ECONNRESET ]"            # 131
233 @ MSG_ERRNO_ENOBUFS         "[ ENOBUFS ]"           # 132
234 @ MSG_ERRNO_EISCONN         "[ EISCONN ]"           # 133
235 @ MSG_ERRNO_ENOTCONN        "[ ENOTCONN ]"          # 134
236 #XENIX has 135 - 142
237 @ MSG_ERRNO_ESHUTDOWN        "[ ESHUTDOWN ]"         # 143
238 @ MSG_ERRNO_ETOOMANYREFS     "[ ETOOMANYREFS ]"          # 144
239 @ MSG_ERRNO_ETIMEDOUT        "[ ETIMEDOUT ]"         # 145
240 @ MSG_ERRNO_ECONNREFUSED     "[ ECONNREFUSED ]"          # 146
241 @ MSG_ERRNO_EHOSTDOWN        "[ EHOSTDOWN ]"         # 147
242 @ MSG_ERRNO_EHOSTUNREACH     "[ EHOSTUNREACH ]"          # 148
243 @ MSG_ERRNO_EALREADY         "[ EALREADY ]"          # 149
244 @ MSG_ERRNO_EINPROGRESS      "[ EINPROGRESS ]"           # 150
245 @ MSG_ERRNO_ESTALE           "[ ESTALE ]"            # 151


248 @ MSG_PR_MODEL_UNKNOWN       "[ PR_MODEL_UNKNOWN ]"
249 @ MSG_PR_MODEL_ILP32        "[ PR_MODEL_ILP32 ]"
250 @ MSG_PR_MODEL_LP64          "[ PR_MODEL_LP64 ]"

252 @ MSG_PR_FLAGS_STOPPED       "PR_STOPPED"
253 @ MSG_PR_FLAGS_ISTOP         "PR_ISTOP"
254 @ MSG_PR_FLAGS_DSTOP         "PR_DSTOP"
255 @ MSG_PR_FLAGS_STEP          "PR_STEP"
256 @ MSG_PR_FLAGS_ASLEEP        "PR_ASLEEP"
```

```
@ MSG_PR_FLAGS_PCINVAL          "PR_PCINVAL"
@ MSG_PR_FLAGS_ASLWP            "PR_ASLWP"
@ MSG_PR_FLAGS_AGENT            "PR_AGENT"
@ MSG_PR_FLAGS_DETACH           "PR_DETACH"
@ MSG_PR_FLAGS_DAEMON           "PR_DAEMON"
@ MSG_PR_FLAGS_IDLE             "PR_IDLE"
@ MSG_PR_FLAGS_ISSYS            "PR_ISSYS"
@ MSG_PR_FLAGS_VFORKP           "PR_VFORKP"
@ MSG_PR_FLAGS_ORPHAN           "PR_ORPHAN"
@ MSG_PR_FLAGS_NOSIGCHLD        "PR_NOSIGCHLD"
@ MSG_PR_FLAGS_WAITPID          "PR_WAITPID"
@ MSG_PR_FLAGS_FORK             "PR_FORK"
@ MSG_PR_FLAGS_RLC              "PR_RLC"
@ MSG_PR_FLAGS_KLC              "PR_KLC"
@ MSG_PR_FLAGS_ASYNC            "PR_ASYNC"
@ MSG_PR_FLAGS_MSACCT           "PR_MSACCT"
@ MSG_PR_FLAGS_BPTADJ           "PR_BPTADJ"
@ MSG_PR_FLAGS_PTRACE           "PR_PTRACE"
@ MSG_PR_FLAGS_MSFORK           "PR_MSFORK"
@ MSG_PR_FLAGS_PCOMPAT          "PR_PCOMPAT"


@ MSG_PROC_FLAG_SSYS            "SSYS"
@ MSG_PROC_FLAG_SMSACCT         "SMSACCT"

@ MSG_PROC_SEC_ASLR             "ASLR"

#endif /* ! codereview */
@ MSG_PS_NONE                   "[ PS_NONE ]"
@ MSG_PS_QUERY                  "[ PS_QUERY ]"
@ MSG_PS_MYID                   "[ PS_MYID ]"
@ MSG_PS_SOFT                   "[ PS_SOFT ]"
@ MSG_PS_HARD                   "[ PS_HARD ]"
@ MSG_PS_QUERY_TYPE             "[ PS_QUERY_TYPE ]"


@ MSG_REG_SPARC_G0              "[ r0/g0 ]"
@ MSG_REG_SPARC_G1              "[ r1/g1 ]"
@ MSG_REG_SPARC_G2              "[ r2/g2 ]"
@ MSG_REG_SPARC_G3              "[ r3/g3 ]"
@ MSG_REG_SPARC_G4              "[ r4/g4 ]"
@ MSG_REG_SPARC_G5              "[ r5/g5 ]"
@ MSG_REG_SPARC_G6              "[ r6/g6 ]"
@ MSG_REG_SPARC_G7              "[ r7/g7 ]"
@ MSG_REG_SPARC_O0              "[ r8/o0 ]"
@ MSG_REG_SPARC_O1              "[ r9/o1 ]"
@ MSG_REG_SPARC_O2              "[ r10/o2 ]"
@ MSG_REG_SPARC_O3              "[ r11/o3 ]"
@ MSG_REG_SPARC_O4              "[ r12/o4 ]"
@ MSG_REG_SPARC_O5              "[ r13/o5 ]"
@ MSG_REG_SPARC_O6              "[ r14/o6/sp ]"
@ MSG_REG_SPARC_O7              "[ r15/o7 ]"
@ MSG_REG_SPARC_L0              "[ r16/l0 ]"
@ MSG_REG_SPARC_L1              "[ r17/l1 ]"
@ MSG_REG_SPARC_L2              "[ r18/l2 ]"
@ MSG_REG_SPARC_L3              "[ r19/l3 ]"
@ MSG_REG_SPARC_L4              "[ r20/l4 ]"
@ MSG_REG_SPARC_L5              "[ r21/l5 ]"
@ MSG_REG_SPARC_L6              "[ r22/l6 ]"
@ MSG_REG_SPARC_L7              "[ r23/l7 ]"
@ MSG_REG_SPARC_I0              "[ r24/i0 ]"
@ MSG_REG_SPARC_I1              "[ r25/i1 ]"
@ MSG_REG_SPARC_I2              "[ r26/i2 ]"
@ MSG_REG_SPARC_I3              "[ r27/i3 ]"
@ MSG_REG_SPARC_I4              "[ r28/i4 ]"
@ MSG_REG_SPARC_I5              "[ r29/i5 ]"
```



```
@ MSG_REG_SPARC_I6              "[ r30/i6/fp ]"
@ MSG_REG_SPARC_I7              "[ r31/i7 ]"
@ MSG_REG_SPARC_CCR             "[ ccr ]"
@ MSG_REG_SPARC_PSR             "[ psr ]"
@ MSG_REG_SPARC_PC              "[ pc ]"
@ MSG_REG_SPARC_nPC             "[ npc ]"
@ MSG_REG_SPARC_Y               "[ y ]"
@ MSG_REG_SPARC_ASI             "[ asi ]"
@ MSG_REG_SPARC_FPRS            "[ fprs ]"
@ MSG_REG_SPARC_WIM             "[ wim ]"
@ MSG_REG_SPARC_TBR             "[ tbr ]"

@ MSG_REG_AMD64_R15             "[ r15 ]"
@ MSG_REG_AMD64_R14             "[ r14 ]"
@ MSG_REG_AMD64_R13             "[ r13 ]"
@ MSG_REG_AMD64_R12             "[ r12 ]"
@ MSG_REG_AMD64_R11             "[ r11 ]"
@ MSG_REG_AMD64_R10             "[ r10 ]"
@ MSG_REG_AMD64_R9              "[ r9 ]"
@ MSG_REG_AMD64_R8              "[ r8 ]"
@ MSG_REG_AMD64_RDI             "[ rdi ]"
@ MSG_REG_AMD64_RSI             "[ rsi ]"
@ MSG_REG_AMD64_RBP             "[ rbp ]"
@ MSG_REG_AMD64_RBX             "[ rbx ]"
@ MSG_REG_AMD64_RDX             "[ rdx ]"
@ MSG_REG_AMD64_RCX             "[ rcx ]"
@ MSG_REG_AMD64_RAX             "[ rax ]"
@ MSG_REG_AMD64_TRAPNO          "[ trapno ]"
@ MSG_REG_AMD64_ERR             "[ err ]"
@ MSG_REG_AMD64_RIP             "[ rip ]"
@ MSG_REG_AMD64_CS              "[ cs ]"
@ MSG_REG_AMD64_RFL             "[ rfl ]"
@ MSG_REG_AMD64_RSP             "[ rsp ]"
@ MSG_REG_AMD64_SS              "[ ss ]"
@ MSG_REG_AMD64_FS              "[ fs ]"
@ MSG_REG_AMD64_GS              "[ gs ]"
@ MSG_REG_AMD64_ES              "[ es ]"
@ MSG_REG_AMD64_DS              "[ ds ]"
@ MSG_REG_AMD64_FSBASE          "[ fsbase ]"
@ MSG_REG_AMD64_GSBASE          "[ gsbase ]"

@ MSG_REG_I86_GS                "[ gs ]"
@ MSG_REG_I86_FS                "[ fs ]"
@ MSG_REG_I86_ES                "[ es ]"
@ MSG_REG_I86_DS                "[ ds ]"
@ MSG_REG_I86_EDI               "[ edi ]"
@ MSG_REG_I86_ESI               "[ esi ]"
@ MSG_REG_I86_EBP               "[ ebp ]"
@ MSG_REG_I86_ESP               "[ esp ]"
@ MSG_REG_I86_EBX               "[ ebx ]"
@ MSG_REG_I86_EDX               "[ edx ]"
@ MSG_REG_I86_ECX               "[ ecx ]"
@ MSG_REG_I86_EAX               "[ eax ]"
@ MSG_REG_I86_TRAPNO            "[ trapno ]"
@ MSG_REG_I86_ERR               "[ err ]"
@ MSG_REG_I86_EIP               "[ eip ]"
@ MSG_REG_I86_CS                "[ cs ]"
@ MSG_REG_I86_EFL               "[ efl ]"
@ MSG_REG_I86_UESP              "[ uesp ]"
@ MSG_REG_I86_SS                "[ ss ]"

@ MSG_PR_WHY_REQUESTED          "[ PR_REQUESTED ]"
@ MSG_PR_WHY_SIGNALLED          "[ PR_SIGNALLED ]"
@ MSG_PR_WHY_SYSENTRY           "[ PR_SYSENTRY ]"
@ MSG_PR_WHY_SYSEXIT            "[ PR_SYSEXIT ]"
@ MSG_PR_WHY_JOBCONTROL         "[ PR_JOBCONTROL ]"
```

```
389 @ MSG_PR_WHY_FAULTED            "[ PR_FAULTED ]"
390 @ MSG_PR_WHY_SUSPENDED          "[ PR_SUSPENDED ]"
391 @ MSG_PR_WHY_CHECKPOINT         "[ PR_CHECKPOINT ]"

393 @ MSG_PRIV_ALL                  "[ PRIV_ALL ]"
394 @ MSG_PRIV_MULTIPLE             "[ PRIV_MULTIPLE ]"
395 @ MSG_PRIV_NONE                 "[ PRIV_NONE ]"
396 @ MSG_PRIV_ALLZONE              "[ PRIV_ALLZONE ]"
397 @ MSG_PRIV_GLOBAL               "[ PRIV_ALLGLOBAL ]"


400 @ MSG_SA_ONSTACK                "SA_ONSTACK"
401 @ MSG_SA_RESETHAND             "SA_RESETHAND"
402 @ MSG_SA_RESTART               "SA_RESTART"
403 @ MSG_SA_SIGINFO              "SA_SIGINFO"
404 @ MSG_SA_NODEFER               "SA_NODEFER"
405 @ MSG_SA_NOCLDWAIT             "SA_NOCLDWAIT"
406 @ MSG_SA_NOCLDSTOP             "SA_NOCLDSTOP"


409 @ MSG_SOBJ_NONE                "[ SOBJ_NONE ]"
410 @ MSG_SOBJ_MUTEX               "[ SOBJ_MUTEX ]"
411 @ MSG_SOBJ_RWLOCK              "[ SOBJ_RWLOCK ]"
412 @ MSG_SOBJ_CV                  "[ SOBJ_CV ]"
413 @ MSG_SOBJ_SEMA                "[ SOBJ_SEMA ]"
414 @ MSG_SOBJ_USER                "[ SOBJ_USER ]"
415 @ MSG_SOBJ_USER_PI             "[ SOBJ_USER_PI ]"
416 @ MSG_SOBJ_SHUTTLE             "[ SOBJ_SHUTTLE ]"


419 @ MSG_SS_ONSTACK               "SS_ONSTACK"
420 @ MSG_SS_DISABLE               "SS_DISABLE"


423 @ MSG_SI_NOINFO                "[ SI_DTRACE ]"
424 @ MSG_SI_DTRACE                "[ SI_DTRACE ]"
425 @ MSG_SI_RCTL                  "[ SI_RCTL ]"
426 @ MSG_SI_USER                  "[ SI_USER ]"
427 @ MSG_SI_LWP                   "[ SI_LWP ]"
428 @ MSG_SI_QUEUE                 "[ SI_QUEUE ]"
429 @ MSG_SI_TIMER                 "[ SI_TIMER ]"
430 @ MSG_SI_ASYNCIO               "[ SI_ASYNCIO ]"
431 @ MSG_SI_MESGQ                 "[ SI_MESGQ ]"

433 @ MSG_SI_ILL_ILLOPC            "[ ILL_ILLOPC ]"
434 @ MSG_SI_ILL_ILLOPN            "[ ILL_ILLOPN ]"
435 @ MSG_SI_ILL_ILLADR            "[ ILL_ILLADR ]"
436 @ MSG_SI_ILL_ILLTRP            "[ ILL_ILLTRP ]"
437 @ MSG_SI_ILL_PRVOPC            "[ ILL_PRVOPC ]"
438 @ MSG_SI_ILL_PRVREG            "[ ILL_PRVREG ]"
439 @ MSG_SI_ILL_COPROC            "[ ILL_COPROC ]"
440 @ MSG_SI_ILL_BADSTK            "[ ILL_BADSTK ]"

442 @ MSG_SI_EMT_TAGOVF            "[ EMT_TAGOVF ]"
443 @ MSG_SI_EMT_CPCOVF            "[ EMT_CPCOVF ]"

445 @ MSG_SI_FPE_INTDIV            "[ FPE_INTDIV ]"
446 @ MSG_SI_FPE_INTOVF            "[ FPE_INTOVF ]"
447 @ MSG_SI_FPE_FLTDIV            "[ FPE_FLTDIV ]"
448 @ MSG_SI_FPE_FLTOVF            "[ FPE_FLTOVF ]"
449 @ MSG_SI_FPE_FLTUND            "[ FPE_FLTUND ]"
450 @ MSG_SI_FPE_FLTRES            "[ FPE_FLTRES ]"
451 @ MSG_SI_FPE_FLTINV            "[ FPE_FLTINV ]"
452 @ MSG_SI_FPE_FLTSUB            "[ FPE_FLTSUB ]"
453 @ MSG_SI_FPE_FLTDEN            "[ FPE_FLTDEN ]"
```



```
455 @ MSG_SI_SEGV_MAPERR           "[ SEGV_MAPERR ]"
456 @ MSG_SI_SEGV_ACCERR           "[ SEGV_ACCERR ]"

458 @ MSG_SI_BUS_ADRALN            "[ BUS_ADRALN ]"
459 @ MSG_SI_BUS_ADRERR            "[ BUS_ADRERR ]"
460 @ MSG_SI_BUS_OBJERR            "[ BUS_OBJERR ]"

462 @ MSG_SI_TRAP_BRKPT            "[ TRAP_BRKPT ]"
463 @ MSG_SI_TRAP_TRACE            "[ TRAP_TRACE ]"
464 @ MSG_SI_TRAP_RWATCH           "[ TRAP_RWATCH ]"
465 @ MSG_SI_TRAP_WWATCH           "[ TRAP_WWATCH ]"
466 @ MSG_SI_TRAP_XWATCH           "[ TRAP_XWATCH ]"
467 @ MSG_SI_TRAP_DTRACE           "[ TRAP_DTRACE ]"

469 @ MSG_SI_CLD_EXITED            "[ CLD_EXITED ]"
470 @ MSG_SI_CLD_KILLED            "[ CLD_KILLED ]"
471 @ MSG_SI_CLD_DUMPED            "[ CLD_DUMPED ]"
472 @ MSG_SI_CLD_TRAPPED           "[ CLD_TRAPPED ]"
473 @ MSG_SI_CLD_STOPPED           "[ CLD_STOPPED ]"
474 @ MSG_SI_CLD_CONTINUED         "[ CLD_CONTINUED ]"

476 @ MSG_SI_POLL_IN               "[ POLL_IN ]"
477 @ MSG_SI_POLL_OUT              "[ POLL_OUT ]"
478 @ MSG_SI_POLL_MSG              "[ POLL_MSG ]"
479 @ MSG_SI_POLL_ERR              "[ POLL_ERR ]"
480 @ MSG_SI_POLL_PRI              "[ POLL_PRI ]"
481 @ MSG_SI_POLL_HUP              "[ POLL_HUP ]"

483 @ MSG_SIGHUP                   "[ SIGHUP ]"
484 @ MSG_SIGHUP_ALT               "SIGHUP"
485 @ MSG_SIGINT                   "[ SIGINT ]"
486 @ MSG_SIGINT_ALT               "SIGINT"
487 @ MSG_SIGQUIT                  "[ SIGQUIT ]"
488 @ MSG_SIGQUIT_ALT              "SIGQUIT"
489 @ MSG_SIGILL                   "[ SIGILL ]"
490 @ MSG_SIGILL_ALT               "SIGILL"
491 @ MSG_SIGTRAP                  "[ SIGTRAP ]"
492 @ MSG_SIGTRAP_ALT              "SIGTRAP"
493 @ MSG_SIGABRT                  "[ SIGABRT ]"
494 @ MSG_SIGABRT_ALT              "SIGABRT"
495 @ MSG_SIGEMT                   "[ SIGEMT ]"
496 @ MSG_SIGEMT_ALT               "SIGEMT"
497 @ MSG_SIGFPE                   "[ SIGFPE ]"
498 @ MSG_SIGFPE_ALT               "SIGFPE"
499 @ MSG_SIGKILL                  "[ SIGKILL ]"
500 @ MSG_SIGKILL_ALT              "SIGKILL"
501 @ MSG_SIGBUS                   "[ SIGBUS ]"
502 @ MSG_SIGBUS_ALT               "SIGBUS"
503 @ MSG_SIGSEGV                  "[ SIGSEGV ]"
504 @ MSG_SIGSEGV_ALT              "SIGSEGV"
505 @ MSG_SIGSYS                   "[ SIGSYS ]"
506 @ MSG_SIGSYS_ALT               "SIGSYS"
507 @ MSG_SIGPIPE                  "[ SIGPIPE ]"
508 @ MSG_SIGPIPE_ALT              "SIGPIPE"
509 @ MSG_SIGALRM                  "[ SIGALRM ]"
510 @ MSG_SIGALRM_ALT              "SIGALRM"
511 @ MSG_SIGTERM                  "[ SIGTERM ]"
512 @ MSG_SIGTERM_ALT              "SIGTERM"
513 @ MSG_SIGUSR1                  "[ SIGUSR1 ]"
514 @ MSG_SIGUSR1_ALT              "SIGUSR1"
515 @ MSG_SIGUSR2                  "[ SIGUSR2 ]"
516 @ MSG_SIGUSR2_ALT              "SIGUSR2"
517 @ MSG_SIGCHLD                  "[ SIGCHLD ]"
518 @ MSG_SIGCHLD_ALT              "SIGCHLD"
519 @ MSG_SIGPWR                   "[ SIGPWR ]"
520 @ MSG_SIGPWR_ALT               "SIGPWR"
```

```
521 @ MSG_SIGWINCH                  "[ SIGWINCH ]"
522 @ MSG_SIGWINCH_ALT              "SIGWINCH"
523 @ MSG_SIGURG                    "[ SIGURG ]"
524 @ MSG_SIGURG_ALT                "SIGURG"
525 @ MSG_SIGPOLL                   "[ SIGPOLL ]"
526 @ MSG_SIGPOLL_ALT               "SIGPOLL"
527 @ MSG_SIGSTOP                   "[ SIGSTOP ]"
528 @ MSG_SIGSTOP_ALT               "SIGSTOP"
529 @ MSG_SIGTSTP                   "[ SIGTSTP ]"
530 @ MSG_SIGTSTP_ALT               "SIGTSTP"
531 @ MSG_SIGCONT                   "[ SIGCONT ]"
532 @ MSG_SIGCONT_ALT               "SIGCONT"
533 @ MSG_SIGTTIN                   "[ SIGTTIN ]"
534 @ MSG_SIGTTIN_ALT               "SIGTTIN"
535 @ MSG_SIGTTOU                   "[ SIGTTOU ]"
536 @ MSG_SIGTTOU_ALT               "SIGTTOU"
537 @ MSG_SIGVTALRM                 "[ SIGVTALRM ]"
538 @ MSG_SIGVTALRM_ALT             "SIGVTALRM"
539 @ MSG_SIGPROF                   "[ SIGPROF ]"
540 @ MSG_SIGPROF_ALT               "SIGPROF"
541 @ MSG_SIGXCPU                   "[ SIGXCPU ]"
542 @ MSG_SIGXCPU_ALT               "SIGXCPU"
543 @ MSG_SIGXFSZ                   "[ SIGXFSZ ]"
544 @ MSG_SIGXFSZ_ALT               "SIGXFSZ"
545 @ MSG_SIGWAITING                "[ SIGWAITING ]"
546 @ MSG_SIGWAITING_ALT            "SIGWAITING"
547 @ MSG_SIGLWP                    "[ SIGLWP ]"
548 @ MSG_SIGLWP_ALT                "SIGLWP"
549 @ MSG_SIGFREEZE                 "[ SIGFREEZE ]"
550 @ MSG_SIGFREEZE_ALT             "SIGFREEZE"
551 @ MSG_SIGTHAW                   "[ SIGTHAW ]"
552 @ MSG_SIGTHAW_ALT               "SIGTHAW"
553 @ MSG_SIGCANCEL                 "[ SIGCANCEL ]"
554 @ MSG_SIGCANCEL_ALT             "SIGCANCEL"
555 @ MSG_SIGLOST                   "[ SIGLOST ]"
556 @ MSG_SIGLOST_ALT               "SIGLOST"
557 @ MSG_SIGXRES                   "[ SIGXRES ]"
558 @ MSG_SIGXRES_ALT               "SIGXRES"
559 @ MSG_SIGJVM1                   "[ SIGJVM1 ]"
560 @ MSG_SIGJVM1_ALT               "SIGJVM1"
561 @ MSG_SIGJVM2                   "[ SIGJVM2 ]"
562 @ MSG_SIGJVM2_ALT               "SIGJVM2"

564 @ MSG_FLTILL                    "[ FLTILL ]"
565 @ MSG_FLTILL_ALT                "FLTILL"
566 @ MSG_FLTPRIV                   "[ FLTPRIV ]"
567 @ MSG_FLTPRIV_ALT               "FLTPRIV"
568 @ MSG_FLTBPT                    "[ FLTBPT ]"
569 @ MSG_FLTBPT_ALT                "FLTBPT"
570 @ MSG_FLTTRACE                  "[ FLTTRACE ]"
571 @ MSG_FLTTRACE_ALT              "FLTTRACE"
572 @ MSG_FLTACCESS                 "[ FLTACCESS ]"
573 @ MSG_FLTACCESS_ALT             "FLTACCESS"
574 @ MSG_FLTBOUNDS                 "[ FLTBOUNDS ]"
575 @ MSG_FLTBOUNDS_ALT             "FLTBOUNDS"
576 @ MSG_FLTIOVF                   "[ FLTIOVF ]"
577 @ MSG_FLTIOVF_ALT               "FLTIOVF"
578 @ MSG_FLTIZDIV                  "[ FLTIZDIV ]"
579 @ MSG_FLTIZDIV_ALT              "FLTIZDIV"
580 @ MSG_FLTFPE                    "[ FLTFPE ]"
581 @ MSG_FLTFPE_ALT                "FLTFPE"
582 @ MSG_FLTSTACK                  "[ FLTSTACK ]"
583 @ MSG_FLTSTACK_ALT              "FLTSTACK"
584 @ MSG_FLTPAGE                   "[ FLTPAGE ]"
585 @ MSG_FLTPAGE_ALT               "FLTPAGE"
586 @ MSG_FLTWATCH                  "[ FLTWATCH ]"
```



```
587 @ MSG_FLTWATCH_ALT              "FLTWATCH"
588 @ MSG_FLTCPCOVF                "[ FLTCPCOVF ]"
589 @ MSG_FLTCPCOVF_ALT            "FLTCPCOVF"

591 @ MSG_SYS_EXIT                          "[ exit ]"                  # 1
592 @ MSG_SYS_EXIT_ALT                      "exit"
593 @ MSG_SYS_2                             "2"                         # 2 (u)
594 @ MSG_SYS_READ                          "[ read ]"                  # 3
595 @ MSG_SYS_READ_ALT                      "read"
596 @ MSG_SYS_WRITE                         "[ write ]"                 # 4
597 @ MSG_SYS_WRITE_ALT                     "write"
598 @ MSG_SYS_OPEN                          "[ open ]"                  # 5
599 @ MSG_SYS_OPEN_ALT                      "open"
600 @ MSG_SYS_CLOSE                         "[ close ]"                 # 6
601 @ MSG_SYS_CLOSE_ALT                     "close"
602 @ MSG_SYS_7                             "7"                         # 7 (u)
603 @ MSG_SYS_8                             "8"                         # 8 (u)
604 @ MSG_SYS_LINK                          "[ link ]"                  # 9
605 @ MSG_SYS_LINK_ALT                      "link"
606 @ MSG_SYS_UNLINK                        "[ unlink ]"                # 10
607 @ MSG_SYS_UNLINK_ALT                    "unlink"
608 @ MSG_SYS_11                            "11"                        # 11 (u)
609 @ MSG_SYS_CHDIR                         "[ chdir ]"                 # 12
610 @ MSG_SYS_CHDIR_ALT                     "chdir"
611 @ MSG_SYS_TIME                          "[ time ]"                  # 13
612 @ MSG_SYS_TIME_ALT                      "time"
613 @ MSG_SYS_MKNOD                         "[ mknod ]"                 # 14
614 @ MSG_SYS_MKNOD_ALT                     "mknod"
615 @ MSG_SYS_CHMOD                         "[ chmod ]"                 # 15
616 @ MSG_SYS_CHMOD_ALT                     "chmod"
617 @ MSG_SYS_CHOWN                         "[ chown ]"                 # 16
618 @ MSG_SYS_CHOWN_ALT                     "chown"
619 @ MSG_SYS_BRK                           "[ brk ]"                   # 17
620 @ MSG_SYS_BRK_ALT                       "brk"
621 @ MSG_SYS_STAT                          "[ stat ]"                  # 18
622 @ MSG_SYS_STAT_ALT                      "stat"
623 @ MSG_SYS_LSEEK                         "[ lseek ]"                 # 19
624 @ MSG_SYS_LSEEK_ALT                     "lseek"
625 @ MSG_SYS_GETPID                        "[ getpid ]"                # 20
626 @ MSG_SYS_GETPID_ALT                    "getpid"
627 @ MSG_SYS_MOUNT                         "[ mount ]"                 # 21
628 @ MSG_SYS_MOUNT_ALT                     "mount"
629 @ MSG_SYS_22                            "22"                        # 22 (u)
630 @ MSG_SYS_SETUID                        "[ setuid ]"                # 23
631 @ MSG_SYS_SETUID_ALT                    "setuid"
632 @ MSG_SYS_GETUID                        "[ getuid ]"                # 24
633 @ MSG_SYS_GETUID_ALT                    "getuid"
634 @ MSG_SYS_STIME                         "[ stime ]"                 # 25
635 @ MSG_SYS_STIME_ALT                     "stime"
636 @ MSG_SYS_PCSAMPLE                      "[ pcsample ]"              # 26
637 @ MSG_SYS_PCSAMPLE_ALT                  "pcsample"
638 @ MSG_SYS_ALARM                         "[ alarm ]"                 # 27
639 @ MSG_SYS_ALARM_ALT                     "alarm"
640 @ MSG_SYS_FSTAT                         "[ fstat ]"                 # 28
641 @ MSG_SYS_FSTAT_ALT                     "fstat"
642 @ MSG_SYS_PAUSE                         "[ pause ]"                 # 29
643 @ MSG_SYS_PAUSE_ALT                     "pause"
644 @ MSG_SYS_30                            "30"                        # 30 (u)
645 @ MSG_SYS_STTY                          "[ stty ]"                  # 31
646 @ MSG_SYS_STTY_ALT                      "stty"
647 @ MSG_SYS_GTTY                          "[ gtty ]"                  # 32
648 @ MSG_SYS_GTTY_ALT                      "gtty"
649 @ MSG_SYS_ACCESS                        "[ access ]"                # 33
650 @ MSG_SYS_ACCESS_ALT                    "access"
651 @ MSG_SYS_NICE                          "[ nice ]"                  # 34
652 @ MSG_SYS_NICE_ALT                      "nice"
```

```
653 @ MSG_SYS_STATFS                    "[ statfs ]"                # 35
654 @ MSG_SYS_STATFS_ALT                "statfs"
655 @ MSG_SYS_SYNC                      "[ sync ]"                  # 36
656 @ MSG_SYS_SYNC_ALT                  "sync"
657 @ MSG_SYS_KILL                      "[ kill ]"                  # 37
658 @ MSG_SYS_KILL_ALT                  "kill"
659 @ MSG_SYS_FSTATFS                   "[ fstatfs ]"               # 38
660 @ MSG_SYS_FSTATFS_ALT               "fstatfs"
661 @ MSG_SYS_PGRPSYS                   "[ pgrpsys ]"               # 39
662 @ MSG_SYS_PGRPSYS_ALT               "pgrpsys"
663 @ MSG_SYS_UUCOPYSTR                 "[ uucopystr ]"             # 40
664 @ MSG_SYS_UUCOPYSTR_ALT             "uucopystr"
665 @ MSG_SYS_41                        "41"                        # 41 (u)
666 @ MSG_SYS_PIPE                      "[ pipe ]"                  # 42
667 @ MSG_SYS_PIPE_ALT                  "pipe"
668 @ MSG_SYS_TIMES                     "[ times ]"                 # 43
669 @ MSG_SYS_TIMES_ALT                 "times"
670 @ MSG_SYS_PROFIL                    "[ profil ]"                # 44
671 @ MSG_SYS_PROFIL_ALT                "profil"
672 @ MSG_SYS_FACCESSAT                 "[ faccessat ]"             # 45
673 @ MSG_SYS_FACCESSAT_ALT             "faccessat"
674 @ MSG_SYS_SETGID                    "[ setgid ]"                # 46
675 @ MSG_SYS_SETGID_ALT                "setgid"
676 @ MSG_SYS_GETGID                    "[ getgid ]"                # 47
677 @ MSG_SYS_GETGID_ALT                "getgid"
678 @ MSG_SYS_48                        "48"                        # 48 (u)
679 @ MSG_SYS_MSGSYS                    "[ msgsys ]"                # 49
680 @ MSG_SYS_MSGSYS_ALT                "msgsys"
681 @ MSG_SYS_SYSI86                    "[ sysi86 ]"                # 50
682 @ MSG_SYS_SYSI86_ALT                "sysi86"
683 @ MSG_SYS_ACCT                      "[ acct ]"                  # 51
684 @ MSG_SYS_ACCT_ALT                  "acct"
685 @ MSG_SYS_SHMSYS                    "[ shmsys ]"                # 52
686 @ MSG_SYS_SHMSYS_ALT                "shmsys"
687 @ MSG_SYS_SEMSYS                    "[ semsys ]"                # 53
688 @ MSG_SYS_SEMSYS_ALT                "semsys"
689 @ MSG_SYS_IOCTL                     "[ ioctl ]"                 # 54
690 @ MSG_SYS_IOCTL_ALT                 "ioctl"
691 @ MSG_SYS_UADMIN                    "[ uadmin ]"                # 55
692 @ MSG_SYS_UADMIN_ALT                "uadmin"
693 @ MSG_SYS_FCHOWNAT                  "[ fchownat ]"              # 56
694 @ MSG_SYS_FCHOWNAT_ALT              "fchownat"
695 @ MSG_SYS_UTSSYS                    "[ utssys ]"                # 57
696 @ MSG_SYS_UTSSYS_ALT                "utssys"
697 @ MSG_SYS_FDSYNC                    "[ fdsync ]"                # 58
698 @ MSG_SYS_FDSYNC_ALT                "fdsync"
699 @ MSG_SYS_EXECVE                    "[ execve ]"                # 59
700 @ MSG_SYS_EXECVE_ALT                "execve"
701 @ MSG_SYS_UMASK                     "[ umask ]"                 # 60
702 @ MSG_SYS_UMASK_ALT                 "umask"
703 @ MSG_SYS_CHROOT                    "[ chroot ]"                # 61
704 @ MSG_SYS_CHROOT_ALT                "chroot"
705 @ MSG_SYS_FCNTL                     "[ fcntl ]"                 # 62
706 @ MSG_SYS_FCNTL_ALT                 "fcntl"
707 @ MSG_SYS_ULIMIT                    "[ ulimit ]"                # 63
708 @ MSG_SYS_ULIMIT_ALT                "ulimit"
709 @ MSG_SYS_RENAMEAT                  "[ renameat ]"              # 64
710 @ MSG_SYS_RENAMEAT_ALT              "renameat"
711 @ MSG_SYS_UNLINKAT                  "[ unlinkat ]"              # 65
712 @ MSG_SYS_UNLINKAT_ALT              "unlinkat"
713 @ MSG_SYS_FSTATAT                   "[ fstatat ]"               # 66
714 @ MSG_SYS_FSTATAT_ALT               "fstatat"
715 @ MSG_SYS_FSTATAT64                 "[ fstatat64 ]"             # 67
716 @ MSG_SYS_FSTATAT64_ALT             "fstatat64"
717 @ MSG_SYS_OPENAT                    "[ openat ]"                # 68
718 @ MSG_SYS_OPENAT_ALT                "openat"
```



```
719 @ MSG_SYS_OPENAT64                  "[ openat64 ]"              # 69
720 @ MSG_SYS_OPENAT64_ALT              "openat64"
721 @ MSG_SYS_TASKSYS                   "[ tasksys ]"               # 70
722 @ MSG_SYS_TASKSYS_ALT               "tasksys       "
723 @ MSG_SYS_ACCTCTL                   "[ acctctl ]"               # 71
724 @ MSG_SYS_ACCTCTL_ALT               "acctctl       "
725 @ MSG_SYS_EXACCTSYS                 "[ exacctsys ]"             # 72
726 @ MSG_SYS_EXACCTSYS_ALT             "exacctsys"
727 @ MSG_SYS_GETPAGESIZES              "[ getpagesizes ]"          # 73
728 @ MSG_SYS_GETPAGESIZES_ALT          "getpagesizes"
729 @ MSG_SYS_RCTLSYS                   "[ rctlsys ]"               # 74
730 @ MSG_SYS_RCTLSYS_ALT               "rctlsys       "
731 @ MSG_SYS_SIDSYS                    "[ sidsys ]"                # 75
732 @ MSG_SYS_SIDSYS_ALT                "sidsys"
733 @ MSG_SYS_76                        "76"                        # 76 (u)
734 @ MSG_SYS_LWP_PARK                  "[ lwp_park ]"              # 77
735 @ MSG_SYS_LWP_PARK_ALT              "lwp_park"
736 @ MSG_SYS_SENDFILEV                 "[ sendfilev ]"             # 78
737 @ MSG_SYS_SENDFILEV_ALT             "sendfilev"
738 @ MSG_SYS_RMDIR                     "[ rmdir ]"                 # 79
739 @ MSG_SYS_RMDIR_ALT                 "rmdir"
740 @ MSG_SYS_MKDIR                     "[ mkdir ]"                 # 80
741 @ MSG_SYS_MKDIR_ALT                 "mkdir"
742 @ MSG_SYS_GETDENTS                  "[ getdents ]"              # 81
743 @ MSG_SYS_GETDENTS_ALT              "getdents"
744 @ MSG_SYS_PRIVSYS                   "[ privsys ]"               # 82
745 @ MSG_SYS_PRIVSYS_ALT               "privsys"
746 @ MSG_SYS_UCREDSYS                  "[ ucredsys ]"              # 83
747 @ MSG_SYS_UCREDSYS_ALT              "ucredsys"
748 @ MSG_SYS_SYSFS                     "[ sysfs ]"                 # 84
749 @ MSG_SYS_SYSFS_ALT                 "sysfs"
750 @ MSG_SYS_GETMSG                    "[ getmsg ]"                # 85
751 @ MSG_SYS_GETMSG_ALT                "getmsg"
752 @ MSG_SYS_PUTMSG                    "[ putmsg ]"                # 86
753 @ MSG_SYS_PUTMSG_ALT                "putmsg"
754 @ MSG_SYS_87                        "87"                        # 87 (u)
755 @ MSG_SYS_LSTAT                     "[ lstat ]"                 # 88
756 @ MSG_SYS_LSTAT_ALT                 "lstat"
757 @ MSG_SYS_SYMLINK                   "[ symlink ]"               # 89
758 @ MSG_SYS_SYMLINK_ALT               "symlink"
759 @ MSG_SYS_READLINK                  "[ readlink ]"              # 90
760 @ MSG_SYS_READLINK_ALT              "readlink"
761 @ MSG_SYS_SETGROUPS                 "[ setgroups ]"             # 91
762 @ MSG_SYS_SETGROUPS_ALT             "setgroups"
763 @ MSG_SYS_GETGROUPS                 "[ getgroups ]"             # 92
764 @ MSG_SYS_GETGROUPS_ALT             "getgroups"
765 @ MSG_SYS_FCHMOD                    "[ fchmod ]"                # 93
766 @ MSG_SYS_FCHMOD_ALT                "fchmod"
767 @ MSG_SYS_FCHOWN                    "[ fchown ]"                # 94
768 @ MSG_SYS_FCHOWN_ALT                "fchown"
769 @ MSG_SYS_SIGPROCMASK               "[ sigprocmask ]"           # 95
770 @ MSG_SYS_SIGPROCMASK_ALT           "sigprocmask"
771 @ MSG_SYS_SIGSUSPEND                "[ sigsuspend ]"            # 96
772 @ MSG_SYS_SIGSUSPEND_ALT            "sigsuspend"
773 @ MSG_SYS_SIGALTSTACK               "[ sigaltstack ]"           # 97
774 @ MSG_SYS_SIGALTSTACK_ALT           "sigaltstack"
775 @ MSG_SYS_SIGACTION                 "[ sigaction ]"             # 98
776 @ MSG_SYS_SIGACTION_ALT             "sigaction"
777 @ MSG_SYS_SIGPENDING                "[ sigpending ]"            # 99
778 @ MSG_SYS_SIGPENDING_ALT            "sigpending"
779 @ MSG_SYS_CONTEXT                   "[ context ]"               # 100
780 @ MSG_SYS_CONTEXT_ALT               "context"
781 @ MSG_SYS_101                       "101"                       # 101(u)
782 @ MSG_SYS_102                       "102"                       # 102(u)
783 @ MSG_SYS_STATVFS                   "[ statvfs ]"               # 103
784 @ MSG_SYS_STATVFS_ALT               "statvfs"
```

```
785 @ MSG_SYS_FSTATVFS                  "[ fstatvfs ]"              # 104
786 @ MSG_SYS_FSTATVFS_ALT              "fstatvfs"
787 @ MSG_SYS_GETLOADAVG                "[ getloadavg ]"            # 105
788 @ MSG_SYS_GETLOADAVG_ALT            "getloadavg"
789 @ MSG_SYS_NFSSYS                    "[ nfssys ]"                # 106
790 @ MSG_SYS_NFSSYS_ALT                "nfssys"
791 @ MSG_SYS_WAITID                    "[ waitid ]"                # 107
792 @ MSG_SYS_WAITID_ALT                "waitid"
793 @ MSG_SYS_SIGSENDSYS                "[ sigsendsys ]"            # 108
794 @ MSG_SYS_SIGSENDSYS_ALT            "sigsendsys"
795 @ MSG_SYS_HRTSYS                    "[ hrtsys ]"                # 109
796 @ MSG_SYS_HRTSYS_ALT                "hrtsys"
797 @ MSG_SYS_UTIMESYS                  "[ utimesys ]"              # 110
798 @ MSG_SYS_UTIMESYS_ALT              "utimesys"
799 @ MSG_SYS_SIGRESEND                 "[ sigresend ]"             # 111
800 @ MSG_SYS_SIGRESEND_ALT             "sigresend"
801 @ MSG_SYS_PRIOCNTLSYS               "[ priocntlsys ]"           # 112
802 @ MSG_SYS_PRIOCNTLSYS_ALT           "priocntlsys"
803 @ MSG_SYS_PATHCONF                  "[ pathconf ]"              # 113
804 @ MSG_SYS_PATHCONF_ALT              "pathconf"
805 @ MSG_SYS_MINCORE                   "[ mincore ]"               # 114
806 @ MSG_SYS_MINCORE_ALT               "mincore"
807 @ MSG_SYS_MMAP                      "[ mmap ]"                  # 115
808 @ MSG_SYS_MMAP_ALT                  "mmap"
809 @ MSG_SYS_MPROTECT                  "[ mprotect ]"              # 116
810 @ MSG_SYS_MPROTECT_ALT              "mprotect"
811 @ MSG_SYS_MUNMAP                    "[ munmap ]"                # 117
812 @ MSG_SYS_MUNMAP_ALT                "munmap"
813 @ MSG_SYS_FPATHCONF                 "[ fpathconf ]"             # 118
814 @ MSG_SYS_FPATHCONF_ALT             "fpathconf"
815 @ MSG_SYS_VFORK                     "[ vfork ]"                 # 119
816 @ MSG_SYS_VFORK_ALT                 "vfork"
817 @ MSG_SYS_FCHDIR                    "[ fchdir ]"                # 120
818 @ MSG_SYS_FCHDIR_ALT                "fchdir"
819 @ MSG_SYS_READV                     "[ readv ]"                 # 121
820 @ MSG_SYS_READV_ALT                 "readv"
821 @ MSG_SYS_WRITEV                    "[ writev ]"                # 122
822 @ MSG_SYS_WRITEV_ALT                "writev"
823 @ MSG_SYS_123                       "123"                       # 123(u)
824 @ MSG_SYS_124                       "124"                       # 124(u)
825 @ MSG_SYS_125                       "125"                       # 125(u)
826 @ MSG_SYS_126                       "126"                       # 126(u)
827 @ MSG_SYS_MMAPOBJ                   "[ mmapobj ]"               # 127
828 @ MSG_SYS_MMAPOBJ_ALT               "mmapobj"
829 @ MSG_SYS_SETRLIMIT                 "[ setrlimit ]"             # 128
830 @ MSG_SYS_SETRLIMIT_ALT             "setrlimit"
831 @ MSG_SYS_GETRLIMIT                 "[ getrlimit ]"             # 129
832 @ MSG_SYS_GETRLIMIT_ALT             "getrlimit"
833 @ MSG_SYS_LCHOWN                    "[ lchown ]"                # 130
834 @ MSG_SYS_LCHOWN_ALT                "lchown"
835 @ MSG_SYS_MEMCNTL                   "[ memcntl ]"               # 131
836 @ MSG_SYS_MEMCNTL_ALT               "memcntl"
837 @ MSG_SYS_GETPMSG                   "[ getpmsg ]"               # 132
838 @ MSG_SYS_GETPMSG_ALT               "getpmsg"
839 @ MSG_SYS_PUTPMSG                   "[ putpmsg ]"               # 133
840 @ MSG_SYS_PUTPMSG_ALT               "putpmsg"
841 @ MSG_SYS_RENAME                    "[ rename ]"                # 134
842 @ MSG_SYS_RENAME_ALT                "rename"
843 @ MSG_SYS_UNAME                     "[ uname ]"                 # 135
844 @ MSG_SYS_UNAME_ALT                 "uname"
845 @ MSG_SYS_SETEGID                   "[ setegid ]"               # 136
846 @ MSG_SYS_SETEGID_ALT               "setegid"
847 @ MSG_SYS_SYSCONFIG                 "[ sysconfig ]"             # 137
848 @ MSG_SYS_SYSCONFIG_ALT             "sysconfig"
849 @ MSG_SYS_ADJTIME                   "[ adjtime ]"               # 138
850 @ MSG_SYS_ADJTIME_ALT               "adjtime"
```



```
851 @ MSG_SYS_SYSTEMINFO                "[ systeminfo ]"            # 139
852 @ MSG_SYS_SYSTEMINFO_ALT            "systeminfo"
853 @ MSG_SYS_SHAREFS                   "[ sharefs ]"               # 140
854 @ MSG_SYS_SHAREFS_ALT               "sharefs"
855 @ MSG_SYS_SETEUID                   "[ seteuid ]"               # 141
856 @ MSG_SYS_SETEUID_ALT               "seteuid"
857 @ MSG_SYS_FORKSYS                   "[ forksys ]"               # 142
858 @ MSG_SYS_FORKSYS_ALT               "forksys"
859 @ MSG_SYS_143                       "143"                       # 143(u)
860 @ MSG_SYS_SIGTIMEDWAIT              "[ sigtimedwait ]"          # 144
861 @ MSG_SYS_SIGTIMEDWAIT_ALT          "sigtimedwait"
862 @ MSG_SYS_LWP_INFO                  "[ lwp_info ]"              # 145
863 @ MSG_SYS_LWP_INFO_ALT              "lwp_info"
864 @ MSG_SYS_YIELD                     "[ yield ]"                 # 146
865 @ MSG_SYS_YIELD_ALT                 "yield"
866 @ MSG_SYS_147                       "147"                       # 147(u)
867 @ MSG_SYS_LWP_SEMA_POST             "[ lwp_sema_post ]"         # 148
868 @ MSG_SYS_LWP_SEMA_POST_ALT         "lwp_sema_post"
869 @ MSG_SYS_LWP_SEMA_TRYWAIT          "[ lwp_sema_trywait ]"      # 149
870 @ MSG_SYS_LWP_SEMA_TRYWAIT_ALT      "lwp_sema_trywait"
871 @ MSG_SYS_LWP_DETACH                "[ lwp_detach ]"            # 150
872 @ MSG_SYS_LWP_DETACH_ALT            "lwp_detach"
873 @ MSG_SYS_CORECTL                   "[ corectl ]"               # 151
874 @ MSG_SYS_CORECTL_ALT               "corectl"
875 @ MSG_SYS_MODCTL                    "[ modctl ]"                # 152
876 @ MSG_SYS_MODCTL_ALT                "modctl"
877 @ MSG_SYS_FCHROOT                   "[ fchroot ]"               # 153
878 @ MSG_SYS_FCHROOT_ALT               "fchroot"
879 @ MSG_SYS_154                       "154"                       # 154(u)
880 @ MSG_SYS_VHANGUP                   "[ vhangup ]"               # 155
881 @ MSG_SYS_VHANGUP_ALT               "vhangup"
882 @ MSG_SYS_GETTIMEOFDAY              "[ gettimeofday ]"          # 156
883 @ MSG_SYS_GETTIMEOFDAY_ALT          "gettimeofday"
884 @ MSG_SYS_GETITIMER                 "[ getitimer ]"             # 157
885 @ MSG_SYS_GETITIMER_ALT             "getitimer"
886 @ MSG_SYS_SETITIMER                 "[ setitimer ]"             # 158
887 @ MSG_SYS_SETITIMER_ALT             "setitimer"
888 @ MSG_SYS_LWP_CREATE                "[ lwp_create ]"            # 159
889 @ MSG_SYS_LWP_CREATE_ALT            "lwp_create"
890 @ MSG_SYS_LWP_EXIT                  "[ lwp_exit ]"              # 160
891 @ MSG_SYS_LWP_EXIT_ALT              "lwp_exit"
892 @ MSG_SYS_LWP_SUSPEND               "[ lwp_suspend ]"           # 161
893 @ MSG_SYS_LWP_SUSPEND_ALT           "lwp_suspend"
894 @ MSG_SYS_LWP_CONTINUE              "[ lwp_continue ]"          # 162
895 @ MSG_SYS_LWP_CONTINUE_ALT          "lwp_continue"
896 @ MSG_SYS_LWP_KILL                  "[ lwp_kill ]"              # 163
897 @ MSG_SYS_LWP_KILL_ALT              "lwp_kill"
898 @ MSG_SYS_LWP_SELF                  "[ lwp_self ]"              # 164
899 @ MSG_SYS_LWP_SELF_ALT              "lwp_self"
900 @ MSG_SYS_LWP_SIGMASK               "[ lwp_sigmask ]"           # 165
901 @ MSG_SYS_LWP_SIGMASK_ALT           "lwp_sigmask"
902 @ MSG_SYS_LWP_PRIVATE               "[ lwp_private ]"           # 166
903 @ MSG_SYS_LWP_PRIVATE_ALT           "lwp_private"
904 @ MSG_SYS_LWP_WAIT                  "[ lwp_wait ]"              # 167
905 @ MSG_SYS_LWP_WAIT_ALT              "lwp_wait"
906 @ MSG_SYS_LWP_MUTEX_WAKEUP          "[ lwp_mutex_wakeup ]"      # 168
907 @ MSG_SYS_LWP_MUTEX_WAKEUP_ALT      "lwp_mutex_wakeup"
908 @ MSG_SYS_169                       "169"                       # 169(u)
909 @ MSG_SYS_LWP_COND_WAIT             "[ lwp_cond_wait ]"         # 170
910 @ MSG_SYS_LWP_COND_WAIT_ALT         "lwp_cond_wait"
911 @ MSG_SYS_LWP_COND_SIGNAL           "[ lwp_cond_signal ]"       # 171
912 @ MSG_SYS_LWP_COND_SIGNAL_ALT       "lwp_cond_signal"
913 @ MSG_SYS_LWP_COND_BROADCAST        "[ lwp_cond_broadcast ]"    # 172
914 @ MSG_SYS_LWP_COND_BROADCAST_ALT    "lwp_cond_broadcast"
915 @ MSG_SYS_PREAD                     "[ pread ]"                 # 173
916 @ MSG_SYS_PREAD_ALT                 "pread"
```

```
917 @ MSG_SYS_PWRITE                    "[ pwrite ]"                # 174
918 @ MSG_SYS_PWRITE_ALT                "pwrite"
919 @ MSG_SYS_LLSEEK                    "[ llseek ]"                # 175
920 @ MSG_SYS_LLSEEK_ALT                "llseek"
921 @ MSG_SYS_INST_SYNC                 "[ inst_sync ]"             # 176
922 @ MSG_SYS_INST_SYNC_ALT             "inst_sync"
923 @ MSG_SYS_BRAND                     "[ brand ]"                 # 177
924 @ MSG_SYS_BRAND_ALT                 "brand"
925 @ MSG_SYS_KAIO                      "[ kaio ]"                  # 178
926 @ MSG_SYS_KAIO_ALT                  "kaio"
927 @ MSG_SYS_CPC                       "[ cpc ]"                   # 179
928 @ MSG_SYS_CPC_ALT                   "cpc"
929 @ MSG_SYS_LGRPSYS                   "[ lgrpsys ]"               # 180
930 @ MSG_SYS_LGRPSYS_ALT               "lgrpsys"
931 @ MSG_SYS_RUSAGESYS                 "[ rusagesys ]"             # 181
932 @ MSG_SYS_RUSAGESYS_ALT             "rusagesys"
933 @ MSG_SYS_PORT                      "[ port ]"                  # 182
934 @ MSG_SYS_PORT_ALT                  "port"
935 @ MSG_SYS_POLLSYS                   "[ pollsys ]"               # 183
936 @ MSG_SYS_POLLSYS_ALT               "pollsys"
937 @ MSG_SYS_LABELSYS                  "[ labelsys ]"              # 184
938 @ MSG_SYS_LABELSYS_ALT              "labelsys"
939 @ MSG_SYS_ACL                       "[ acl ]"                   # 185
940 @ MSG_SYS_ACL_ALT                   "acl"
941 @ MSG_SYS_AUDITSYS                  "[ auditsys ]"              # 186
942 @ MSG_SYS_AUDITSYS_ALT              "auditsys"
943 @ MSG_SYS_PROCESSOR_BIND            "[ processor_bind ]"        # 187
944 @ MSG_SYS_PROCESSOR_BIND_ALT        "processor_bind"
945 @ MSG_SYS_PROCESSOR_INFO            "[ processor_info ]"        # 188
946 @ MSG_SYS_PROCESSOR_INFO_ALT        "processor_info"
947 @ MSG_SYS_P_ONLINE                  "[ p_online ]"              # 189
948 @ MSG_SYS_P_ONLINE_ALT              "p_online"
949 @ MSG_SYS_SIGQUEUE                  "[ sigqueue ]"              # 190
950 @ MSG_SYS_SIGQUEUE_ALT              "sigqueue"
951 @ MSG_SYS_CLOCK_GETTIME             "[ clock_gettime ]"         # 191
952 @ MSG_SYS_CLOCK_GETTIME_ALT         "clock_gettime"
953 @ MSG_SYS_CLOCK_SETTIME             "[ clock_settime ]"         # 192
954 @ MSG_SYS_CLOCK_SETTIME_ALT         "clock_settime"
955 @ MSG_SYS_CLOCK_GETRES              "[ clock_getres ]"          # 193
956 @ MSG_SYS_CLOCK_GETRES_ALT          "clock_getres"
957 @ MSG_SYS_TIMER_CREATE              "[ timer_create ]"          # 194
958 @ MSG_SYS_TIMER_CREATE_ALT          "timer_create"
959 @ MSG_SYS_TIMER_DELETE              "[ timer_delete ]"          # 195
960 @ MSG_SYS_TIMER_DELETE_ALT          "timer_delete"
961 @ MSG_SYS_TIMER_SETTIME             "[ timer_settime ]"         # 196
962 @ MSG_SYS_TIMER_SETTIME_ALT         "timer_settime"
963 @ MSG_SYS_TIMER_GETTIME             "[ timer_gettime ]"         # 197
964 @ MSG_SYS_TIMER_GETTIME_ALT         "timer_gettime"
965 @ MSG_SYS_TIMER_GETOVERRUN          "[ timer_getoverrun ]"      # 198
966 @ MSG_SYS_TIMER_GETOVERRUN_ALT      "timer_getoverrun"
967 @ MSG_SYS_NANOSLEEP                 "[ nanosleep ]"             # 199
968 @ MSG_SYS_NANOSLEEP_ALT             "nanosleep"
969 @ MSG_SYS_FACL                      "[ facl ]"                  # 200
970 @ MSG_SYS_FACL_ALT                  "facl"
971 @ MSG_SYS_DOOR                      "[ door ]"                  # 201
972 @ MSG_SYS_DOOR_ALT                  "door"
973 @ MSG_SYS_SETREUID                  "[ setreuid ]"              # 202
974 @ MSG_SYS_SETREUID_ALT              "setreuid"
975 @ MSG_SYS_SETREGID                  "[ setregid ]"              # 203
976 @ MSG_SYS_SETREGID_ALT              "setregid"
977 @ MSG_SYS_INSTALL_UTRAP             "[ install_utrap ]"         # 204
978 @ MSG_SYS_INSTALL_UTRAP_ALT         "install_utrap"
979 @ MSG_SYS_SIGNOTIFY                 "[ signotify ]"             # 205
980 @ MSG_SYS_SIGNOTIFY_ALT             "signotify"
981 @ MSG_SYS_SCHEDCTL                  "[ schedctl ]"              # 206
982 @ MSG_SYS_SCHEDCTL_ALT              "schedctl"
```



```
983 @ MSG_SYS_PSET                     "[ pset ]"                  # 207
984 @ MSG_SYS_PSET_ALT                 "pset"
985 @ MSG_SYS_SPARC_UTRAP_INSTALL      "[ sparc_utrap_install ]"   # 208
986 @ MSG_SYS_SPARC_UTRAP_INSTALL_ALT  "sparc_utrap_install"
987 @ MSG_SYS_RESOLVEPATH              "[ resolvepath ]"           # 209
988 @ MSG_SYS_RESOLVEPATH_ALT          "resolvepath"
989 @ MSG_SYS_LWP_MUTEX_TIMEDLOCK     "[ lwp_mutex_timedlock ]"   # 210
990 @ MSG_SYS_LWP_MUTEX_TIMEDLOCK_ALT "lwp_mutex_timedlock"
991 @ MSG_SYS_LWP_SEMA_TIMEDWAIT      "[ lwp_sema_timedwait ]"    # 211
992 @ MSG_SYS_LWP_SEMA_TIMEDWAIT_ALT  "lwp_sema_timedwait"
993 @ MSG_SYS_LWP_RWLOCK_SYS          "[ lwp_rwlock_sys ]"        # 212
994 @ MSG_SYS_LWP_RWLOCK_SYS_ALT      "lwp_rwlock_sys"
995 @ MSG_SYS_GETDENTS64              "[ getdents64 ]"            # 213
996 @ MSG_SYS_GETDENTS64_ALT          "getdents64"
997 @ MSG_SYS_MMAP64                  "[ mmap64 ]"                # 214
998 @ MSG_SYS_MMAP64_ALT              "mmap64"
999 @ MSG_SYS_STAT64                  "[ stat64 ]"                # 215
1000 @ MSG_SYS_STAT64_ALT              "stat64"
1001 @ MSG_SYS_LSTAT64                 "[ lstat64 ]"               # 216
1002 @ MSG_SYS_LSTAT64_ALT             "lstat64"
1003 @ MSG_SYS_FSTAT64                 "[ fstat64 ]"               # 217
1004 @ MSG_SYS_FSTAT64_ALT             "fstat64"
1005 @ MSG_SYS_STATVFS64               "[ statvfs64 ]"             # 218
1006 @ MSG_SYS_STATVFS64_ALT           "statvfs64"
1007 @ MSG_SYS_FSTATVFS64              "[ fstatvfs64 ]"            # 219
1008 @ MSG_SYS_FSTATVFS64_ALT          "fstatvfs64"
1009 @ MSG_SYS_SETRLIMIT64             "[ setrlimit64 ]"           # 220
1010 @ MSG_SYS_SETRLIMIT64_ALT         "setrlimit64"
1011 @ MSG_SYS_GETRLIMIT64             "[ getrlimit64 ]"           # 221
1012 @ MSG_SYS_GETRLIMIT64_ALT         "getrlimit64"
1013 @ MSG_SYS_PREAD64                 "[ pread64 ]"               # 222
1014 @ MSG_SYS_PREAD64_ALT             "pread64"
1015 @ MSG_SYS_PWRITE64                "[ pwrite64 ]"              # 223
1016 @ MSG_SYS_PWRITE64_ALT            "pwrite64"
1017 @ MSG_SYS_224                     "224"                       # 224(u)
1018 @ MSG_SYS_OPEN64                  "[ open64 ]"                # 225
1019 @ MSG_SYS_OPEN64_ALT              "open64"
1020 @ MSG_SYS_RPCSYS                  "[ rpcsys ]"                # 226
1021 @ MSG_SYS_RPCSYS_ALT              "rpcsys"
1022 @ MSG_SYS_ZONE                    "[ zone ]"                  # 227
1023 @ MSG_SYS_ZONE_ALT                "zone"
1024 @ MSG_SYS_AUTOFSSYS               "[ autofssys ]"             # 228
1025 @ MSG_SYS_AUTOFSSYS_ALT           "autofssys"
1026 @ MSG_SYS_GETCWD                  "[ getcwd ]"                # 229
1027 @ MSG_SYS_GETCWD_ALT              "getcwd"
1028 @ MSG_SYS_SO_SOCKET               "[ so_socket ]"             # 230
1029 @ MSG_SYS_SO_SOCKET_ALT           "so_socket"
1030 @ MSG_SYS_SO_SOCKETPAIR           "[ so_socketpair ]"         # 231
1031 @ MSG_SYS_SO_SOCKETPAIR_ALT       "so_socketpair"
1032 @ MSG_SYS_BIND                    "[ bind ]"                  # 232
1033 @ MSG_SYS_BIND_ALT                "bind"
1034 @ MSG_SYS_LISTEN                  "[ listen ]"                # 233
1035 @ MSG_SYS_LISTEN_ALT              "listen"
1036 @ MSG_SYS_ACCEPT                  "[ accept ]"                # 234
1037 @ MSG_SYS_ACCEPT_ALT              "accept"
1038 @ MSG_SYS_CONNECT                 "[ connect ]"               # 235
1039 @ MSG_SYS_CONNECT_ALT             "connect"
1040 @ MSG_SYS_SHUTDOWN                "[ shutdown ]"              # 236
1041 @ MSG_SYS_SHUTDOWN_ALT            "shutdown"
1042 @ MSG_SYS_RECV                    "[ recv ]"                  # 237
1043 @ MSG_SYS_RECV_ALT                "recv"
1044 @ MSG_SYS_RECVFROM                "[ recvfrom ]"              # 238
1045 @ MSG_SYS_RECVFROM_ALT            "recvfrom"
1046 @ MSG_SYS_RECVMSG                 "[ recvmsg ]"               # 239
1047 @ MSG_SYS_RECVMSG_ALT             "recvmsg"
1048 @ MSG_SYS_SEND                    "[ send ]"                  # 240
```

```
1049 @ MSG_SYS_SEND_ALT                    "send"
1050 @ MSG_SYS_SENDMSG                     "[ sendmsg ]"                  # 241
1051 @ MSG_SYS_SENDMSG_ALT                 "sendmsg"
1052 @ MSG_SYS_SENDTO                      "[ sendto ]"                   # 242
1053 @ MSG_SYS_SENDTO_ALT                  "sendto"
1054 @ MSG_SYS_GETPEERNAME                 "[ getpeername ]"              # 243
1055 @ MSG_SYS_GETPEERNAME_ALT             "getpeername"
1056 @ MSG_SYS_GETSOCKNAME                 "[ getsockname ]"              # 244
1057 @ MSG_SYS_GETSOCKNAME_ALT             "getsockname"
1058 @ MSG_SYS_GETSOCKOPT                  "[ getsockopt ]"               # 245
1059 @ MSG_SYS_GETSOCKOPT_ALT              "getsockopt"
1060 @ MSG_SYS_SETSOCKOPT                  "[ setsockopt ]"               # 246
1061 @ MSG_SYS_SETSOCKOPT_ALT              "setsockopt"
1062 @ MSG_SYS_SOCKCONFIG                  "[ sockconfig ]"               # 247
1063 @ MSG_SYS_SOCKCONFIG_ALT              "sockconfig"
1064 @ MSG_SYS_NTP_GETTIME                 "[ ntp_gettime ]"              # 248
1065 @ MSG_SYS_NTP_GETTIME_ALT             "ntp_gettime"
1066 @ MSG_SYS_NTP_ADJTIME                 "[ ntp_adjtime ]"              # 249
1067 @ MSG_SYS_NTP_ADJTIME_ALT             "ntp_adjtime"
1068 @ MSG_SYS_LWP_MUTEX_UNLOCK            "[ lwp_mutex_unlock ]"         # 250
1069 @ MSG_SYS_LWP_MUTEX_UNLOCK_ALT        "lwp_mutex_unlock"
1070 @ MSG_SYS_LWP_MUTEX_TRYLOCK           "[ lwp_mutex_trylock ]"        # 251
1071 @ MSG_SYS_LWP_MUTEX_TRYLOCK_ALT       "lwp_mutex_trylock"
1072 @ MSG_SYS_LWP_MUTEX_REGISTER          "[ lwp_mutex_register ]"       # 252
1073 @ MSG_SYS_LWP_MUTEX_REGISTER_ALT      "lwp_mutex_register"
1074 @ MSG_SYS_CLADM                       "[ cladm ]"                    # 253
1075 @ MSG_SYS_CLADM_ALT                   "cladm"
1076 @ MSG_SYS_UUCOPY                      "[ uucopy ]"                   # 254
1077 @ MSG_SYS_UUCOPY_ALT                  "uucopy"
1078 @ MSG_SYS_UMOUNT2                     "[ umount2 ]"                  # 255
1079 @ MSG_SYS_UMOUNT2_ALT                 "umount2"

1081 @ MSG_PR_O_RDONLY                     "O_RDONLY"
1082 @ MSG_PR_O_WRONLY                     "O_WRONLY"
1083 @ MSG_PR_O_RDWR                       "O_RDWR"
1084 @ MSG_PR_O_SEARCH                     "O_SEARCH"
1085 @ MSG_PR_O_EXEC                       "O_EXEC"
1086 @ MSG_PR_O_NDELAY                     "O_NDELAY"
1087 @ MSG_PR_O_NONBLOCK                   "O_NONBLOCK"
1088 @ MSG_PR_O_APPEND                     "O_APPEND"
1089 @ MSG_PR_O_SYNC                       "O_SYNC"
1090 @ MSG_PR_O_DSYNC                      "O_DSYNC"
1091 @ MSG_PR_O_RSYNC                      "O_RSYNC"
1092 @ MSG_PR_O_CREAT                      "O_CREAT"
1093 @ MSG_PR_O_TRUNC                      "O_TRUNC"
1094 @ MSG_PR_O_EXCL                       "O_EXCL"
1095 @ MSG_PR_O_NOCTTY                     "O_NOCTTY"
1096 @ MSG_PR_O_LARGEFILE                  "O_LARGEFILE"
1097 @ MSG_PR_O_XATTR                      "O_XATTR"
1098 @ MSG_PR_O_NOFOLLOW                   "O_NOFOLLOW"
1099 @ MSG_PR_O_NOLINKS                    "O_NOLINKS"

1101 @ MSG_S_IFIFO                         "S_IFIFO"
1102 @ MSG_S_IFCHR                         "S_IFCHR"
1103 @ MSG_S_IFDIR                         "S_IFDIR"
1104 @ MSG_S_IFNAM                         "S_IFNAM"
1105 @ MSG_S_IFBLK                         "S_IFBLK"
1106 @ MSG_S_IFREG                         "S_IFREG"
1107 @ MSG_S_IFLNK                         "S_IFLNK"
1108 @ MSG_S_IFSOCK                        "S_IFSOCK"
1109 @ MSG_S_IFDOOR                        "S_IFDOOR"
1110 @ MSG_S_IFPORT                        "S_IFPORT"
1111 @ MSG_S_ISUID                         "S_ISUID"
1112 @ MSG_S_ISGID                         "S_ISGID"
1113 @ MSG_S_ISVTX                         "S_ISVTX"
1114 @ MSG_S_IRUSR                         "S_IRUSR"
```



```
1115 @ MSG_S_IWUSR                         "S_IWUSR"
1116 @ MSG_S_IXUSR                         "S_IXUSR"
1117 @ MSG_S_IRGRP                         "S_IRGRP"
1118 @ MSG_S_IWGRP                         "S_IWGRP"
1119 @ MSG_S_IXGRP                         "S_IXGRP"
1120 @ MSG_S_IROTH                         "S_IROTH"
1121 @ MSG_S_IWOTH                         "S_IWOTH"
1122 @ MSG_S_IXOTH                         "S_IXOTH"

1124 @ MSG_GBL_ZERO          "0"

1126 @ MSG_FMT_INT           "%d"
1127 @ MSG_FMT_WORD          "%u"
1128 @ MSG_FMT_HEXINT        "%#x"
```

```
**********************************************************
   34856 Wed May 27 19:49:03 2015
new/usr/src/cmd/sgs/libconv/common/dynamic.c
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
______unchanged_portion_omitted_
```

```
 419 const conv_ds_t **
 420 conv_dyn_tag_strings(conv_iter_osabi_t osabi, Half mach,
 421     Conv_fmt_flags_t fmt_flags)
 422 {
 423         /*
 424          * Maximum # of items that can be in the returned array. Size this
 425          * by counting the maximum depth in the switch statement that fills
 426          * retarr at the end of this function.
 427          */
 428 #define MAX_RET 12

 430         /*
 431          * Generic dynamic tags:
 432          * -    Note hole between DT_FLAGS and DT_PREINIT_ARRAY (tag 32).
 433          *      We use a 0, which is the signal for "not defined".
 434          * -    This range has alternative names for dump, requiring an
 435          *      additional array.
 436          */
 437         static const Msg        tags_null_cf[] = {
 438                 MSG_DT_NULL_CF,                 MSG_DT_NEEDED_CF,
 439                 MSG_DT_PLTRELSZ_CF,             MSG_DT_PLTGOT_CF,
 440                 MSG_DT_HASH_CF,                 MSG_DT_STRTAB_CF,
 441                 MSG_DT_SYMTAB_CF,               MSG_DT_RELA_CF,
 442                 MSG_DT_RELASZ_CF,               MSG_DT_RELAENT_CF,
 443                 MSG_DT_STRSZ_CF,                MSG_DT_SYMENT_CF,
 444                 MSG_DT_INIT_CF,                 MSG_DT_FINI_CF,
 445                 MSG_DT_SONAME_CF,               MSG_DT_RPATH_CF,
 446                 MSG_DT_SYMBOLIC_CF,             MSG_DT_REL_CF,
 447                 MSG_DT_RELSZ_CF,                MSG_DT_RELENT_CF,
 448                 MSG_DT_PLTREL_CF,               MSG_DT_DEBUG_CF,
 449                 MSG_DT_TEXTREL_CF,              MSG_DT_JMPREL_CF,
 450                 MSG_DT_BIND_NOW_CF,             MSG_DT_INIT_ARRAY_CF,
 451                 MSG_DT_FINI_ARRAY_CF,           MSG_DT_INIT_ARRAYSZ_CF,
 452                 MSG_DT_FINI_ARRAYSZ_CF,         MSG_DT_RUNPATH_CF,
 453                 MSG_DT_FLAGS_CF,                0,
 454                 MSG_DT_PREINIT_ARRAY_CF,        MSG_DT_PREINIT_ARRAYSZ_CF
 455         };
 456         static const Msg        tags_null_cfnp[] = {
 457                 MSG_DT_NULL_CFNP,               MSG_DT_NEEDED_CFNP,
 458                 MSG_DT_PLTRELSZ_CFNP,           MSG_DT_PLTGOT_CFNP,
 459                 MSG_DT_HASH_CFNP,               MSG_DT_STRTAB_CFNP,
 460                 MSG_DT_SYMTAB_CFNP,             MSG_DT_RELA_CFNP,
 461                 MSG_DT_RELASZ_CFNP,             MSG_DT_RELAENT_CFNP,
 462                 MSG_DT_STRSZ_CFNP,              MSG_DT_SYMENT_CFNP,
 463                 MSG_DT_INIT_CFNP,               MSG_DT_FINI_CFNP,
 464                 MSG_DT_SONAME_CFNP,             MSG_DT_RPATH_CFNP,
 465                 MSG_DT_SYMBOLIC_CFNP,           MSG_DT_REL_CFNP,
 466                 MSG_DT_RELSZ_CFNP,              MSG_DT_RELENT_CFNP,
 467                 MSG_DT_PLTREL_CFNP,             MSG_DT_DEBUG_CFNP,
 468                 MSG_DT_TEXTREL_CFNP,            MSG_DT_JMPREL_CFNP,
 469                 MSG_DT_BIND_NOW_CFNP,           MSG_DT_INIT_ARRAY_CFNP,
 470                 MSG_DT_FINI_ARRAY_CFNP,         MSG_DT_INIT_ARRAYSZ_CFNP,
 471                 MSG_DT_FINI_ARRAYSZ_CFNP,       MSG_DT_RUNPATH_CFNP,
 472                 MSG_DT_FLAGS_CFNP,              0,
 473                 MSG_DT_PREINIT_ARRAY_CFNP,      MSG_DT_PREINIT_ARRAYSZ_CFNP
 474         };
```



```
 475         static const Msg        tags_null_nf[] = {
 476                 MSG_DT_NULL_NF,                 MSG_DT_NEEDED_NF,
 477                 MSG_DT_PLTRELSZ_NF,             MSG_DT_PLTGOT_NF,
 478                 MSG_DT_HASH_NF,                 MSG_DT_STRTAB_NF,
 479                 MSG_DT_SYMTAB_NF,               MSG_DT_RELA_NF,
 480                 MSG_DT_RELASZ_NF,               MSG_DT_RELAENT_NF,
 481                 MSG_DT_STRSZ_NF,                MSG_DT_SYMENT_NF,
 482                 MSG_DT_INIT_NF,                 MSG_DT_FINI_NF,
 483                 MSG_DT_SONAME_NF,               MSG_DT_RPATH_NF,
 484                 MSG_DT_SYMBOLIC_NF,             MSG_DT_REL_NF,
 485                 MSG_DT_RELSZ_NF,                MSG_DT_RELENT_NF,
 486                 MSG_DT_PLTREL_NF,               MSG_DT_DEBUG_NF,
 487                 MSG_DT_TEXTREL_NF,              MSG_DT_JMPREL_NF,
 488                 MSG_DT_BIND_NOW_NF,             MSG_DT_INIT_ARRAY_NF,
 489                 MSG_DT_FINI_ARRAY_NF,           MSG_DT_INIT_ARRAYSZ_NF,
 490                 MSG_DT_FINI_ARRAYSZ_NF,         MSG_DT_RUNPATH_NF,
 491                 MSG_DT_FLAGS_NF,                0,
 492                 MSG_DT_PREINIT_ARRAY_NF,        MSG_DT_PREINIT_ARRAYSZ_NF
 493         };
 494         static const Msg        tags_null_dmp[] = {
 495                 MSG_DT_NULL_CFNP,               MSG_DT_NEEDED_CFNP,
 496                 MSG_DT_PLTRELSZ_DMP,            MSG_DT_PLTGOT_CFNP,
 497                 MSG_DT_HASH_CFNP,               MSG_DT_STRTAB_CFNP,
 498                 MSG_DT_SYMTAB_CFNP,             MSG_DT_RELA_CFNP,
 499                 MSG_DT_RELASZ_CFNP,             MSG_DT_RELAENT_CFNP,
 500                 MSG_DT_STRSZ_CFNP,              MSG_DT_SYMENT_CFNP,
 501                 MSG_DT_INIT_CFNP,               MSG_DT_FINI_CFNP,
 502                 MSG_DT_SONAME_CFNP,             MSG_DT_RPATH_CFNP,
 503                 MSG_DT_SYMBOLIC_DMP,            MSG_DT_REL_CFNP,
 504                 MSG_DT_RELSZ_CFNP,              MSG_DT_RELENT_CFNP,
 505                 MSG_DT_PLTREL_CFNP,             MSG_DT_DEBUG_CFNP,
 506                 MSG_DT_TEXTREL_CFNP,            MSG_DT_JMPREL_CFNP,
 507                 MSG_DT_BIND_NOW_CFNP,           MSG_DT_INIT_ARRAY_CFNP,
 508                 MSG_DT_FINI_ARRAY_CFNP,         MSG_DT_INIT_ARRAYSZ_CFNP,
 509                 MSG_DT_FINI_ARRAYSZ_CFNP,       MSG_DT_RUNPATH_CFNP,
 510                 MSG_DT_FLAGS_CFNP,              0,
 511                 MSG_DT_PREINIT_ARRAY_CFNP,      MSG_DT_PREINIT_ARRAYSZ_CFNP
 512         };
 513         static const conv_ds_msg_t ds_null_cf = {
 514             CONV_DS_MSG_INIT(DT_NULL, tags_null_cf) };
 515         static const conv_ds_msg_t ds_null_cfnp = {
 516             CONV_DS_MSG_INIT(DT_NULL, tags_null_cfnp) };
 517         static const conv_ds_msg_t ds_null_nf = {
 518             CONV_DS_MSG_INIT(DT_NULL, tags_null_nf) };
 519         static const conv_ds_msg_t ds_null_dmp = {
 520             CONV_DS_MSG_INIT(DT_NULL, tags_null_dmp) };

 522         /*
 523          * DT_SPARC_REGISTER was originally assigned 0x7000001. It is processor
 524          * specific, and should have been in the range DT_LOPROC-DT_HIPROC
 525          * instead of here. When the error was fixed,
 526          * DT_DEPRECATED_SPARC_REGISTER was created to maintain backward
 527          * compatability.
 528          */
 529         static const Msg        tags_sdreg_cf[] = {
 530             MSG_DT_DEP_SPARC_REG_CF };
 531         static const Msg        tags_sdreg_cfnp[] = {
 532             MSG_DT_DEP_SPARC_REG_CFNP };
 533         static const Msg        tags_sdreg_nf[] = {
 534             MSG_DT_DEP_SPARC_REG_NF };

 536         static const conv_ds_msg_t ds_sdreg_cf = {
 537             CONV_DS_MSG_INIT(DT_DEPRECATED_SPARC_REGISTER, tags_sdreg_cf) };
 538         static const conv_ds_msg_t ds_sdreg_cfnp = {
 539             CONV_DS_MSG_INIT(DT_DEPRECATED_SPARC_REGISTER, tags_sdreg_cfnp) };
 540         static const conv_ds_msg_t ds_sdreg_nf = {
```

```
541             CONV_DS_MSG_INIT(DT_DEPRECATED_SPARC_REGISTER, tags_sdreg_nf) };


544         /*
545          * SUNW: DT_LOOS -> DT_HIOS range. Note holes between DT_SUNW_TLSSORTSZ,
546          * DT_SUNW_STRPAD, and DT_SUNW_LDMACH. We handle the outliers
547          * separately below as single values.
548          */
549         static const Msg        tags_sunw_auxiliary_cf[] = {
550                 MSG_DT_SUNW_AUXILIARY_CF,       MSG_DT_SUNW_RTLDINF_CF,
551                 MSG_DT_SUNW_FILTER_CF,          MSG_DT_SUNW_CAP_CF,
552                 MSG_DT_SUNW_SYMTAB_CF,          MSG_DT_SUNW_SYMSZ_CF,
553                 MSG_DT_SUNW_SORTENT_CF,         MSG_DT_SUNW_SYMSORT_CF,
554                 MSG_DT_SUNW_SYMSORTSZ_CF,       MSG_DT_SUNW_TLSSORT_CF,
555                 MSG_DT_SUNW_TLSSORTSZ_CF,       MSG_DT_SUNW_CAPINFO_CF,
556                 MSG_DT_SUNW_STRPAD_CF,          MSG_DT_SUNW_CAPCHAIN_CF,
557                 MSG_DT_SUNW_LDMACH_CF,          0,
558                 MSG_DT_SUNW_CAPCHAINENT_CF,     0,
559                 MSG_DT_SUNW_CAPCHAINSZ_CF,      0,
560                 0,                              0,
561                 MSG_DT_SUNW_ASLR_CF
559                 MSG_DT_SUNW_CAPCHAINSZ_CF
562         };
563         static const Msg        tags_sunw_auxiliary_cfnp[] = {
564                 MSG_DT_SUNW_AUXILIARY_CFNP,     MSG_DT_SUNW_RTLDINF_CFNP,
565                 MSG_DT_SUNW_FILTER_CFNP,        MSG_DT_SUNW_CAP_CFNP,
566                 MSG_DT_SUNW_SYMTAB_CFNP,        MSG_DT_SUNW_SYMSZ_CFNP,
567                 MSG_DT_SUNW_SORTENT_CFNP,       MSG_DT_SUNW_SYMSORT_CFNP,
568                 MSG_DT_SUNW_SYMSORTSZ_CFNP,     MSG_DT_SUNW_TLSSORT_CFNP,
569                 MSG_DT_SUNW_TLSSORTSZ_CFNP,     MSG_DT_SUNW_CAPINFO_CFNP,
570                 MSG_DT_SUNW_STRPAD_CFNP,        MSG_DT_SUNW_CAPCHAIN_CFNP,
571                 MSG_DT_SUNW_LDMACH_CFNP,        0,
572                 MSG_DT_SUNW_CAPCHAINENT_CFNP,   0,
573                 MSG_DT_SUNW_CAPCHAINSZ_CFNP,    0,
574                 0,                              0,
575                 MSG_DT_SUNW_ASLR_CFNP
571                 MSG_DT_SUNW_CAPCHAINSZ_CFNP
576         };
577         static const Msg        tags_sunw_auxiliary_nf[] = {
578                 MSG_DT_SUNW_AUXILIARY_NF,       MSG_DT_SUNW_RTLDINF_NF,
579                 MSG_DT_SUNW_FILTER_NF,          MSG_DT_SUNW_CAP_NF,
580                 MSG_DT_SUNW_SYMTAB_NF,          MSG_DT_SUNW_SYMSZ_NF,
581                 MSG_DT_SUNW_SORTENT_NF,         MSG_DT_SUNW_SYMSORT_NF,
582                 MSG_DT_SUNW_SYMSORTSZ_NF,       MSG_DT_SUNW_TLSSORT_NF,
583                 MSG_DT_SUNW_TLSSORTSZ_NF,       MSG_DT_SUNW_CAPINFO_NF,
584                 MSG_DT_SUNW_STRPAD_NF,          MSG_DT_SUNW_CAPCHAIN_NF,
585                 MSG_DT_SUNW_LDMACH_NF,          0,
586                 MSG_DT_SUNW_CAPCHAINENT_NF,     0,
587                 MSG_DT_SUNW_CAPCHAINSZ_NF,      0,
588                 0,                              0,
589                 MSG_DT_SUNW_ASLR_NF
583                 MSG_DT_SUNW_CAPCHAINSZ_NF
590         };
591         static const conv_ds_msg_t ds_sunw_auxiliary_cf = {
592             CONV_DS_MSG_INIT(DT_SUNW_AUXILIARY, tags_sunw_auxiliary_cf) };
593         static const conv_ds_msg_t ds_sunw_auxiliary_cfnp = {
594             CONV_DS_MSG_INIT(DT_SUNW_AUXILIARY, tags_sunw_auxiliary_cfnp) };
595         static const conv_ds_msg_t ds_sunw_auxiliary_nf = {
596             CONV_DS_MSG_INIT(DT_SUNW_AUXILIARY, tags_sunw_auxiliary_nf) };

598         /*
599          * GNU: (In DT_VALRNGLO section) DT_GNU_PRELINKED - DT_GNU_LIBLISTSZ
600          */
601         static const Msg        tags_gnu_prelinked_cf[] = {
602                 MSG_DT_GNU_PRELINKED_CF,        MSG_DT_GNU_CONFLICTSZ_CF,
603                 MSG_DT_GNU_LIBLISTSZ_CF
```



```
604         };
605         static const Msg        tags_gnu_prelinked_cfnp[] = {
606                 MSG_DT_GNU_PRELINKED_CFNP,      MSG_DT_GNU_CONFLICTSZ_CFNP,
607                 MSG_DT_GNU_LIBLISTSZ_CFNP
608         };
609         static const Msg        tags_gnu_prelinked_nf[] = {
610                 MSG_DT_GNU_PRELINKED_NF,        MSG_DT_GNU_CONFLICTSZ_NF,
611                 MSG_DT_GNU_LIBLISTSZ_NF
612         };
613         static const conv_ds_msg_t ds_gnu_prelinked_cf = {
614             CONV_DS_MSG_INIT(DT_GNU_PRELINKED, tags_gnu_prelinked_cf) };
615         static const conv_ds_msg_t ds_gnu_prelinked_cfnp = {
616             CONV_DS_MSG_INIT(DT_GNU_PRELINKED, tags_gnu_prelinked_cfnp) };
617         static const conv_ds_msg_t ds_gnu_prelinked_nf = {
618             CONV_DS_MSG_INIT(DT_GNU_PRELINKED, tags_gnu_prelinked_nf) };

620         /*
621          * SUNW: DT_VALRNGLO - DT_VALRNGHI range.
622          */
623         static const Msg        tags_checksum_cf[] = {
624                 MSG_DT_CHECKSUM_CF,             MSG_DT_PLTPADSZ_CF,
625                 MSG_DT_MOVEENT_CF,              MSG_DT_MOVESZ_CF,
626                 MSG_DT_FEATURE_1_CF,            MSG_DT_POSFLAG_1_CF,
627                 MSG_DT_SYMINSZ_CF,              MSG_DT_SYMINENT_CF
628         };
629         static const Msg        tags_checksum_cfnp[] = {
630                 MSG_DT_CHECKSUM_CFNP,           MSG_DT_PLTPADSZ_CFNP,
631                 MSG_DT_MOVEENT_CFNP,            MSG_DT_MOVESZ_CFNP,
632                 MSG_DT_FEATURE_1_CFNP,          MSG_DT_POSFLAG_1_CFNP,
633                 MSG_DT_SYMINSZ_CFNP,            MSG_DT_SYMINENT_CFNP
634         };
635         static const Msg        tags_checksum_nf[] = {
636                 MSG_DT_CHECKSUM_NF,             MSG_DT_PLTPADSZ_NF,
637                 MSG_DT_MOVEENT_NF,              MSG_DT_MOVESZ_NF,
638                 MSG_DT_FEATURE_1_NF,            MSG_DT_POSFLAG_1_NF,
639                 MSG_DT_SYMINSZ_NF,              MSG_DT_SYMINENT_NF
640         };
641         static const conv_ds_msg_t ds_checksum_cf = {
642             CONV_DS_MSG_INIT(DT_CHECKSUM, tags_checksum_cf) };
643         static const conv_ds_msg_t ds_checksum_cfnp = {
644             CONV_DS_MSG_INIT(DT_CHECKSUM, tags_checksum_cfnp) };
645         static const conv_ds_msg_t ds_checksum_nf = {
646             CONV_DS_MSG_INIT(DT_CHECKSUM, tags_checksum_nf) };

648         /*
649          * GNU: (In DT_ADDRRNGLO section) DT_GNU_HASH - DT_GNU_LIBLIST
650          */
651         static const Msg        tags_gnu_hash_cf[] = {
652                 MSG_DT_GNU_HASH_CF,             MSG_DT_TLSDESC_PLT_CF,
653                 MSG_DT_TLSDESC_GOT_CF,          MSG_DT_GNU_CONFLICT_CF,
654                 MSG_DT_GNU_LIBLIST_CF
655         };
656         static const Msg        tags_gnu_hash_cfnp[] = {
657                 MSG_DT_GNU_HASH_CFNP,           MSG_DT_TLSDESC_PLT_CFNP,
658                 MSG_DT_TLSDESC_GOT_CFNP,        MSG_DT_GNU_CONFLICT_CFNP,
659                 MSG_DT_GNU_LIBLIST_CFNP
660         };
661         static const Msg        tags_gnu_hash_nf[] = {
662                 MSG_DT_GNU_HASH_NF,             MSG_DT_TLSDESC_PLT_NF,
663                 MSG_DT_TLSDESC_GOT_NF,          MSG_DT_GNU_CONFLICT_NF,
664                 MSG_DT_GNU_LIBLIST_NF
665         };
666         static const conv_ds_msg_t ds_gnu_hash_cf = {
667             CONV_DS_MSG_INIT(DT_GNU_HASH, tags_gnu_hash_cf) };
668         static const conv_ds_msg_t ds_gnu_hash_cfnp = {
669             CONV_DS_MSG_INIT(DT_GNU_HASH, tags_gnu_hash_cfnp) };
```

```
670         static const conv_ds_msg_t ds_gnu_hash_nf = {
671             CONV_DS_MSG_INIT(DT_GNU_HASH, tags_gnu_hash_nf) };

673         /*
674          * SUNW: DT_ADDRRNGLO - DT_ADDRRNGHI range.
675          */
676         static const Msg        tags_config_cf[] = {
677                 MSG_DT_CONFIG_CF,               MSG_DT_DEPAUDIT_CF,
678                 MSG_DT_AUDIT_CF,                MSG_DT_PLTPAD_CF,
679                 MSG_DT_MOVETAB_CF,              MSG_DT_SYMINFO_CF
680         };
681         static const Msg        tags_config_cfnp[] = {
682                 MSG_DT_CONFIG_CFNP,             MSG_DT_DEPAUDIT_CFNP,
683                 MSG_DT_AUDIT_CFNP,              MSG_DT_PLTPAD_CFNP,
684                 MSG_DT_MOVETAB_CFNP,            MSG_DT_SYMINFO_CFNP
685         };
686         static const Msg        tags_config_nf[] = {
687                 MSG_DT_CONFIG_NF,               MSG_DT_DEPAUDIT_NF,
688                 MSG_DT_AUDIT_NF,                MSG_DT_PLTPAD_NF,
689                 MSG_DT_MOVETAB_NF,              MSG_DT_SYMINFO_NF
690         };
691         static const conv_ds_msg_t ds_config_cf = {
692             CONV_DS_MSG_INIT(DT_CONFIG, tags_config_cf) };
693         static const conv_ds_msg_t ds_config_cfnp = {
694             CONV_DS_MSG_INIT(DT_CONFIG, tags_config_cfnp) };
695         static const conv_ds_msg_t ds_config_nf = {
696             CONV_DS_MSG_INIT(DT_CONFIG, tags_config_nf) };

698         /*
699          * SUNW: generic range. Note hole between DT_VERSYM and DT_RELACOUNT.
700          */
701         static const Msg        tags_versym_cf[] = { MSG_DT_VERSYM_CF };
702         static const Msg        tags_versym_cfnp[] = { MSG_DT_VERSYM_CFNP };
703         static const Msg        tags_versym_nf[] = { MSG_DT_VERSYM_NF };
704         static const conv_ds_msg_t ds_versym_cf = {
705             CONV_DS_MSG_INIT(DT_VERSYM, tags_versym_cf) };
706         static const conv_ds_msg_t ds_versym_cfnp = {
707             CONV_DS_MSG_INIT(DT_VERSYM, tags_versym_cfnp) };
708         static const conv_ds_msg_t ds_versym_nf = {
709             CONV_DS_MSG_INIT(DT_VERSYM, tags_versym_nf) };

711         static const Msg        tags_relacount_cf[] = {
712                 MSG_DT_RELACOUNT_CF,            MSG_DT_RELCOUNT_CF,
713                 MSG_DT_FLAGS_1_CF,              MSG_DT_VERDEF_CF,
714                 MSG_DT_VERDEFNUM_CF,            MSG_DT_VERNEED_CF,
715                 MSG_DT_VERNEEDNUM_CF
716         };
717         static const Msg        tags_relacount_cfnp[] = {
718                 MSG_DT_RELACOUNT_CFNP,          MSG_DT_RELCOUNT_CFNP,
719                 MSG_DT_FLAGS_1_CFNP,            MSG_DT_VERDEF_CFNP,
720                 MSG_DT_VERDEFNUM_CFNP,          MSG_DT_VERNEED_CFNP,
721                 MSG_DT_VERNEEDNUM_CFNP
722         };
723         static const Msg        tags_relacount_nf[] = {
724                 MSG_DT_RELACOUNT_NF,            MSG_DT_RELCOUNT_NF,
725                 MSG_DT_FLAGS_1_NF,              MSG_DT_VERDEF_NF,
726                 MSG_DT_VERDEFNUM_NF,            MSG_DT_VERNEED_NF,
727                 MSG_DT_VERNEEDNUM_NF
728         };
729         static const conv_ds_msg_t ds_relacount_cf = {
730             CONV_DS_MSG_INIT(DT_RELACOUNT, tags_relacount_cf) };
731         static const conv_ds_msg_t ds_relacount_cfnp = {
732             CONV_DS_MSG_INIT(DT_RELACOUNT, tags_relacount_cfnp) };
733         static const conv_ds_msg_t ds_relacount_nf = {
734             CONV_DS_MSG_INIT(DT_RELACOUNT, tags_relacount_nf) };
```



```
736         /*
737          * DT_LOPROC - DT_HIPROC range: solaris/sparc-only
738          */
739         static const Msg tags_sparc_reg_cf[] = { MSG_DT_SPARC_REGISTER_CF };
740         static const Msg tags_sparc_reg_cfnp[] = { MSG_DT_SPARC_REGISTER_CFNP };
741         static const Msg tags_sparc_reg_nf[] = { MSG_DT_SPARC_REGISTER_NF };
742         static const Msg tags_sparc_reg_dmp[] = { MSG_DT_SPARC_REGISTER_DMP };
743         static const conv_ds_msg_t ds_sparc_reg_cf = {
744             CONV_DS_MSG_INIT(DT_SPARC_REGISTER, tags_sparc_reg_cf) };
745         static const conv_ds_msg_t ds_sparc_reg_cfnp = {
746             CONV_DS_MSG_INIT(DT_SPARC_REGISTER, tags_sparc_reg_cfnp) };
747         static const conv_ds_msg_t ds_sparc_reg_nf = {
748             CONV_DS_MSG_INIT(DT_SPARC_REGISTER, tags_sparc_reg_nf) };
749         static const conv_ds_msg_t ds_sparc_reg_dmp = {
750             CONV_DS_MSG_INIT(DT_SPARC_REGISTER, tags_sparc_reg_dmp) };

752         /*
753          * DT_LOPROC - DT_HIPROC range: Solaris osabi, all hardware
754          */
755         static const Msg        tags_auxiliary_cf[] = {
756                 MSG_DT_AUXILIARY_CF,    MSG_DT_USED_CF,
757                 MSG_DT_FILTER_CF
758         };
759         static const Msg        tags_auxiliary_cfnp[] = {
760                 MSG_DT_AUXILIARY_CFNP,  MSG_DT_USED_CFNP,
761                 MSG_DT_FILTER_CFNP
762         };
763         static const Msg        tags_auxiliary_nf[] = {
764                 MSG_DT_AUXILIARY_NF,    MSG_DT_USED_NF,
765                 MSG_DT_FILTER_NF
766         };
767         static const conv_ds_msg_t ds_auxiliary_cf = {
768             CONV_DS_MSG_INIT(DT_AUXILIARY, tags_auxiliary_cf) };
769         static const conv_ds_msg_t ds_auxiliary_cfnp = {
770             CONV_DS_MSG_INIT(DT_AUXILIARY, tags_auxiliary_cfnp) };
771         static const conv_ds_msg_t ds_auxiliary_nf = {
772             CONV_DS_MSG_INIT(DT_AUXILIARY, tags_auxiliary_nf) };


775         static const conv_ds_t  *retarr[MAX_RET];

777         int     ndx = 0;
778         int     fmt_osabi = CONV_TYPE_FMT_ALT(fmt_flags);
779         int     mach_sparc, osabi_solaris, osabi_linux;



783         osabi_solaris = (osabi == ELFOSABI_NONE) ||
784             (osabi == ELFOSABI_SOLARIS) || (osabi == CONV_OSABI_ALL);
785         osabi_linux = (osabi == ELFOSABI_LINUX) || (osabi == CONV_OSABI_ALL);
786         mach_sparc = (mach == EM_SPARC) || (mach == EM_SPARCV9) ||
787             (mach == EM_SPARC32PLUS) || (mach == CONV_MACH_ALL);

789         /*
790          * Fill in retarr with the descriptors for the messages that
791          * apply to the current osabi. Note that we order these items such
792          * that the more common are placed at the beginning, and the less
793          * likely at the end. This should speed the common case.
794          *
795          * Note that the CFNP and DMP styles are very similar, so they
796          * are combined in 'default', and fmt_osabi is consulted when there
797          * are differences.
798          */
799         switch (fmt_osabi) {
800         case CONV_FMT_ALT_CF:
801                 retarr[ndx++] = CONV_DS_ADDR(ds_null_cf);
```

```
802                 if (osabi_solaris)
803                         retarr[ndx++] = CONV_DS_ADDR(ds_sunw_auxiliary_cf);
804                 retarr[ndx++] = CONV_DS_ADDR(ds_checksum_cf);
805                 retarr[ndx++] = CONV_DS_ADDR(ds_config_cf);
806                 retarr[ndx++] = CONV_DS_ADDR(ds_versym_cf);
807                 retarr[ndx++] = CONV_DS_ADDR(ds_relacount_cf);
808                 if (osabi_solaris) {
809                         retarr[ndx++] = CONV_DS_ADDR(ds_auxiliary_cf);
810                         if (mach_sparc) {
811                                 retarr[ndx++] = CONV_DS_ADDR(ds_sparc_reg_cf);
812                                 retarr[ndx++] = CONV_DS_ADDR(ds_sdreg_cf);
813                         }
814                 }
815                 if (osabi_linux) {
816                         retarr[ndx++] = CONV_DS_ADDR(ds_gnu_prelinked_cf);
817                         retarr[ndx++] = CONV_DS_ADDR(ds_gnu_hash_cf);
818                 }
819                 break;

821         case CONV_FMT_ALT_NF:
822                 retarr[ndx++] = CONV_DS_ADDR(ds_null_nf);
823                 if (osabi_solaris)
824                         retarr[ndx++] = CONV_DS_ADDR(ds_sunw_auxiliary_nf);
825                 retarr[ndx++] = CONV_DS_ADDR(ds_checksum_nf);
826                 retarr[ndx++] = CONV_DS_ADDR(ds_config_nf);
827                 retarr[ndx++] = CONV_DS_ADDR(ds_versym_nf);
828                 retarr[ndx++] = CONV_DS_ADDR(ds_relacount_nf);
829                 if (osabi_solaris) {
830                         retarr[ndx++] = CONV_DS_ADDR(ds_auxiliary_nf);
831                         if (mach_sparc) {
832                                 retarr[ndx++] = CONV_DS_ADDR(ds_sparc_reg_nf);
833                                 retarr[ndx++] = CONV_DS_ADDR(ds_sdreg_nf);
834                         }
835                 }
836                 if (osabi_linux) {
837                         retarr[ndx++] = CONV_DS_ADDR(ds_gnu_prelinked_nf);
838                         retarr[ndx++] = CONV_DS_ADDR(ds_gnu_hash_nf);
839                 }
840                 break;
841         default:
842                 /*
843                  * The default style for the generic range is CFNP,
844                  * while dump has a couple of different strings.
845                  */

847                 retarr[ndx++] = (fmt_osabi == CONV_FMT_ALT_DUMP) ?
848                     CONV_DS_ADDR(ds_null_dmp) : CONV_DS_ADDR(ds_null_cfnp);
849                 if (osabi_solaris)
850                         retarr[ndx++] = CONV_DS_ADDR(ds_sunw_auxiliary_cfnp);
851                 retarr[ndx++] = CONV_DS_ADDR(ds_checksum_cfnp);
852                 retarr[ndx++] = CONV_DS_ADDR(ds_config_cfnp);
853                 retarr[ndx++] = CONV_DS_ADDR(ds_versym_cfnp);
854                 retarr[ndx++] = CONV_DS_ADDR(ds_relacount_cfnp);
855                 if (osabi_solaris) {
856                         retarr[ndx++] = CONV_DS_ADDR(ds_auxiliary_cfnp);
857                         if (mach_sparc) {
858                                 /*
859                                  * The default style for DT_SPARC_REGISTER
860                                  * is the dump style, which omits the 'SPARC_'.
861                                  * CFNP keeps the prefix.
862                                  */
863                                 retarr[ndx++] =
864                                     (fmt_osabi == CONV_FMT_ALT_CFNP) ?
865                                     CONV_DS_ADDR(ds_sparc_reg_cfnp) :
866                                     CONV_DS_ADDR(ds_sparc_reg_dmp);
867                                 retarr[ndx++] = CONV_DS_ADDR(ds_sdreg_cfnp);
```



```
868                         }
869                 }
870                 if (osabi_linux) {
871                         retarr[ndx++] = CONV_DS_ADDR(ds_gnu_prelinked_cfnp);
872                         retarr[ndx++] = CONV_DS_ADDR(ds_gnu_hash_cfnp);
873                 }
874                 break;
875         }

877         retarr[ndx++] = NULL;
878         assert(ndx <= MAX_RET);
879         return (retarr);
880 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   16568 Wed May 27 19:49:03 2015
new/usr/src/cmd/sgs/libconv/common/dynamic.msg
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#

#
# Copyright (c) 1995, 2010, Oracle and/or its affiliates. All rights reserved.
#

@ MSG_DT_NULL_CF              "DT_NULL"                 # 0
@ MSG_DT_NULL_CFNP                    "NULL"
@ MSG_DT_NULL_NF                      "null"
@ MSG_DT_NEEDED_CF            "DT_NEEDED"               # 1
@ MSG_DT_NEEDED_CFNP                  "NEEDED"
@ MSG_DT_NEEDED_NF                    "needed"
@ MSG_DT_PLTRELSZ_CF          "DT_PLTRELSZ"             # 2
@ MSG_DT_PLTRELSZ_CFNP                "PLTRELSZ"
@ MSG_DT_PLTRELSZ_NF                  "pltrelsz"
@ MSG_DT_PLTRELSZ_DMP                 "PLTSZ"
@ MSG_DT_PLTGOT_CF            "DT_PLTGOT"               # 3
@ MSG_DT_PLTGOT_CFNP                  "PLTGOT"
@ MSG_DT_PLTGOT_NF                    "pltgot"
@ MSG_DT_HASH_CF              "DT_HASH"                 # 4
@ MSG_DT_HASH_CFNP                    "HASH"
@ MSG_DT_HASH_NF                      "hash"
@ MSG_DT_STRTAB_CF            "DT_STRTAB"               # 5
@ MSG_DT_STRTAB_CFNP                  "STRTAB"
@ MSG_DT_STRTAB_NF                    "strtab"
@ MSG_DT_SYMTAB_CF            "DT_SYMTAB"               # 6
@ MSG_DT_SYMTAB_CFNP                  "SYMTAB"
@ MSG_DT_SYMTAB_NF                    "symtab"
@ MSG_DT_RELA_CF              "DT_RELA"                 # 7
@ MSG_DT_RELA_CFNP                    "RELA"
@ MSG_DT_RELA_NF                      "rela"
@ MSG_DT_RELASZ_CF            "DT_RELASZ"               # 8
@ MSG_DT_RELASZ_CFNP                  "RELASZ"
@ MSG_DT_RELASZ_NF                    "relasz"
@ MSG_DT_RELAENT_CF           "DT_RELAENT"              # 9
@ MSG_DT_RELAENT_CFNP                 "RELAENT"
@ MSG_DT_RELAENT_NF                   "relaent"
@ MSG_DT_STRSZ_CF             "DT_STRSZ"                # 10
@ MSG_DT_STRSZ_CFNP                   "STRSZ"
```



```
@ MSG_DT_STRSZ_NF                     "strsz"
@ MSG_DT_SYMENT_CF            "DT_SYMENT"               # 11
@ MSG_DT_SYMENT_CFNP                  "SYMENT"
@ MSG_DT_SYMENT_NF                    "syment"
@ MSG_DT_INIT_CF              "DT_INIT"                 # 12
@ MSG_DT_INIT_CFNP                    "INIT"
@ MSG_DT_INIT_NF                      "init"
@ MSG_DT_FINI_CF              "DT_FINI"                 # 13
@ MSG_DT_FINI_CFNP                    "FINI"
@ MSG_DT_FINI_NF                      "fini"
@ MSG_DT_SONAME_CF            "DT_SONAME"               # 14
@ MSG_DT_SONAME_CFNP                  "SONAME"
@ MSG_DT_SONAME_NF                    "soname"
@ MSG_DT_RPATH_CF             "DT_RPATH"                # 15
@ MSG_DT_RPATH_CFNP                   "RPATH"
@ MSG_DT_RPATH_NF                     "rpath"
@ MSG_DT_SYMBOLIC_CF          "DT_SYMBOLIC"             # 16
@ MSG_DT_SYMBOLIC_CFNP                "SYMBOLIC"
@ MSG_DT_SYMBOLIC_NF                  "symbolic"
@ MSG_DT_SYMBOLIC_DMP                 "SYMB"
@ MSG_DT_REL_CF               "DT_REL"                  # 17
@ MSG_DT_REL_CFNP                     "REL"
@ MSG_DT_REL_NF                       "rel"
@ MSG_DT_RELSZ_CF             "DT_RELSZ"                # 18
@ MSG_DT_RELSZ_CFNP                   "RELSZ"
@ MSG_DT_RELSZ_NF                     "relsz"
@ MSG_DT_RELENT_CF            "DT_RELENT"               # 19
@ MSG_DT_RELENT_CFNP                  "RELENT"
@ MSG_DT_RELENT_NF                    "relent"
@ MSG_DT_PLTREL_CF            "DT_PLTREL"               # 20
@ MSG_DT_PLTREL_CFNP                  "PLTREL"
@ MSG_DT_PLTREL_NF                    "pltrel"
@ MSG_DT_DEBUG_CF             "DT_DEBUG"                # 21
@ MSG_DT_DEBUG_CFNP                   "DEBUG"
@ MSG_DT_DEBUG_NF                     "debug"
@ MSG_DT_TEXTREL_CF           "DT_TEXTREL"              # 22
@ MSG_DT_TEXTREL_CFNP                 "TEXTREL"
@ MSG_DT_TEXTREL_NF                   "textrel"
@ MSG_DT_JMPREL_CF            "DT_JMPREL"               # 23
@ MSG_DT_JMPREL_CFNP                  "JMPREL"
@ MSG_DT_JMPREL_NF                    "jmprel"
@ MSG_DT_BIND_NOW_CF          "DT_BIND_NOW"             # 24
@ MSG_DT_BIND_NOW_CFNP                "BIND_NOW"
@ MSG_DT_BIND_NOW_NF                  "bind_now"
@ MSG_DT_INIT_ARRAY_CF        "DT_INIT_ARRAY"           # 25
@ MSG_DT_INIT_ARRAY_CFNP              "INIT_ARRAY"
@ MSG_DT_INIT_ARRAY_NF                "init_array"
@ MSG_DT_FINI_ARRAY_CF        "DT_FINI_ARRAY"           # 26
@ MSG_DT_FINI_ARRAY_CFNP              "FINI_ARRAY"
@ MSG_DT_FINI_ARRAY_NF                "fini_array"
@ MSG_DT_INIT_ARRAYSZ_CF      "DT_INIT_ARRAYSZ"         # 27
@ MSG_DT_INIT_ARRAYSZ_CFNP            "INIT_ARRAYSZ"
@ MSG_DT_INIT_ARRAYSZ_NF              "init_arraysz"
@ MSG_DT_FINI_ARRAYSZ_CF      "DT_FINI_ARRAYSZ"         # 28
@ MSG_DT_FINI_ARRAYSZ_CFNP            "FINI_ARRAYSZ"
@ MSG_DT_FINI_ARRAYSZ_NF              "fini_arraysz"
@ MSG_DT_RUNPATH_CF           "DT_RUNPATH"              # 29
@ MSG_DT_RUNPATH_CFNP                 "RUNPATH"
@ MSG_DT_RUNPATH_NF                   "runpath"
@ MSG_DT_FLAGS_CF             "DT_FLAGS"                # 30
@ MSG_DT_FLAGS_CFNP                   "FLAGS"
@ MSG_DT_FLAGS_NF                     "flags"
@ MSG_DT_PREINIT_ARRAY_CF     "DT_PREINIT_ARRAY"        # 32
@ MSG_DT_PREINIT_ARRAY_CFNP           "PREINIT_ARRAY"
@ MSG_DT_PREINIT_ARRAY_NF             "preinit_array"
@ MSG_DT_PREINIT_ARRAYSZ_CF   "DT_PREINIT_ARRAYSZ"      # 33
```

```
125 @ MSG_DT_PREINIT_ARRAYSZ_CFNP           "PREINIT_ARRAYSZ"
126 @ MSG_DT_PREINIT_ARRAYSZ_NF             "preinit_arraysz"
127 @ MSG_DT_DEP_SPARC_REG_CF       "DT_DEPRECATED_SPARC_REGISTER"  # 0x07000001
128 @ MSG_DT_DEP_SPARC_REG_CFNP             "DEPRECATED_SPARC_REGISTER"
129 @ MSG_DT_DEP_SPARC_REG_NF               "deprecated_sparc_register"
130 @ MSG_DT_SUNW_AUXILIARY_CF      "DT_SUNW_AUXILIARY"             # 0x6000000d
131 @ MSG_DT_SUNW_AUXILIARY_CFNP            "SUNW_AUXILIARY"
132 @ MSG_DT_SUNW_AUXILIARY_NF              "sunw_auxiliary"
133 @ MSG_DT_SUNW_RTLDINF_CF        "DT_SUNW_RTLDINF"               # 0x6000000e
134 @ MSG_DT_SUNW_RTLDINF_CFNP              "SUNW_RTLDINF"
135 @ MSG_DT_SUNW_RTLDINF_NF                "sunw_rtldinf"
136 @ MSG_DT_SUNW_FILTER_CF         "DT_SUNW_FILTER"                # 0x6000000f
137 @ MSG_DT_SUNW_FILTER_CFNP               "SUNW_FILTER"
138 @ MSG_DT_SUNW_FILTER_NF                 "sunw_filter"
139 @ MSG_DT_SUNW_CAP_CF            "DT_SUNW_CAP"                   # 0x60000010
140 @ MSG_DT_SUNW_CAP_CFNP                  "SUNW_CAP"
141 @ MSG_DT_SUNW_CAP_NF                    "sunw_cap"
142 @ MSG_DT_SUNW_SYMTAB_CF         "DT_SUNW_SYMTAB"                # 0x60000011
143 @ MSG_DT_SUNW_SYMTAB_CFNP               "SUNW_SYMTAB"
144 @ MSG_DT_SUNW_SYMTAB_NF                 "sunw_symtab"
145 @ MSG_DT_SUNW_SYMSZ_CF           "DT_SUNW_SYMSZ"                 # 0x60000012
146 @ MSG_DT_SUNW_SYMSZ_CFNP                "SUNW_SYMSZ"
147 @ MSG_DT_SUNW_SYMSZ_NF                  "sunw_symsz"
148 @ MSG_DT_SUNW_SORTENT_CF        "DT_SUNW_SORTENT"               # 0x60000013
149 @ MSG_DT_SUNW_SORTENT_CFNP              "SUNW_SORTENT"
150 @ MSG_DT_SUNW_SORTENT_NF                "sunw_sortent"
151 @ MSG_DT_SUNW_SYMSORT_CF        "DT_SUNW_SYMSORT"               # 0x60000014
152 @ MSG_DT_SUNW_SYMSORT_CFNP              "SUNW_SYMSORT"
153 @ MSG_DT_SUNW_SYMSORT_NF                "sunw_symsort"
154 @ MSG_DT_SUNW_SYMSORTSZ_CF      "DT_SUNW_SYMSORTSZ"             # 0x60000015
155 @ MSG_DT_SUNW_SYMSORTSZ_CFNP            "SUNW_SYMSORTSZ"
156 @ MSG_DT_SUNW_SYMSORTSZ_NF              "sunw_symsortsz"
157 @ MSG_DT_SUNW_TLSSORT_CF        "DT_SUNW_TLSSORT"               # 0x60000016
158 @ MSG_DT_SUNW_TLSSORT_CFNP              "SUNW_TLSSORT"
159 @ MSG_DT_SUNW_TLSSORT_NF                "sunw_tlssort"
160 @ MSG_DT_SUNW_TLSSORTSZ_CF      "DT_SUNW_TLSSORTSZ"             # 0x60000017
161 @ MSG_DT_SUNW_TLSSORTSZ_CFNP            "SUNW_TLSSORTSZ"
162 @ MSG_DT_SUNW_TLSSORTSZ_NF              "sunw_tlssortsz"
163 @ MSG_DT_SUNW_CAPINFO_CF        "DT_SUNW_CAPINFO"               # 0x60000018
164 @ MSG_DT_SUNW_CAPINFO_CFNP              "SUNW_CAPINFO"
165 @ MSG_DT_SUNW_CAPINFO_NF                "sunw_capinfo"
166 @ MSG_DT_SUNW_STRPAD_CF         "DT_SUNW_STRPAD"                # 0x60000019
167 @ MSG_DT_SUNW_STRPAD_CFNP               "SUNW_STRPAD"
168 @ MSG_DT_SUNW_STRPAD_NF                 "sunw_strpad"
169 @ MSG_DT_SUNW_CAPCHAIN_CF       "DT_SUNW_CAPCHAIN"              # 0x6000001a
170 @ MSG_DT_SUNW_CAPCHAIN_CFNP             "SUNW_CAPCHAIN"
171 @ MSG_DT_SUNW_CAPCHAIN_NF               "sunw_capchain"
172 @ MSG_DT_SUNW_LDMACH_CF         "DT_SUNW_LDMACH"                # 0x6000001b
173 @ MSG_DT_SUNW_LDMACH_CFNP               "SUNW_LDMACH"
174 @ MSG_DT_SUNW_LDMACH_NF                 "sunw_ldmach"
175 @ MSG_DT_SUNW_CAPCHAINENT_CF    "DT_SUNW_CAPCHAINENT"           # 0x6000001d
176 @ MSG_DT_SUNW_CAPCHAINENT_CFNP          "SUNW_CAPCHAINENT"
177 @ MSG_DT_SUNW_CAPCHAINENT_NF            "sunw_capchainent"
178 @ MSG_DT_SUNW_CAPCHAINSZ_CF     "DT_SUNW_CAPCHAINSZ"            # 0x6000001d
179 @ MSG_DT_SUNW_CAPCHAINSZ_CFNP           "SUNW_CAPCHAINSZ"
180 @ MSG_DT_SUNW_CAPCHAINSZ_NF             "sunw_capchainsz"
181 @ MSG_DT_SUNW_ASLR_CF          "DT_SUNW_ASLR"                  # 0x60000023
182 @ MSG_DT_SUNW_ASLR_CFNP                "SUNW_ASLR"
183 @ MSG_DT_SUNW_ASLR_NF                  "sunw_aslr"
184 #endif /* ! codereview */

186 @ MSG_DT_GNU_PRELINKED_CF       "DT_GNU_PRELINKED"              # 0x6ffffdf5
187 @ MSG_DT_GNU_PRELINKED_CFNP             "GNU_PRELINKED"
188 @ MSG_DT_GNU_PRELINKED_NF               "gnu_prelinked"
189 @ MSG_DT_GNU_CONFLICTSZ_CF      "DT_GNU_CONFLICTSZ"             # 0x6ffffdf6
190 @ MSG_DT_GNU_CONFLICTSZ_CFNP            "GNU_CONFLICTSZ"
```



```
191 @ MSG_DT_GNU_CONFLICTSZ_NF             "gnu_conflictsz"
192 @ MSG_DT_GNU_LIBLISTSZ_CF      "DT_GNU_LIBLISTSZ"              # 0x6ffffdf7
193 @ MSG_DT_GNU_LIBLISTSZ_CFNP            "GNU_LIBLISTSZ"
194 @ MSG_DT_GNU_LIBLISTSZ_NF              "gnu_liblistsz"
195 @ MSG_DT_CHECKSUM_CF           "DT_CHECKSUM"                   # 0x6ffffdf8
196 @ MSG_DT_CHECKSUM_CFNP                 "CHECKSUM"
197 @ MSG_DT_CHECKSUM_NF                   "checksum"
198 @ MSG_DT_PLTPADSZ_CF            "DT_PLTPADSZ"                   # 0x6ffffdf9
199 @ MSG_DT_PLTPADSZ_CFNP                 "PLTPADSZ"
200 @ MSG_DT_PLTPADSZ_NF                   "pltpadsz"
201 @ MSG_DT_MOVEENT_CF             "DT_MOVEENT"                    # 0x6ffffdfa
202 @ MSG_DT_MOVEENT_CFNP                  "MOVEENT"
203 @ MSG_DT_MOVEENT_NF                    "moveent"
204 @ MSG_DT_MOVESZ_CF              "DT_MOVESZ"                     # 0x6ffffdfb
205 @ MSG_DT_MOVESZ_CFNP                  "MOVESZ"
206 @ MSG_DT_MOVESZ_NF                    "movesz"
207 @ MSG_DT_FEATURE_1_CF           "DT_FEATURE_1"                  # 0x6ffffdfc
208 @ MSG_DT_FEATURE_1_CFNP                "FEATURE_1"
209 @ MSG_DT_FEATURE_1_NF                  "feature_1"
210 @ MSG_DT_POSFLAG_1_CF           "DT_POSFLAG_1"                  # 0x6ffffdfd
211 @ MSG_DT_POSFLAG_1_CFNP                "POSFLAG_1"
212 @ MSG_DT_POSFLAG_1_NF                  "posflag_1"
213 @ MSG_DT_SYMINSZ_CF             "DT_SYMINSZ"                    # 0x6ffffdfe
214 @ MSG_DT_SYMINSZ_CFNP                 "SYMINSZ"
215 @ MSG_DT_SYMINSZ_NF                   "syminsz"
216 @ MSG_DT_SYMINENT_CF            "DT_SYMINENT"                   # 0x6ffffdff
217 @ MSG_DT_SYMINENT_CFNP                "SYMINENT"
218 @ MSG_DT_SYMINENT_NF                  "syminent"
219 @ MSG_DT_GNU_HASH_CF            "DT_GNU_HASH"                   # 0x6ffffef5
220 @ MSG_DT_GNU_HASH_CFNP                 "GNU_HASH"
221 @ MSG_DT_GNU_HASH_NF                   "gnu_hash"
222 @ MSG_DT_TLSDESC_PLT_CF         "DT_TLSDESC_PLT"                # 0x6ffffef6
223 @ MSG_DT_TLSDESC_PLT_CFNP               "TLSDESC_PLT"
224 @ MSG_DT_TLSDESC_PLT_NF                 "tlsdesc_plt"
225 @ MSG_DT_TLSDESC_GOT_CF         "DT_TLSDESC_GOT"                # 0x6ffffef7
226 @ MSG_DT_TLSDESC_GOT_CFNP               "TLSDESC_GOT"
227 @ MSG_DT_TLSDESC_GOT_NF                 "tlsdesc_got"
228 @ MSG_DT_GNU_CONFLICT_CF        "DT_GNU_CONFLICT"               # 0x6ffffef8
229 @ MSG_DT_GNU_CONFLICT_CFNP              "GNU_CONFLICT"
230 @ MSG_DT_GNU_CONFLICT_NF                "gnu_conflict"
231 @ MSG_DT_GNU_LIBLIST_CF         "DT_GNU_LIBLIST"                # 0x6ffffef9
232 @ MSG_DT_GNU_LIBLIST_CFNP               "GNU_LIBLIST"
233 @ MSG_DT_GNU_LIBLIST_NF                 "gnu_liblist"
234 @ MSG_DT_CONFIG_CF              "DT_CONFIG"                     # 0x6ffffefa
235 @ MSG_DT_CONFIG_CFNP                   "CONFIG"
236 @ MSG_DT_CONFIG_NF                     "config"
237 @ MSG_DT_DEPAUDIT_CF            "DT_DEPAUDIT"                   # 0x6ffffefb
238 @ MSG_DT_DEPAUDIT_CFNP                 "DEPAUDIT"
239 @ MSG_DT_DEPAUDIT_NF                   "depaudit"
240 @ MSG_DT_AUDIT_CF               "DT_AUDIT"                      # 0x6ffffefc
241 @ MSG_DT_AUDIT_CFNP                    "AUDIT"
242 @ MSG_DT_AUDIT_NF                      "audit"
243 @ MSG_DT_PLTPAD_CF              "DT_PLTPAD"                     # 0x6ffffefd
244 @ MSG_DT_PLTPAD_CFNP                   "PLTPAD"
245 @ MSG_DT_PLTPAD_NF                     "pltpad"
246 @ MSG_DT_MOVETAB_CF             "DT_MOVETAB"                    # 0x6ffffefe
247 @ MSG_DT_MOVETAB_CFNP                  "MOVETAB"
248 @ MSG_DT_MOVETAB_NF                    "movetab"
249 @ MSG_DT_SYMINFO_CF             "DT_SYMINFO"                    # 0x6ffffeff
250 @ MSG_DT_SYMINFO_CFNP                 "SYMINFO"
251 @ MSG_DT_SYMINFO_NF                   "syminfo"
252 @ MSG_DT_VERSYM_CF              "DT_VERSYM"                     # 0x6ffffff0
253 @ MSG_DT_VERSYM_CFNP                  "VERSYM"
254 @ MSG_DT_VERSYM_NF                    "versym"
255 @ MSG_DT_RELACOUNT_CF           "DT_RELACOUNT"                  # 0x6ffffff9
256 @ MSG_DT_RELACOUNT_CFNP                "RELACOUNT"
```

```
257 @ MSG_DT_RELACOUNT_NF               "relacount"
258 @ MSG_DT_RELCOUNT_CF           "DT_RELCOUNT"                   # 0x6ffffffa
259 @ MSG_DT_RELCOUNT_CFNP              "RELCOUNT"
260 @ MSG_DT_RELCOUNT_NF               "relcount"
261 @ MSG_DT_FLAGS_1_CF             "DT_FLAGS_1"                    # 0x6ffffffb
262 @ MSG_DT_FLAGS_1_CFNP               "FLAGS_1"
263 @ MSG_DT_FLAGS_1_NF                "flags_1"
264 @ MSG_DT_VERDEF_CF               "DT_VERDEF"                     # 0x6ffffffc
265 @ MSG_DT_VERDEF_CFNP                "VERDEF"
266 @ MSG_DT_VERDEF_NF                 "verdef"
267 @ MSG_DT_VERDEFNUM_CF            "DT_VERDEFNUM"                  # 0x6ffffffd
268 @ MSG_DT_VERDEFNUM_CFNP             "VERDEFNUM"
269 @ MSG_DT_VERDEFNUM_NF              "verdefnum"
270 @ MSG_DT_VERNEED_CF              "DT_VERNEED"                    # 0x6ffffffe
271 @ MSG_DT_VERNEED_CFNP               "VERNEED"
272 @ MSG_DT_VERNEED_NF                "verneed"
273 @ MSG_DT_VERNEEDNUM_CF           "DT_VERNEEDNUM"                 # 0x6fffffff
274 @ MSG_DT_VERNEEDNUM_CFNP            "VERNEEDNUM"
275 @ MSG_DT_VERNEEDNUM_NF             "verneednum"
276 @ MSG_DT_SPARC_REGISTER_CF      "DT_SPARC_REGISTER"             # 0x70000001
277 @ MSG_DT_SPARC_REGISTER_CFNP         "SPARC_REGISTER"
278 @ MSG_DT_SPARC_REGISTER_NF          "sparc_register"
279 @ MSG_DT_SPARC_REGISTER_DMP         "REGISTER"
280 @ MSG_DT_AUXILIARY_CF            "DT_AUXILIARY"                  # 0x7ffffffd
281 @ MSG_DT_AUXILIARY_CFNP             "AUXILIARY"
282 @ MSG_DT_AUXILIARY_NF              "auxiliary"
283 @ MSG_DT_USED_CF                 "DT_USED"                       # 0x7ffffffe
284 @ MSG_DT_USED_CFNP                  "USED"
285 @ MSG_DT_USED_NF                   "used"
286 @ MSG_DT_FILTER_CF               "DT_FILTER"                     # 0x7fffffff
287 @ MSG_DT_FILTER_CFNP                "FILTER"
288 @ MSG_DT_FILTER_NF                 "filter"


291 @ MSG_DF_ORIGIN_CF              "DF_ORIGIN"             # 0x00000001
292 @ MSG_DF_ORIGIN_CFNP               "ORIGIN"
293 @ MSG_DF_ORIGIN_NF                "origin"
294 @ MSG_DF_SYMBOLIC_CF            "DF_SYMBOLIC"           # 0x00000002
295 @ MSG_DF_SYMBOLIC_CFNP             "SYMBOLIC"
296 @ MSG_DF_SYMBOLIC_NF              "symbolic"
297 @ MSG_DF_TEXTREL_CF              "DF_TEXTREL"            # 0x00000004
298 @ MSG_DF_TEXTREL_CFNP               "TEXTREL"
299 @ MSG_DF_TEXTREL_NF                "textrel"
300 @ MSG_DF_BIND_NOW_CF             "DF_BIND_NOW"           # 0x00000008
301 @ MSG_DF_BIND_NOW_CFNP              "BIND_NOW"
302 @ MSG_DF_BIND_NOW_NF               "bind_now"
303 @ MSG_DF_STATIC_TLS_CF           "DF_STATIC_TLS"         # 0x00000010
304 @ MSG_DF_STATIC_TLS_CFNP            "STATIC_TLS"
305 @ MSG_DF_STATIC_TLS_NF             "static_tls"


308 @ MSG_DF_1_NOW_CF               "DF_1_NOW"              # 0x00000001
309 @ MSG_DF_1_NOW_CFNP                "NOW"
310 @ MSG_DF_1_NOW_NF                 "now"
311 @ MSG_DF_1_GLOBAL_CF            "DF_1_GLOBAL"           # 0x00000002
312 @ MSG_DF_1_GLOBAL_CFNP             "GLOBAL"
313 @ MSG_DF_1_GLOBAL_NF              "global"
314 @ MSG_DF_1_GROUP_CF              "DF_1_GROUP"            # 0x00000004
315 @ MSG_DF_1_GROUP_CFNP               "GROUP"
316 @ MSG_DF_1_GROUP_NF                "group"
317 @ MSG_DF_1_NODELETE_CF           "DF_1_NODELETE"         # 0x00000008
318 @ MSG_DF_1_NODELETE_CFNP            "NODELETE"
319 @ MSG_DF_1_NODELETE_NF             "nodelete"
320 @ MSG_DF_1_LOADFLTR_CF           "DF_1_LOADFLTR"         # 0x00000010
321 @ MSG_DF_1_LOADFLTR_CFNP            "LOADFLTR"
322 @ MSG_DF_1_LOADFLTR_NF             "loadfltr"
```



```
323 @ MSG_DF_1_INITFIRST_CF         "DF_1_INITFIRST"        # 0x00000020
324 @ MSG_DF_1_INITFIRST_CFNP            "INITFIRST"
325 @ MSG_DF_1_INITFIRST_NF             "initfirst"
326 @ MSG_DF_1_NOOPEN_CF             "DF_1_NOOPEN"           # 0x00000040
327 @ MSG_DF_1_NOOPEN_CFNP              "NOOPEN"
328 @ MSG_DF_1_NOOPEN_NF               "noopen"
329 @ MSG_DF_1_ORIGIN_CF             "DF_1_ORIGIN"           # 0x00000080
330 @ MSG_DF_1_ORIGIN_CFNP              "ORIGIN"
331 @ MSG_DF_1_ORIGIN_NF               "origin"
332 @ MSG_DF_1_DIRECT_CF             "DF_1_DIRECT"           # 0x00000100
333 @ MSG_DF_1_DIRECT_CFNP              "DIRECT"
334 @ MSG_DF_1_DIRECT_NF               "direct"
335 @ MSG_DF_1_TRANS_CF              "DF_1_TRANS"            # 0x00000200
336 @ MSG_DF_1_TRANS_CFNP               "TRANS"
337 @ MSG_DF_1_TRANS_NF                "trans"
338 @ MSG_DF_1_INTERPOSE_CF          "DF_1_INTERPOSE"        # 0x00000400
339 @ MSG_DF_1_INTERPOSE_CFNP           "INTERPOSE"
340 @ MSG_DF_1_INTERPOSE_NF            "interpose"
341 @ MSG_DF_1_INTERPOSE_DEF           "OBJECT-INTERPOSE"
342 @ MSG_DF_1_NODEFLIB_CF           "DF_1_NODEFLIB"         # 0x00000800
343 @ MSG_DF_1_NODEFLIB_CFNP            "NODEFLIB"
344 @ MSG_DF_1_NODEFLIB_NF             "nodeflib"
345 @ MSG_DF_1_NODUMP_CF             "DF_1_NODUMP"           # 0x00001000
346 @ MSG_DF_1_NODUMP_CFNP              "NODUMP"
347 @ MSG_DF_1_NODUMP_NF               "nodump"
348 @ MSG_DF_1_CONFALT_CF            "DF_1_CONFALT"          # 0x00002000
349 @ MSG_DF_1_CONFALT_CFNP             "CONFALT"
350 @ MSG_DF_1_CONFALT_NF              "confalt"
351 @ MSG_DF_1_ENDFILTEE_CF          "DF_1_ENDFILTEE"        # 0x00004000
352 @ MSG_DF_1_ENDFILTEE_CFNP           "ENDFILTEE"
353 @ MSG_DF_1_ENDFILTEE_NF            "endfiltee"
354 @ MSG_DF_1_DISPRELDNE_CF         "DF_1_DISPRELDNE"       # 0x00008000
355 @ MSG_DF_1_DISPRELDNE_CFNP          "DISPRELDNE"
356 @ MSG_DF_1_DISPRELDNE_NF           "dispreldne"
357 @ MSG_DF_1_DISPRELDNE_DEF          "DISPLACE-RELOCS-DONE"
358 @ MSG_DF_1_DISPRELPND_CF         "DF_1_DISPRELPND"       # 0x00010000
359 @ MSG_DF_1_DISPRELPND_CFNP          "DISPRELPND"
360 @ MSG_DF_1_DISPRELPND_NF           "disprelpnd"
361 @ MSG_DF_1_DISPRELPND_DEF          "DISPLACE-RELOCS-PEND"
362 @ MSG_DF_1_NODIRECT_CF           "DF_1_NODIRECT"         # 0x00020000
363 @ MSG_DF_1_NODIRECT_CFNP            "NODIRECT"
364 @ MSG_DF_1_NODIRECT_NF             "nodirect"
365 @ MSG_DF_1_IGNMULDEF_CF          "DF_1_IGNMULDEF"        # 0x00040000
366 @ MSG_DF_1_IGNMULDEF_CFNP           "IGNMULDEF"
367 @ MSG_DF_1_IGNMULDEF_NF            "ignmuldef"
368 @ MSG_DF_1_IGNMULDEF_DEF           "IGNORE-MULDEFS"
369 @ MSG_DF_1_NOKSYMS_CF            "DF_1_NOKSYMS"          # 0x00080000
370 @ MSG_DF_1_NOKSYMS_CFNP             "NOKSYMS"
371 @ MSG_DF_1_NOKSYMS_NF              "noksyms"
372 @ MSG_DF_1_NOHDR_CF              "DF_1_NOHDR"            # 0x00100000
373 @ MSG_DF_1_NOHDR_CFNP               "NOHDR"
374 @ MSG_DF_1_NOHDR_NF                "nohdr"
375 @ MSG_DF_1_EDITED_CF             "DF_1_EDITED"           # 0x00200000
376 @ MSG_DF_1_EDITED_CFNP              "EDITED"
377 @ MSG_DF_1_EDITED_NF               "edited"
378 @ MSG_DF_1_NORELOC_CF            "DF_1_NORELOC"          # 0x00400000
379 @ MSG_DF_1_NORELOC_CFNP             "NORELOC"
380 @ MSG_DF_1_NORELOC_NF              "noreloc"
381 @ MSG_DF_1_SYMINTPOSE_CF         "DF_1_SYMINTPOSE"       # 0x00800000
382 @ MSG_DF_1_SYMINTPOSE_CFNP          "SYMINTPOSE"
383 @ MSG_DF_1_SYMINTPOSE_NF           "symintpose"
384 @ MSG_DF_1_SYMINTPOSE_DEF          "SYMBOL-INTERPOSE"
385 @ MSG_DF_1_GLOBAUDIT_CF          "DF_1_GLOBAUDIT"        # 0x01000000
386 @ MSG_DF_1_GLOBAUDIT_CFNP           "GLOBAUDIT"
387 @ MSG_DF_1_GLOBAUDIT_NF            "globaudit"
388 @ MSG_DF_1_GLOBAUDIT_DEF           "GLOBAL-AUDITING"
```

```
389 @ MSG_DF_1_SINGLETON_CF         "DF_1_SINGLETON"        # 0x02000000
390 @ MSG_DF_1_SINGLETON_CFNP               "SINGLETON"
391 @ MSG_DF_1_SINGLETON_NF                 "singleton"
392 @ MSG_DF_1_SINGLETON_DEF                "SINGLETON-EXISTS"


395 @ MSG_DF_P1_LAZYLOAD_CF         "DF_P1_LAZYLOAD"        # 0x00000001
396 @ MSG_DF_P1_LAZYLOAD_CFNP               "LAZYLOAD"
397 @ MSG_DF_P1_LAZYLOAD_NF                 "lazyload"
398 @ MSG_DF_P1_LAZYLOAD_DEF                "LAZY"
399 @ MSG_DF_P1_GROUPPERM_CF        "DF_P1_GROUPPERM"       # 0x00000002
400 @ MSG_DF_P1_GROUPPERM_CFNP              "GROUPPERM"
401 @ MSG_DF_P1_GROUPPERM_NF                "groupperm"
402 @ MSG_DF_P1_GROUPPERM_DEF               "GROUP"
403 @ MSG_DF_P1_DEFERRED_CF         "DF_P1_DEFERRED"        # 0x00000004
404 @ MSG_DF_P1_DEFERRED_CFNP               "DEFERRED"
405 @ MSG_DF_P1_DEFERRED_NF                 "deferred"
406 @ MSG_DF_P1_DEFERRED_DEF                "DEFERRED"


409 @ MSG_DTF_1_PARINIT_CF          "DTF_1_PARINIT"         # 0x00000001
410 @ MSG_DTF_1_PARINIT_CFNP                "PARINIT"
411 @ MSG_DTF_1_PARINIT_NF                  "parinit"
412 @ MSG_DTF_1_CONFEXP_CF          "DTF_1_CONFEXP"         # 0x00000002
413 @ MSG_DTF_1_CONFEXP_CFNP                "CONFEXP"
414 @ MSG_DTF_1_CONFEXP_NF                  "confexp"


417 @ MSG_BND_NEEDED        "NEEDED"
418 @ MSG_BND_REFER         "REFERENCED"
419 @ MSG_BND_FILTER        "FILTER"


422 @ MSG_BND_ADDED         "OBJECTS-ADDED"
423 @ MSG_BND_REEVAL        "OBJECTS-REEVALUATED"
424 @ MSG_BND_DELETED       "OBJECTS-DELETED"
425 @ MSG_BND_ATEXIT        "ATEXIT-PROCESSING"
426 @ MSG_BND_REVISIT       "(revisiting)"

428 @ MSG_STR_EMPTY         ""

430 @ MSG_GBL_ZERO          "0"
```

```
**********************************************************
   65165 Wed May 27 19:49:04 2015
new/usr/src/cmd/sgs/libld/common/args.c
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
______unchanged_portion_omitted_

 916 /*
 917  * Parse the -z assert-deflib option. This option can appear in two different
 918  * forms:
 919  *      -z assert-deflib
 920  *      -z assert-deflib=libfred.so
 921  *
 922  * Either form enables this option, the latter form marks libfred.so as an
 923  * exempt library from the check. It is valid to have multiple invocations of
 924  * the second form. We silently ignore mulitple occurrences of the first form
 925  * and multiple invocations of the first form when the second form also occurs.
 926  *
 927  * We only return false when we have an internal error, such as the failure of
 928  * aplist_append. Every other time we return true, but we have the appropriate
 929  * fatal flags set beacuse of the ld_eprintf.
 930  */
 931 static int
 932 assdeflib_parse(Ofl_desc *ofl, char *optarg)
 933 {
 934         size_t olen, mlen;
 935         ofl->ofl_flags |= FLG_OF_ADEFLIB;

 937         olen = strlen(optarg);
 938         /* Minimum size of assert-deflib=lib%s.so */
 939         mlen = MSG_ARG_ASSDEFLIB_SIZE + 1 + MSG_STR_LIB_SIZE +
 940             MSG_STR_SOEXT_SIZE;
 941         if (olen > MSG_ARG_ASSDEFLIB_SIZE) {
 942                 if (optarg[MSG_ARG_ASSDEFLIB_SIZE] != '=') {
 943                         ld_eprintf(ofl, ERR_FATAL, "Missing =\n");
 943                         ld_eprintf(ofl, ERR_FATAL, MSG_INTL(MSG_ARG_ILLEGAL),
 944                             MSG_ORIG(MSG_ARG_ASSDEFLIB), optarg);
 945                         return (TRUE);
 946                 }

 948                 if (strncmp(optarg + MSG_ARG_ASSDEFLIB_SIZE + 1,
 949                     MSG_ORIG(MSG_STR_LIB), MSG_STR_LIB_SIZE) != 0 ||
 950                     strcmp(optarg + olen - MSG_STR_SOEXT_SIZE,
 951                     MSG_ORIG(MSG_STR_SOEXT)) != 0 || olen <= mlen) {
 952                         ld_eprintf(ofl, ERR_FATAL,
 953                             MSG_INTL(MSG_ARG_ASSDEFLIB_MALFORMED), optarg);
 954                         return (TRUE);
 955                 }

 957                 if (aplist_append(&ofl->ofl_assdeflib, optarg +
 958                     MSG_ARG_ASSDEFLIB_SIZE + 1, AL_CNT_ASSDEFLIB) == NULL)
 959                         return (FALSE);
 960         }

 962         return (TRUE);
 963 }

 965 static int      optitle = 0;
 966 /*
 967  * Parsing options pass1 for process_flags().
 968  */
 969 static uintptr_t
 970 parseopt_pass1(Ofl_desc *ofl, int argc, char **argv, int *usage)
```



```
 971 {
 972         int     c, ndx = optind;

 974         /*
 975          * The -32, -64 and -ztarget options are special, in that we validate
 976          * them, but otherwise ignore them. libld.so (this code) is called
 977          * from the ld front end program. ld has already examined the
 978          * arguments to determine the output class and machine type of the
 979          * output object, as reflected in the version (32/64) of ld_main()
 980          * that was called and the value of the 'mach' argument passed.
 981          * By time execution reaches this point, these options have already
 982          * been seen and acted on.
 983          */
 984         while ((c = ld_getopt(ofl->ofl_lml, ndx, argc, argv)) != -1) {

 986                 switch (c) {
 987                 case '3':
 988                         DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));

 990                         /*
 991                          * -32 is processed by ld to determine the output class.
 992                          * Here we sanity check the option incase some other
 993                          * -3* option is mistakenly passed to us.
 994                          */
 995                         if (optarg[0] != '2')
 996                                 ld_eprintf(ofl, ERR_FATAL,
 997                                     MSG_INTL(MSG_ARG_ILLEGAL),
 998                                     MSG_ORIG(MSG_ARG_3), optarg);
 999                         continue;

1001                 case '6':
1002                         DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));

1004                         /*
1005                          * -64 is processed by ld to determine the output class.
1006                          * Here we sanity check the option incase some other
1007                          * -6* option is mistakenly passed to us.
1008                          */
1009                         if (optarg[0] != '4')
1010                                 ld_eprintf(ofl, ERR_FATAL,
1011                                     MSG_INTL(MSG_ARG_ILLEGAL),
1012                                     MSG_ORIG(MSG_ARG_6), optarg);
1013                         continue;

1015                 case 'a':
1016                         DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, NULL));
1017                         aflag = TRUE;
1018                         break;

1020                 case 'b':
1021                         DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, NULL));
1022                         bflag = TRUE;

1024                         /*
1025                          * This is a hack, and may be undone later.
1026                          * The -b option is only used to build the Unix
1027                          * kernel and its related kernel-mode modules.
1028                          * We do not want those files to get a .SUNW_ldynsym
1029                          * section. At least for now, the kernel makes no
1030                          * use of .SUNW_ldynsym, and we do not want to use
1031                          * the space to hold it. Therefore, we overload
1032                          * the use of -b to also imply -znoldynsym.
1033                          */
1034                         ofl->ofl_flags |= FLG_OF_NOLDYNSYM;
1035                         break;
```

```
1037                case 'c':
1038                        DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));
1039                        if (ofl->ofl_config)
1040                                ld_eprintf(ofl, ERR_WARNING_NF,
1041                                    MSG_INTL(MSG_ARG_MTONCE),
1042                                    MSG_ORIG(MSG_ARG_C));
1043                        else
1044                                ofl->ofl_config = optarg;
1045                        break;

1047                case 'C':
1048                        DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, NULL));
1049                        demangle_flag = 1;
1050                        break;

1052                case 'd':
1053                        DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));
1054                        if ((optarg[0] == 'n') && (optarg[1] == '\0')) {
1055                                if (dflag != SET_UNKNOWN)
1056                                        ld_eprintf(ofl, ERR_WARNING_NF,
1057                                            MSG_INTL(MSG_ARG_MTONCE),
1058                                            MSG_ORIG(MSG_ARG_D));
1059                                else
1060                                        dflag = SET_FALSE;
1061                        } else if ((optarg[0] == 'y') && (optarg[1] == '\0')) {
1062                                if (dflag != SET_UNKNOWN)
1063                                        ld_eprintf(ofl, ERR_WARNING_NF,
1064                                            MSG_INTL(MSG_ARG_MTONCE),
1065                                            MSG_ORIG(MSG_ARG_D));
1066                                else
1067                                        dflag = SET_TRUE;
1068                        } else {
1069                                ld_eprintf(ofl, ERR_FATAL,
1070                                    MSG_INTL(MSG_ARG_ILLEGAL),
1071                                    MSG_ORIG(MSG_ARG_D), optarg);
1072                        }
1073                        break;

1075                case 'e':
1076                        DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));
1077                        if (ofl->ofl_entry)
1078                                ld_eprintf(ofl, ERR_WARNING_NF,
1079                                    MSG_INTL(MSG_MARG_MTONCE),
1080                                    MSG_INTL(MSG_MARG_ENTRY));
1081                        else
1082                                ofl->ofl_entry = (void *)optarg;
1083                        break;

1085                case 'f':
1086                        DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));
1087                        if (ofl->ofl_filtees &&
1088                            (!(ofl->ofl_flags & FLG_OF_AUX))) {
1089                                ld_eprintf(ofl, ERR_FATAL,
1090                                    MSG_INTL(MSG_MARG_INCOMP),
1091                                    MSG_INTL(MSG_MARG_FILTER_AUX),
1092                                    MSG_INTL(MSG_MARG_FILTER));
1093                        } else {
1094                                if ((ofl->ofl_filtees =
1095                                    add_string(ofl->ofl_filtees, optarg)) ==
1096                                    (const char *)S_ERROR)
1097                                        return (S_ERROR);
1098                                ofl->ofl_flags |= FLG_OF_AUX;
1099                        }
1100                        break;

1102                case 'F':
```



```
1103                        DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));
1104                        if (ofl->ofl_filtees &&
1105                            (ofl->ofl_flags & FLG_OF_AUX)) {
1106                                ld_eprintf(ofl, ERR_FATAL,
1107                                    MSG_INTL(MSG_MARG_INCOMP),
1108                                    MSG_INTL(MSG_MARG_FILTER),
1109                                    MSG_INTL(MSG_MARG_FILTER_AUX));
1110                        } else {
1111                                if ((ofl->ofl_filtees =
1112                                    add_string(ofl->ofl_filtees, optarg)) ==
1113                                    (const char *)S_ERROR)
1114                                        return (S_ERROR);
1115                        }
1116                        break;

1118                case 'h':
1119                        DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));
1120                        if (ofl->ofl_soname)
1121                                ld_eprintf(ofl, ERR_WARNING_NF,
1122                                    MSG_INTL(MSG_MARG_MTONCE),
1123                                    MSG_INTL(MSG_MARG_SONAME));
1124                        else
1125                                ofl->ofl_soname = (const char *)optarg;
1126                        break;

1128                case 'i':
1129                        DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, NULL));
1130                        ofl->ofl_flags |= FLG_OF_IGNENV;
1131                        break;

1133                case 'I':
1134                        DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));
1135                        if (ofl->ofl_interp)
1136                                ld_eprintf(ofl, ERR_WARNING_NF,
1137                                    MSG_INTL(MSG_ARG_MTONCE),
1138                                    MSG_ORIG(MSG_ARG_CI));
1139                        else
1140                                ofl->ofl_interp = (const char *)optarg;
1141                        break;

1143                case 'l':
1144                        DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));
1145                        /*
1146                         * For now, count any library as a shared object.  This
1147                         * is used to size the internal symbol cache.  This
1148                         * value is recalculated later on actual file processing
1149                         * to get an accurate shared object count.
1150                         */
1151                        ofl->ofl_soscnt++;
1152                        break;

1154                case 'm':
1155                        DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, NULL));
1156                        ofl->ofl_flags |= FLG_OF_GENMAP;
1157                        break;

1159                case 'o':
1160                        DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));
1161                        if (ofl->ofl_name)
1162                                ld_eprintf(ofl, ERR_WARNING_NF,
1163                                    MSG_INTL(MSG_MARG_MTONCE),
1164                                    MSG_INTL(MSG_MARG_OUTFILE));
1165                        else
1166                                ofl->ofl_name = (const char *)optarg;
1167                        break;
```

```
1169                case 'p':
1170                        DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));

1172                        /*
1173                         * Multiple instances of this option may occur.  Each
1174                         * additional instance is effectively concatenated to
1175                         * the previous separated by a colon.
1176                         */
1177                        if (*optarg != '\0') {
1178                                if ((ofl->ofl_audit =
1179                                    add_string(ofl->ofl_audit,
1180                                    optarg)) == (const char *)S_ERROR)
1181                                        return (S_ERROR);
1182                        }
1183                        break;

1185                case 'P':
1186                        DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));

1188                        /*
1189                         * Multiple instances of this option may occur.  Each
1190                         * additional instance is effectively concatenated to
1191                         * the previous separated by a colon.
1192                         */
1193                        if (*optarg != '\0') {
1194                                if ((ofl->ofl_depaudit =
1195                                    add_string(ofl->ofl_depaudit,
1196                                    optarg)) == (const char *)S_ERROR)
1197                                        return (S_ERROR);
1198                        }
1199                        break;

1201                case 'r':
1202                        DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, NULL));
1203                        rflag = TRUE;
1204                        break;

1206                case 'R':
1207                        DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));

1209                        /*
1210                         * Multiple instances of this option may occur.  Each
1211                         * additional instance is effectively concatenated to
1212                         * the previous separated by a colon.
1213                         */
1214                        if (*optarg != '\0') {
1215                                if ((ofl->ofl_rpath =
1216                                    add_string(ofl->ofl_rpath,
1217                                    optarg)) == (const char *)S_ERROR)
1218                                        return (S_ERROR);
1219                        }
1220                        break;

1222                case 's':
1223                        DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, NULL));
1224                        sflag = TRUE;
1225                        break;

1227                case 't':
1228                        DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, NULL));
1229                        ofl->ofl_flags |= FLG_OF_NOWARN;
1230                        break;

1232                case 'u':
1233                        DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));
1234                        break;
```



```
1236                case 'z':
1237                        DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));

1239                        /*
1240                         * For specific help, print our usage message and exit
1241                         * immediately to ensure a 0 return code.
1242                         */
1243                        if (strncmp(optarg, MSG_ORIG(MSG_ARG_HELP),
1244                            MSG_ARG_HELP_SIZE) == 0) {
1245                                usage_mesg(TRUE);
1246                                exit(0);
1247                        }

1249                        /*
1250                         * For some options set a flag - further consistancy
1251                         * checks will be carried out in check_flags().
1252                         */
1253                        if ((strncmp(optarg, MSG_ORIG(MSG_ARG_LD32),
1254                            MSG_ARG_LD32_SIZE) == 0) ||
1255                            (strncmp(optarg, MSG_ORIG(MSG_ARG_LD64),
1256                            MSG_ARG_LD64_SIZE) == 0)) {
1257                                if (createargv(ofl, usage) == S_ERROR)
1258                                        return (S_ERROR);

1260                        } else if (
1261                            strcmp(optarg, MSG_ORIG(MSG_ARG_DEFS)) == 0) {
1262                                if (zdflag != SET_UNKNOWN)
1263                                        ld_eprintf(ofl, ERR_WARNING_NF,
1264                                            MSG_INTL(MSG_ARG_MTONCE),
1265                                            MSG_ORIG(MSG_ARG_ZDEFNODEF));
1266                                else
1267                                        zdflag = SET_TRUE;
1268                                ofl->ofl_guideflags |= FLG_OFG_NO_DEFS;
1269                        } else if (strcmp(optarg,
1270                            MSG_ORIG(MSG_ARG_NODEFS)) == 0) {
1271                                if (zdflag != SET_UNKNOWN)
1272                                        ld_eprintf(ofl, ERR_WARNING_NF,
1273                                            MSG_INTL(MSG_ARG_MTONCE),
1274                                            MSG_ORIG(MSG_ARG_ZDEFNODEF));
1275                                else
1276                                        zdflag = SET_FALSE;
1277                                ofl->ofl_guideflags |= FLG_OFG_NO_DEFS;
1278                        } else if (strcmp(optarg,
1279                            MSG_ORIG(MSG_ARG_TEXT)) == 0) {
1280                                if (ztflag &&
1281                                    (ztflag != MSG_ORIG(MSG_ARG_ZTEXT)))
1282                                        ld_eprintf(ofl, ERR_FATAL,
1283                                            MSG_INTL(MSG_ARG_INCOMP),
1284                                            MSG_ORIG(MSG_ARG_ZTEXT),
1285                                            ztflag);
1286                                ztflag = MSG_ORIG(MSG_ARG_ZTEXT);
1287                        } else if (strcmp(optarg,
1288                            MSG_ORIG(MSG_ARG_TEXTOFF)) == 0) {
1289                                if (ztflag &&
1290                                    (ztflag != MSG_ORIG(MSG_ARG_ZTEXTOFF)))
1291                                        ld_eprintf(ofl, ERR_FATAL,
1292                                            MSG_INTL(MSG_ARG_INCOMP),
1293                                            MSG_ORIG(MSG_ARG_ZTEXTOFF),
1294                                            ztflag);
1295                                ztflag = MSG_ORIG(MSG_ARG_ZTEXTOFF);
1296                        } else if (strcmp(optarg,
1297                            MSG_ORIG(MSG_ARG_TEXTWARN)) == 0) {
1298                                if (ztflag &&
1299                                    (ztflag != MSG_ORIG(MSG_ARG_ZTEXTWARN)))
1300                                        ld_eprintf(ofl, ERR_FATAL,
```

```
1301                                            MSG_INTL(MSG_ARG_INCOMP),
1302                                            MSG_ORIG(MSG_ARG_ZTEXTWARN),
1303                                            ztflag);
1304                                    ztflag = MSG_ORIG(MSG_ARG_ZTEXTWARN);
1305
1306                            /*
1307                             * For other options simply set the ofl flags directly.
1308                             */
1309                            } else if (strcmp(optarg,
1310                                MSG_ORIG(MSG_ARG_RESCAN)) == 0) {
1311                                    ofl->ofl_flags1 |= FLG_OF1_RESCAN;
1312                            } else if (strcmp(optarg,
1313                                MSG_ORIG(MSG_ARG_ABSEXEC)) == 0) {
1314                                    ofl->ofl_flags1 |= FLG_OF1_ABSEXEC;
1315                            } else if (strcmp(optarg,
1316                                MSG_ORIG(MSG_ARG_LOADFLTR)) == 0) {
1317                                    zlflag = TRUE;
1318                            } else if (strcmp(optarg,
1319                                MSG_ORIG(MSG_ARG_NORELOC)) == 0) {
1320                                    ofl->ofl_dtflags_1 |= DF_1_NORELOC;
1321                            } else if (strcmp(optarg,
1322                                MSG_ORIG(MSG_ARG_NOVERSION)) == 0) {
1323                                    ofl->ofl_flags |= FLG_OF_NOVERSEC;
1324                            } else if (strcmp(optarg,
1325                                MSG_ORIG(MSG_ARG_MULDEFS)) == 0) {
1326                                    ofl->ofl_flags |= FLG_OF_MULDEFS;
1327                            } else if (strcmp(optarg,
1328                                MSG_ORIG(MSG_ARG_REDLOCSYM)) == 0) {
1329                                    ofl->ofl_flags |= FLG_OF_REDLSYM;
1330                            } else if (strcmp(optarg,
1331                                MSG_ORIG(MSG_ARG_INITFIRST)) == 0) {
1332                                    ofl->ofl_dtflags_1 |= DF_1_INITFIRST;
1333                            } else if (strcmp(optarg,
1334                                MSG_ORIG(MSG_ARG_NODELETE)) == 0) {
1335                                    ofl->ofl_dtflags_1 |= DF_1_NODELETE;
1336                            } else if (strcmp(optarg,
1337                                MSG_ORIG(MSG_ARG_NOPARTIAL)) == 0) {
1338                                    ofl->ofl_flags1 |= FLG_OF1_NOPARTI;
1339                            } else if (strcmp(optarg,
1340                                MSG_ORIG(MSG_ARG_NOOPEN)) == 0) {
1341                                    ofl->ofl_dtflags_1 |= DF_1_NOOPEN;
1342                            } else if (strcmp(optarg,
1343                                MSG_ORIG(MSG_ARG_NOW)) == 0) {
1344                                    ofl->ofl_dtflags_1 |= DF_1_NOW;
1345                                    ofl->ofl_dtflags |= DF_BIND_NOW;
1346                            } else if (strcmp(optarg,
1347                                MSG_ORIG(MSG_ARG_ORIGIN)) == 0) {
1348                                    ofl->ofl_dtflags_1 |= DF_1_ORIGIN;
1349                                    ofl->ofl_dtflags |= DF_ORIGIN;
1350                            } else if (strcmp(optarg,
1351                                MSG_ORIG(MSG_ARG_NODEFAULTLIB)) == 0) {
1352                                    ofl->ofl_dtflags_1 |= DF_1_NODEFLIB;
1353                            } else if (strcmp(optarg,
1354                                MSG_ORIG(MSG_ARG_NODUMP)) == 0) {
1355                                    ofl->ofl_dtflags_1 |= DF_1_NODUMP;
1356                            } else if (strcmp(optarg,
1357                                MSG_ORIG(MSG_ARG_ENDFILTEE)) == 0) {
1358                                    ofl->ofl_dtflags_1 |= DF_1_ENDFILTEE;
1359                            } else if (strcmp(optarg,
1360                                MSG_ORIG(MSG_ARG_VERBOSE)) == 0) {
1361                                    ofl->ofl_flags |= FLG_OF_VERBOSE;
1362                            } else if (strcmp(optarg,
1363                                MSG_ORIG(MSG_ARG_COMBRELOC)) == 0) {
1364                                    ofl->ofl_flags |= FLG_OF_COMREL;
1365                            } else if (strcmp(optarg,
1366                                MSG_ORIG(MSG_ARG_NOCOMBRELOC)) == 0) {
```



```
1367                                    ofl->ofl_flags |= FLG_OF_NOCOMREL;
1368                            } else if (strcmp(optarg,
1369                                MSG_ORIG(MSG_ARG_NOCOMPSTRTAB)) == 0) {
1370                                    ofl->ofl_flags1 |= FLG_OF1_NCSTTAB;
1371                            } else if (strcmp(optarg,
1372                                MSG_ORIG(MSG_ARG_NOINTERP)) == 0) {
1373                                    ofl->ofl_flags1 |= FLG_OF1_NOINTRP;
1374                            } else if (strcmp(optarg,
1375                                MSG_ORIG(MSG_ARG_INTERPOSE)) == 0) {
1376                                    zinflag = TRUE;
1377                            } else if (strcmp(optarg,
1378                                MSG_ORIG(MSG_ARG_IGNORE)) == 0) {
1379                                    ofl->ofl_flags1 |= FLG_OF1_IGNPRC;
1380                            } else if (strcmp(optarg,
1381                                MSG_ORIG(MSG_ARG_RELAXRELOC)) == 0) {
1382                                    ofl->ofl_flags1 |= FLG_OF1_RLXREL;
1383                            } else if (strcmp(optarg,
1384                                MSG_ORIG(MSG_ARG_NORELAXRELOC)) == 0) {
1385                                    ofl->ofl_flags1 |= FLG_OF1_NRLXREL;
1386                            } else if (strcmp(optarg,
1387                                MSG_ORIG(MSG_ARG_NOLDYNSYM)) == 0) {
1388                                    ofl->ofl_flags |= FLG_OF_NOLDYNSYM;
1389                            } else if (strcmp(optarg,
1390                                MSG_ORIG(MSG_ARG_GLOBAUDIT)) == 0) {
1391                                    ofl->ofl_dtflags_1 |= DF_1_GLOBAUDIT;
1392                            } else if (strcmp(optarg,
1393                                MSG_ORIG(MSG_ARG_NOSIGHANDLER)) == 0) {
1394                                    ofl->ofl_flags1 |= FLG_OF1_NOSGHND;
1395                            } else if (strcmp(optarg,
1396                                MSG_ORIG(MSG_ARG_SYMBOLCAP)) == 0) {
1397                                    ofl->ofl_flags |= FLG_OF_OTOSCAP;
1398
1399                            /*
1400                             * Check archive group usage
1401                             *      -z rescan-start ... -z rescan-end
1402                             * to ensure they don't overlap and are well formed.
1403                             */
1404                            } else if (strcmp(optarg,
1405                                MSG_ORIG(MSG_ARG_RESCAN_START)) == 0) {
1406                                    if (ofl->ofl_ars_gsandx == 0) {
1407                                            ofl->ofl_ars_gsandx = ndx;
1408                                    } else if (ofl->ofl_ars_gsandx > 0) {
1409                                            /* Another group is still open */
1410                                            ld_eprintf(ofl, ERR_FATAL,
1411                                                MSG_INTL(MSG_ARG_AR_GRP_OLAP),
1412                                                MSG_INTL(MSG_MARG_AR_GRPS));
1413                                            /* Don't report cascading errors */
1414                                            ofl->ofl_ars_gsandx = -1;
1415                                    }
1416                            } else if (strcmp(optarg,
1417                                MSG_ORIG(MSG_ARG_RESCAN_END)) == 0) {
1418                                    if (ofl->ofl_ars_gsandx > 0) {
1419                                            ofl->ofl_ars_gsandx = 0;
1420                                    } else if (ofl->ofl_ars_gsandx == 0) {
1421                                            /* There was no matching begin */
1422                                            ld_eprintf(ofl, ERR_FATAL,
1423                                                MSG_INTL(MSG_ARG_AR_GRP_BAD),
1424                                                MSG_INTL(MSG_MARG_AR_GRP_END),
1425                                                MSG_INTL(MSG_MARG_AR_GRP_START));
1426                                            /* Don't report cascading errors */
1427                                            ofl->ofl_ars_gsandx = -1;
1428                                    }
1429
1430                            /*
1431                             * If -z wrap is seen, enter the symbol to be wrapped
1432                             * into the wrap AVL tree.
```

```
1433                         */
1434                        } else if (strncmp(optarg, MSG_ORIG(MSG_ARG_WRAP),
1435                            MSG_ARG_WRAP_SIZE) == 0) {
1436                                if (ld_wrap_enter(ofl,
1437                                    optarg + MSG_ARG_WRAP_SIZE) == NULL)
1438                                        return (S_ERROR);
1439                        } else if (strncmp(optarg, MSG_ORIG(MSG_ARG_ASLR),
1440                            MSG_ARG_ASLR_SIZE) == 0) {
1441                                char *p = optarg + MSG_ARG_ASLR_SIZE;
1442                                if (*p == '\0') {
1443                                        ofl->ofl_aslr = 1;
1444                                } else if (*p == '=') {
1445                                        p++;
1447                                        if (strcmp(p,
1448                                            MSG_ORIG(MSG_ARG_ENABLED)) == 0) {
1449                                                ofl->ofl_aslr = 1;
1450                                        } else if (strcmp(p,
1451                                            MSG_ORIG(MSG_ARG_DISABLED)) == 0) {
1452                                                ofl->ofl_aslr = -1;
1453                                        } else {
1454                                                ld_eprintf(ofl, ERR_FATAL,
1455                                                    MSG_INTL(MSG_ARG_ILLEGAL),
1456                                                    MSG_ORIG(MSG_ARG_ZASLR), p);
1457                                                return (S_ERROR);
1458                                        }
1459                                } else {
1460                                        ld_eprintf(ofl, ERR_FATAL,
1461                                            MSG_INTL(MSG_ARG_ILLEGAL),
1462                                            MSG_ORIG(MSG_ARG_Z), optarg);
1463                                        return (S_ERROR);
1464                                }
1465 #endif /* ! codereview */
1466                        } else if ((strncmp(optarg, MSG_ORIG(MSG_ARG_GUIDE),
1467                            MSG_ARG_GUIDE_SIZE) == 0) &&
1468                            ((optarg[MSG_ARG_GUIDE_SIZE] == '=') ||
1469                            (optarg[MSG_ARG_GUIDE_SIZE] == '\0'))) {
1470                                if (!guidance_parse(ofl, optarg))
1471                                        return (S_ERROR);
1472                        } else if (strcmp(optarg,
1473                            MSG_ORIG(MSG_ARG_FATWARN)) == 0) {
1474                                if (zfwflag  == SET_FALSE) {
1475                                        ld_eprintf(ofl, ERR_WARNING_NF,
1476                                            MSG_INTL(MSG_ARG_MTONCE),
1477                                            MSG_ORIG(MSG_ARG_ZFATWNOFATW));
1478                                } else {
1479                                        zfwflag = SET_TRUE;
1480                                        ofl->ofl_flags |= FLG_OF_FATWARN;
1481                                }
1482                        } else if (strcmp(optarg,
1483                            MSG_ORIG(MSG_ARG_NOFATWARN)) == 0) {
1484                                if (zfwflag  == SET_TRUE)
1485                                        ld_eprintf(ofl, ERR_WARNING_NF,
1486                                            MSG_INTL(MSG_ARG_MTONCE),
1487                                            MSG_ORIG(MSG_ARG_ZFATWNOFATW));
1488                                else
1489                                        zfwflag = SET_FALSE;
1491                        /*
1492                         * Process everything related to -z assert-deflib. This
1493                         * must be done in pass 1 because it gets used in pass
1494                         * 2.
1495                         */
1496                        } else if (strncmp(optarg, MSG_ORIG(MSG_ARG_ASSDEFLIB),
1497                            MSG_ARG_ASSDEFLIB_SIZE) == 0) {
1498                                if (assdeflib_parse(ofl, optarg) != TRUE)
```



```
1499                                        return (S_ERROR);
1500                        /*
1501                         * The following options just need validation as they
1502                         * are interpreted on the second pass through the
1503                         * command line arguments.
1504                         */
1505                        } else if (
1506                            strncmp(optarg, MSG_ORIG(MSG_ARG_INITARRAY),
1507                            MSG_ARG_INITARRAY_SIZE) &&
1508                            strncmp(optarg, MSG_ORIG(MSG_ARG_FINIARRAY),
1509                            MSG_ARG_FINIARRAY_SIZE) &&
1510                            strncmp(optarg, MSG_ORIG(MSG_ARG_PREINITARRAY),
1511                            MSG_ARG_PREINITARRAY_SIZE) &&
1512                            strncmp(optarg, MSG_ORIG(MSG_ARG_RTLDINFO),
1513                            MSG_ARG_RTLDINFO_SIZE) &&
1514                            strncmp(optarg, MSG_ORIG(MSG_ARG_DTRACE),
1515                            MSG_ARG_DTRACE_SIZE) &&
1516                            strcmp(optarg, MSG_ORIG(MSG_ARG_ALLEXTRT)) &&
1517                            strcmp(optarg, MSG_ORIG(MSG_ARG_DFLEXTRT)) &&
1518                            strcmp(optarg, MSG_ORIG(MSG_ARG_DIRECT)) &&
1519                            strcmp(optarg, MSG_ORIG(MSG_ARG_NODIRECT)) &&
1520                            strcmp(optarg, MSG_ORIG(MSG_ARG_GROUPPERM)) &&
1521                            strcmp(optarg, MSG_ORIG(MSG_ARG_LAZYLOAD)) &&
1522                            strcmp(optarg, MSG_ORIG(MSG_ARG_NOGROUPPERM)) &&
1523                            strcmp(optarg, MSG_ORIG(MSG_ARG_NOLAZYLOAD)) &&
1524                            strcmp(optarg, MSG_ORIG(MSG_ARG_NODEFERRED)) &&
1525                            strcmp(optarg, MSG_ORIG(MSG_ARG_RECORD)) &&
1526                            strcmp(optarg, MSG_ORIG(MSG_ARG_ALTEXEC64)) &&
1527                            strcmp(optarg, MSG_ORIG(MSG_ARG_WEAKEXT)) &&
1528                            strncmp(optarg, MSG_ORIG(MSG_ARG_TARGET),
1529                            MSG_ARG_TARGET_SIZE) &&
1530                            strcmp(optarg, MSG_ORIG(MSG_ARG_RESCAN_NOW)) &&
1531                            strcmp(optarg, MSG_ORIG(MSG_ARG_DEFERRED))) {
1532                                ld_eprintf(ofl, ERR_FATAL,
1533                                    MSG_INTL(MSG_ARG_ILLEGAL),
1534                                    MSG_ORIG(MSG_ARG_Z), optarg);
1535                        }
1537                        break;
1539                case 'D':
1540                        /*
1541                         * If we have not yet read any input files go ahead
1542                         * and process any debugging options (this allows any
1543                         * argument processing, entrance criteria and library
1544                         * initialization to be displayed).  Otherwise, if an
1545                         * input file has been seen, skip interpretation until
1546                         * process_files (this allows debugging to be turned
1547                         * on and off around individual groups of files).
1548                         */
1549                        Dflag = 1;
1550                        if (ofl->ofl_objscnt == 0) {
1551                                if (dbg_setup(ofl, optarg, 2) == 0)
1552                                        return (S_ERROR);
1553                        }
1555                        /*
1556                         * A diagnostic can only be provided after dbg_setup().
1557                         * As this is the first diagnostic that can be produced
1558                         * by ld(1), issue a title for timing and basic output.
1559                         */
1560                        if ((optitle == 0) && DBG_ENABLED) {
1561                                optitle++;
1562                                DBG_CALL(Dbg_basic_options(ofl->ofl_lml));
1563                        }
1564                        DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));
```

```
1565                            break;
1567                    case 'B':
1568                            DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));
1569                            if (strcmp(optarg, MSG_ORIG(MSG_ARG_DIRECT)) == 0) {
1570                                    if (Bdflag == SET_FALSE) {
1571                                            ld_eprintf(ofl, ERR_FATAL,
1572                                                MSG_INTL(MSG_ARG_INCOMP),
1573                                                MSG_ORIG(MSG_ARG_BNODIRECT),
1574                                                MSG_ORIG(MSG_ARG_BDIRECT));
1575                                    } else {
1576                                            Bdflag = SET_TRUE;
1577                                            ofl->ofl_guideflags |= FLG_OFG_NO_DB;
1578                                    }
1579                            } else if (strcmp(optarg,
1580                                MSG_ORIG(MSG_ARG_NODIRECT)) == 0) {
1581                                    if (Bdflag == SET_TRUE) {
1582                                            ld_eprintf(ofl, ERR_FATAL,
1583                                                MSG_INTL(MSG_ARG_INCOMP),
1584                                                MSG_ORIG(MSG_ARG_BDIRECT),
1585                                                MSG_ORIG(MSG_ARG_BNODIRECT));
1586                                    } else {
1587                                            Bdflag = SET_FALSE;
1588                                            ofl->ofl_guideflags |= FLG_OFG_NO_DB;
1589                                    }
1590                            } else if (strcmp(optarg,
1591                                MSG_ORIG(MSG_STR_SYMBOLIC)) == 0)
1592                                    Bsflag = TRUE;
1593                            else if (strcmp(optarg, MSG_ORIG(MSG_ARG_REDUCE)) == 0)
1594                                    ofl->ofl_flags |= FLG_OF_PROCRED;
1595                            else if (strcmp(optarg, MSG_ORIG(MSG_STR_LOCAL)) == 0)
1596                                    Blflag = TRUE;
1597                            else if (strcmp(optarg, MSG_ORIG(MSG_ARG_GROUP)) == 0)
1598                                    Bgflag = TRUE;
1599                            else if (strcmp(optarg,
1600                                MSG_ORIG(MSG_STR_ELIMINATE)) == 0)
1601                                    Beflag = TRUE;
1602                            else if (strcmp(optarg,
1603                                MSG_ORIG(MSG_ARG_TRANSLATOR)) == 0) {
1604                                    ld_eprintf(ofl, ERR_WARNING,
1605                                        MSG_INTL(MSG_ARG_UNSUPPORTED),
1606                                        MSG_ORIG(MSG_ARG_BTRANSLATOR));
1607                            } else if (strcmp(optarg,
1608                                MSG_ORIG(MSG_STR_LD_DYNAMIC)) &&
1609                                strcmp(optarg, MSG_ORIG(MSG_ARG_STATIC))) {
1610                                    ld_eprintf(ofl, ERR_FATAL,
1611                                        MSG_INTL(MSG_ARG_ILLEGAL),
1612                                        MSG_ORIG(MSG_ARG_CB), optarg);
1613                            }
1614                            break;
1616                    case 'G':
1617                            DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, NULL));
1618                            Gflag = TRUE;
1619                            break;
1621                    case 'L':
1622                            DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));
1623                            break;
1625                    case 'M':
1626                            DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));
1627                            if (aplist_append(&(ofl->ofl_maps), optarg,
1628                                AL_CNT_OFL_MAPFILES) == NULL)
1629                                    return (S_ERROR);
1630                            break;
```



```
1632                    case 'N':
1633                            DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));
1634                            break;
1636                    case 'Q':
1637                            DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));
1638                            if ((optarg[0] == 'n') && (optarg[1] == '\0')) {
1639                                    if (Qflag != SET_UNKNOWN)
1640                                            ld_eprintf(ofl, ERR_WARNING_NF,
1641                                                MSG_INTL(MSG_ARG_MTONCE),
1642                                                MSG_ORIG(MSG_ARG_CQ));
1643                                    else
1644                                            Qflag = SET_FALSE;
1645                            } else if ((optarg[0] == 'y') && (optarg[1] == '\0')) {
1646                                    if (Qflag != SET_UNKNOWN)
1647                                            ld_eprintf(ofl, ERR_WARNING_NF,
1648                                                MSG_INTL(MSG_ARG_MTONCE),
1649                                                MSG_ORIG(MSG_ARG_CQ));
1650                                    else
1651                                            Qflag = SET_TRUE;
1652                            } else {
1653                                    ld_eprintf(ofl, ERR_FATAL,
1654                                        MSG_INTL(MSG_ARG_ILLEGAL),
1655                                        MSG_ORIG(MSG_ARG_CQ), optarg);
1656                            }
1657                            break;
1659                    case 'S':
1660                            DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));
1661                            if (aplist_append(&lib_support, optarg,
1662                                AL_CNT_SUPPORT) == NULL)
1663                                    return (S_ERROR);
1664                            break;
1666                    case 'V':
1667                            DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, NULL));
1668                            if (!Vflag)
1669                                    (void) fprintf(stderr, MSG_ORIG(MSG_STR_STRNL),
1670                                        ofl->ofl_sgsid);
1671                            Vflag = TRUE;
1672                            break;
1674                    case 'Y':
1675                            DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));
1676                            if (strncmp(optarg, MSG_ORIG(MSG_ARG_LCOM), 2) == 0) {
1677                                    if (Llibdir)
1678                                            ld_eprintf(ofl, ERR_WARNING_NF,
1679                                                MSG_INTL(MSG_ARG_MTONCE),
1680                                                MSG_ORIG(MSG_ARG_CYL));
1681                                    else
1682                                            Llibdir = optarg + 2;
1683                            } else if (strncmp(optarg,
1684                                MSG_ORIG(MSG_ARG_UCOM), 2) == 0) {
1685                                    if (Ulibdir)
1686                                            ld_eprintf(ofl, ERR_WARNING_NF,
1687                                                MSG_INTL(MSG_ARG_MTONCE),
1688                                                MSG_ORIG(MSG_ARG_CYU));
1689                                    else
1690                                            Ulibdir = optarg + 2;
1691                            } else if (strncmp(optarg,
1692                                MSG_ORIG(MSG_ARG_PCOM), 2) == 0) {
1693                                    if (Plibpath)
1694                                            ld_eprintf(ofl, ERR_WARNING_NF,
1695                                                MSG_INTL(MSG_ARG_MTONCE),
1696                                                MSG_ORIG(MSG_ARG_CYP));
```

```
1697                                    else
1698                                            Plibpath = optarg + 2;
1699                            } else {
1700                                    ld_eprintf(ofl, ERR_FATAL,
1701                                        MSG_INTL(MSG_ARG_ILLEGAL),
1702                                        MSG_ORIG(MSG_ARG_CY), optarg);
1703                            }
1704                            break;

1706                    case '?':
1707                            DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, NULL));
1708                            /*
1709                             * If the option character is '-', we're looking at a
1710                             * long option which couldn't be translated, display a
1711                             * more useful error.
1712                             */
1713                            if (optopt == '-') {
1714                                    eprintf(ofl->ofl_lml, ERR_FATAL,
1715                                        MSG_INTL(MSG_ARG_LONG_UNKNOWN),
1716                                        argv[optind-1]);
1717                            } else {
1718                                    eprintf(ofl->ofl_lml, ERR_FATAL,
1719                                        MSG_INTL(MSG_ARG_UNKNOWN), optopt);
1720                            }
1721                            (*usage)++;
1722                            break;

1724                    default:
1725                            break;
1726                    }

1728                    /*
1729                     * Update the argument index for the next getopt() iteration.
1730                     */
1731                    ndx = optind;
1732            }
1733            return (1);
1734    }

1736    /*
1737     * Parsing options pass2 for
1738     */
1739    static uintptr_t
1740    parseopt_pass2(Ofl_desc *ofl, int argc, char **argv)
1741    {
1742            int     c, ndx = optind;

1744            while ((c = ld_getopt(ofl->ofl_lml, ndx, argc, argv)) != -1) {
1745                    Ifl_desc        *ifl;
1746                    Sym_desc        *sdp;

1748                    switch (c) {
1749                            case 'l':
1750                                    DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c,
1751                                        optarg));
1752                                    if (ld_find_library(optarg, ofl) == S_ERROR)
1753                                            return (S_ERROR);
1754                                    break;
1755                            case 'B':
1756                                    DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c,
1757                                        optarg));
1758                                    if (strcmp(optarg,
1759                                        MSG_ORIG(MSG_STR_LD_DYNAMIC)) == 0) {
1760                                        if (ofl->ofl_flags & FLG_OF_DYNAMIC)
1761                                                ofl->ofl_flags |=
1762                                                    FLG_OF_DYNLIBS;
```



```
1763                                        else {
1764                                                ld_eprintf(ofl, ERR_FATAL,
1765                                                    MSG_INTL(MSG_ARG_ST_INCOMP),
1766                                                    MSG_ORIG(MSG_ARG_BDYNAMIC));
1767                                        }
1768                                    } else if (strcmp(optarg,
1769                                        MSG_ORIG(MSG_ARG_STATIC)) == 0)
1770                                            ofl->ofl_flags &= ~FLG_OF_DYNLIBS;
1771                                    break;
1772                            case 'L':
1773                                    DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c,
1774                                        optarg));
1775                                    if (ld_add_libdir(ofl, optarg) == S_ERROR)
1776                                            return (S_ERROR);
1777                                    break;
1778                            case 'N':
1779                                    DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c,
1780                                        optarg));
1781                                    /*
1782                                     * Record DT_NEEDED string
1783                                     */
1784                                    if (!(ofl->ofl_flags & FLG_OF_DYNAMIC))
1785                                            ld_eprintf(ofl, ERR_FATAL,
1786                                                MSG_INTL(MSG_ARG_ST_INCOMP),
1787                                                MSG_ORIG(MSG_ARG_CN));
1788                                    if (((ifl = libld_calloc(1,
1789                                        sizeof (Ifl_desc))) == NULL) ||
1790                                        (aplist_append(&ofl->ofl_sos, ifl,
1791                                        AL_CNT_OFL_LIBS) == NULL))
1792                                            return (S_ERROR);

1794                                    ifl->ifl_name = MSG_INTL(MSG_STR_COMMAND);
1795                                    ifl->ifl_soname = optarg;
1796                                    ifl->ifl_flags = (FLG_IF_NEEDSTR |
1797                                        FLG_IF_FILEREF | FLG_IF_DEPREQD);

1799                                    break;
1800                            case 'D':
1801                                    DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c,
1802                                        optarg));
1803                                    (void) dbg_setup(ofl, optarg, 3);
1804                                    break;
1805                            case 'u':
1806                                    DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c,
1807                                        optarg));
1808                                    if (ld_sym_add_u(optarg, ofl,
1809                                        MSG_STR_COMMAND) == (Sym_desc *)S_ERROR)
1810                                            return (S_ERROR);
1811                                    break;
1812                            case 'z':
1813                                    DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c,
1814                                        optarg));
1815                                    if ((strncmp(optarg, MSG_ORIG(MSG_ARG_LD32),
1816                                        MSG_ARG_LD32_SIZE) == 0) ||
1817                                        (strncmp(optarg, MSG_ORIG(MSG_ARG_LD64),
1818                                        MSG_ARG_LD64_SIZE) == 0)) {
1819                                            if (createargv(ofl, 0) == S_ERROR)
1820                                                    return (S_ERROR);
1821                                    } else if (strcmp(optarg,
1822                                        MSG_ORIG(MSG_ARG_ALLEXTRT)) == 0) {
1823                                            ofl->ofl_flags1 |= FLG_OF1_ALLEXRT;
1824                                            ofl->ofl_flags1 &= ~FLG_OF1_WEAKEXT;
1825                                    } else if (strcmp(optarg,
1826                                        MSG_ORIG(MSG_ARG_WEAKEXT)) == 0) {
1827                                            ofl->ofl_flags1 |= FLG_OF1_WEAKEXT;
1828                                            ofl->ofl_flags1 &= ~FLG_OF1_ALLEXRT;
```

```
1829                                } else if (strcmp(optarg,
1830                                    MSG_ORIG(MSG_ARG_DFLEXTRT)) == 0) {
1831                                        ofl->ofl_flags1 &=
1832                                            ~(FLG_OF1_ALLEXRT |
1833                                            FLG_OF1_WEAKEXT);
1834                                } else if (strcmp(optarg,
1835                                    MSG_ORIG(MSG_ARG_DIRECT)) == 0) {
1836                                        ofl->ofl_flags1 |= FLG_OF1_ZDIRECT;
1837                                        ofl->ofl_guideflags |= FLG_OFG_NO_DB;
1838                                } else if (strcmp(optarg,
1839                                    MSG_ORIG(MSG_ARG_NODIRECT)) == 0) {
1840                                        ofl->ofl_flags1 &= ~FLG_OF1_ZDIRECT;
1841                                        ofl->ofl_guideflags |= FLG_OFG_NO_DB;
1842                                } else if (strcmp(optarg,
1843                                    MSG_ORIG(MSG_ARG_IGNORE)) == 0) {
1844                                        ofl->ofl_flags1 |= FLG_OF1_IGNORE;
1845                                } else if (strcmp(optarg,
1846                                    MSG_ORIG(MSG_ARG_RECORD)) == 0) {
1847                                        ofl->ofl_flags1 &= ~FLG_OF1_IGNORE;
1848                                } else if (strcmp(optarg,
1849                                    MSG_ORIG(MSG_ARG_LAZYLOAD)) == 0) {
1850                                        ofl->ofl_flags1 |= FLG_OF1_LAZYLD;
1851                                        ofl->ofl_guideflags |= FLG_OFG_NO_LAZY;
1852                                } else if (strcmp(optarg,
1853                                    MSG_ORIG(MSG_ARG_NOLAZYLOAD)) == 0) {
1854                                        ofl->ofl_flags1 &= ~ FLG_OF1_LAZYLD;
1855                                        ofl->ofl_guideflags |= FLG_OFG_NO_LAZY;
1856                                } else if (strcmp(optarg,
1857                                    MSG_ORIG(MSG_ARG_GROUPPERM)) == 0) {
1858                                        ofl->ofl_flags1 |= FLG_OF1_GRPPRM;
1859                                } else if (strcmp(optarg,
1860                                    MSG_ORIG(MSG_ARG_NOGROUPPERM)) == 0) {
1861                                        ofl->ofl_flags1 &= ~FLG_OF1_GRPPRM;
1862                                } else if (strncmp(optarg,
1863                                    MSG_ORIG(MSG_ARG_INITARRAY),
1864                                    MSG_ARG_INITARRAY_SIZE) == 0) {
1865                                        if (((sdp = ld_sym_add_u(optarg +
1866                                            MSG_ARG_INITARRAY_SIZE, ofl,
1867                                            MSG_STR_COMMAND)) ==
1868                                            (Sym_desc *)S_ERROR) ||
1869                                            (aplist_append(&ofl->ofl_initarray,
1870                                            sdp, AL_CNT_OFL_ARRAYS) == NULL))
1871                                                return (S_ERROR);
1872                                } else if (strncmp(optarg,
1873                                    MSG_ORIG(MSG_ARG_FINIARRAY),
1874                                    MSG_ARG_FINIARRAY_SIZE) == 0) {
1875                                        if (((sdp = ld_sym_add_u(optarg +
1876                                            MSG_ARG_FINIARRAY_SIZE, ofl,
1877                                            MSG_STR_COMMAND)) ==
1878                                            (Sym_desc *)S_ERROR) ||
1879                                            (aplist_append(&ofl->ofl_finiarray,
1880                                            sdp, AL_CNT_OFL_ARRAYS) == NULL))
1881                                                return (S_ERROR);
1882                                } else if (strncmp(optarg,
1883                                    MSG_ORIG(MSG_ARG_PREINITARRAY),
1884                                    MSG_ARG_PREINITARRAY_SIZE) == 0) {
1885                                        if (((sdp = ld_sym_add_u(optarg +
1886                                            MSG_ARG_PREINITARRAY_SIZE, ofl,
1887                                            MSG_STR_COMMAND)) ==
1888                                            (Sym_desc *)S_ERROR) ||
1889                                            (aplist_append(&ofl->ofl_preiarray,
1890                                            sdp, AL_CNT_OFL_ARRAYS) == NULL))
1891                                                return (S_ERROR);
1892                                } else if (strncmp(optarg,
1893                                    MSG_ORIG(MSG_ARG_RTLDINFO),
1894                                    MSG_ARG_RTLDINFO_SIZE) == 0) {
```



```
1895                                        if (((sdp = ld_sym_add_u(optarg +
1896                                            MSG_ARG_RTLDINFO_SIZE, ofl,
1897                                            MSG_STR_COMMAND)) ==
1898                                            (Sym_desc *)S_ERROR) ||
1899                                            (aplist_append(&ofl->ofl_rtldinfo,
1900                                            sdp, AL_CNT_OFL_ARRAYS) == NULL))
1901                                                return (S_ERROR);
1902                                } else if (strncmp(optarg,
1903                                    MSG_ORIG(MSG_ARG_DTRACE),
1904                                    MSG_ARG_DTRACE_SIZE) == 0) {
1905                                        if ((sdp = ld_sym_add_u(optarg +
1906                                            MSG_ARG_DTRACE_SIZE, ofl,
1907                                            MSG_STR_COMMAND)) ==
1908                                            (Sym_desc *)S_ERROR)
1909                                                return (S_ERROR);
1910                                        ofl->ofl_dtracesym = sdp;
1911                                } else if (strcmp(optarg,
1912                                    MSG_ORIG(MSG_ARG_RESCAN_NOW)) == 0) {
1913                                        if (ld_rescan_archives(ofl, 0, ndx) ==
1914                                            S_ERROR)
1915                                                return (S_ERROR);
1916                                } else if (strcmp(optarg,
1917                                    MSG_ORIG(MSG_ARG_RESCAN_START)) == 0) {
1918                                        ofl->ofl_ars_gsndx = ofl->ofl_arscnt;
1919                                        ofl->ofl_ars_gsandx = ndx;
1920                                } else if (strcmp(optarg,
1921                                    MSG_ORIG(MSG_ARG_RESCAN_END)) == 0) {
1922                                        if (ld_rescan_archives(ofl, 1, ndx) ==
1923                                            S_ERROR)
1924                                                return (S_ERROR);
1925                                } else if (strcmp(optarg,
1926                                    MSG_ORIG(MSG_ARG_DEFERRED)) == 0) {
1927                                        ofl->ofl_flags1 |= FLG_OF1_DEFERRED;
1928                                } else if (strcmp(optarg,
1929                                    MSG_ORIG(MSG_ARG_NODEFERRED)) == 0) {
1930                                        ofl->ofl_flags1 &= ~FLG_OF1_DEFERRED;
1931                                }
1932                        default:
1933                                break;
1934                }

1936                /*
1937                 * Update the argument index for the next getopt() iteration.
1938                 */
1939                ndx = optind;
1940        }
1941        return (1);
1942 }

1944 /*
1945  *
1946  * Pass 1 -- process_flags: collects all options and sets flags
1947  */
1948 static uintptr_t
1949 process_flags_com(Ofl_desc *ofl, int argc, char **argv, int *usage)
1950 {
1951         for (; optind < argc; optind++) {
1952                 /*
1953                  * If we detect some more options return to getopt().
1954                  * Checking argv[optind][1] against null prevents a forever
1955                  * loop if an unadorned '-' argument is passed to us.
1956                  */
1957                 while ((optind < argc) && (argv[optind][0] == '-')) {
1958                         if (argv[optind][1] != '\0') {
1959                                 if (parseopt_pass1(ofl, argc, argv,
1960                                     usage) == S_ERROR)
```

```
1961                                        return (S_ERROR);
1962                        } else if (++optind < argc)
1963                                continue;
1964                }
1965                if (optind >= argc)
1966                        break;
1967                ofl->ofl_objscnt++;
1968        }

1970        /* Did an unterminated archive group run off the end? */
1971        if (ofl->ofl_ars_gsandx > 0) {
1972                ld_eprintf(ofl, ERR_FATAL, MSG_INTL(MSG_ARG_AR_GRP_BAD),
1973                    MSG_INTL(MSG_MARG_AR_GRP_START),
1974                    MSG_INTL(MSG_MARG_AR_GRP_END));
1975                return (S_ERROR);
1976        }

1978        return (1);
1979 }

1981 uintptr_t
1982 ld_process_flags(Ofl_desc *ofl, int argc, char **argv)
1983 {
1984        int     usage = 0;      /* Collect all argument errors before exit */

1986        if (argc < 2) {
1987                usage_mesg(FALSE);
1988                return (S_ERROR);
1989        }

1991        /*
1992         * Option handling
1993         */
1994        opterr = 0;
1995        optind = 1;
1996        if (process_flags_com(ofl, argc, argv, &usage) == S_ERROR)
1997                return (S_ERROR);

1999        /*
2000         * Having parsed everything, did we have any usage errors.
2001         */
2002        if (usage) {
2003                eprintf(ofl->ofl_lml, ERR_FATAL, MSG_INTL(MSG_ARG_USEHELP));
2004                return (S_ERROR);
2005        }

2007        return (check_flags(ofl, argc));
2008 }

2010 /*
2011  * Pass 2 -- process_files: skips the flags collected in pass 1 and processes
2012  * files.
2013  */
2014 static uintptr_t
2015 process_files_com(Ofl_desc *ofl, int argc, char **argv)
2016 {
2017        for (; optind < argc; optind++) {
2018                int             fd;
2019                uintptr_t       open_ret;
2020                char            *path;
2021                Rej_desc        rej = { 0 };

2023                /*
2024                 * If we detect some more options return to getopt().
2025                 * Checking argv[optind][1] against null prevents a forever
2026                 * loop if an unadorned '-' argument is passed to us.
```



```
2027                 */
2028                while ((optind < argc) && (argv[optind][0] == '-')) {
2029                        if (argv[optind][1] != '\0') {
2030                                if (parseopt_pass2(ofl, argc, argv) == S_ERROR)
2031                                        return (S_ERROR);
2032                        } else if (++optind < argc)
2033                                continue;
2034                }
2035                if (optind >= argc)
2036                        break;

2038                path = argv[optind];
2039                if ((fd = open(path, O_RDONLY)) == -1) {
2040                        int err = errno;

2042                        ld_eprintf(ofl, ERR_FATAL,
2043                            MSG_INTL(MSG_SYS_OPEN), path, strerror(err));
2044                        continue;
2045                }

2047                DBG_CALL(Dbg_args_file(ofl->ofl_lml, optind, path));

2049                open_ret = ld_process_open(path, path, &fd, ofl,
2050                    (FLG_IF_CMDLINE | FLG_IF_NEEDED), &rej, NULL);
2051                if (fd != -1)
2052                        (void) close(fd);
2053                if (open_ret == S_ERROR)
2054                        return (S_ERROR);

2056                /*
2057                 * Check for mismatched input.
2058                 */
2059                if (rej.rej_type) {
2060                        Conv_reject_desc_buf_t rej_buf;

2062                        ld_eprintf(ofl, ERR_FATAL,
2063                            MSG_INTL(reject[rej.rej_type]),
2064                            rej.rej_name ? rej.rej_name :
2065                            MSG_INTL(MSG_STR_UNKNOWN),
2066                            conv_reject_desc(&rej, &rej_buf,
2067                            ld_targ.t_m.m_mach));
2068                        return (1);
2069                }
2070        }
2071        return (1);
2072 }

2074 uintptr_t
2075 ld_process_files(Ofl_desc *ofl, int argc, char **argv)
2076 {
2077        DBG_CALL(Dbg_basic_files(ofl->ofl_lml));

2079        /*
2080         * Process command line files (taking into account any applicable
2081         * preceding flags).  Return if any fatal errors have occurred.
2082         */
2083        opterr = 0;
2084        optind = 1;
2085        if (process_files_com(ofl, argc, argv) == S_ERROR)
2086                return (S_ERROR);
2087        if (ofl->ofl_flags & FLG_OF_FATAL)
2088                return (1);

2090        /*
2091         * Guidance: Use -B direct/nodirect or -z direct/nodirect.
2092         *
```

```
2093          * This is a backstop for the case where the link had no dependencies.
2094          * Otherwise, it will get caught by ld_process_ifl(). We need both,
2095          * because -z direct is positional, and its value at the time where
2096          * the first dependency is seen might be different than it is now.
2097          */
2098         if ((ofl->ofl_flags & FLG_OF_DYNAMIC) &&
2099             OFL_GUIDANCE(ofl, FLG_OFG_NO_DB)) {
2100                 ld_eprintf(ofl, ERR_GUIDANCE, MSG_INTL(MSG_GUIDE_DIRECT));
2101                 ofl->ofl_guideflags |= FLG_OFG_NO_DB;
2102         }

2104         /*
2105          * Now that all command line files have been processed see if there are
2106          * any additional 'needed' shared object dependencies.
2107          */
2108         if (ofl->ofl_soneed)
2109                 if (ld_finish_libs(ofl) == S_ERROR)
2110                         return (S_ERROR);

2112         /*
2113          * If rescanning archives is enabled, do so now to determine whether
2114          * there might still be members extracted to satisfy references from any
2115          * explicit objects.  Continue until no new objects are extracted.  Note
2116          * that this pass is carried out *after* processing any implicit objects
2117          * (above) as they may already have resolved any undefined references
2118          * from any explicit dependencies.
2119          */
2120         if (ofl->ofl_flags1 & FLG_OF1_RESCAN) {
2121                 if (ld_rescan_archives(ofl, 0, argc) == S_ERROR)
2122                         return (S_ERROR);
2123                 if (ofl->ofl_flags & FLG_OF_FATAL)
2124                         return (1);
2125         }

2127         /*
2128          * If debugging, provide statistics on each archives extraction, or flag
2129          * any archive that has provided no members.  Note that this could be a
2130          * nice place to free up much of the archive infrastructure, as we've
2131          * extracted any members we need.  However, as we presently don't free
2132          * anything under ld(1) there's not much point in proceeding further.
2133          */
2134         DBG_CALL(Dbg_statistics_ar(ofl));

2136         /*
2137          * If any version definitions have been established, either via input
2138          * from a mapfile or from the input relocatable objects, make sure any
2139          * version dependencies are satisfied, and version symbols created.
2140          */
2141         if (ofl->ofl_verdesc)
2142                 if (ld_vers_check_defs(ofl) == S_ERROR)
2143                         return (S_ERROR);

2145         /*
2146          * If input section ordering was specified within some segment
2147          * using a mapfile, verify that the expected sections were seen.
2148          */
2149         if (ofl->ofl_flags & FLG_OF_IS_ORDER)
2150                 ld_ent_check(ofl);

2152         return (1);
2153 }

2155 uintptr_t
2156 ld_init_strings(Ofl_desc *ofl)
2157 {
2158         uint_t  stflags;
```



```
2160         if (ofl->ofl_flags1 & FLG_OF1_NCSTTAB)
2161                 stflags = 0;
2162         else
2163                 stflags = FLG_STNEW_COMPRESS;

2165         if (((ofl->ofl_shdrsttab = st_new(stflags)) == NULL) ||
2166             ((ofl->ofl_strtab = st_new(stflags)) == NULL) ||
2167             ((ofl->ofl_dynstrtab = st_new(stflags)) == NULL))
2168                 return (S_ERROR);

2170         return (0);
2171 }
```

```
**********************************************************
   60530 Wed May 27 19:49:04 2015
new/usr/src/cmd/sgs/libld/common/libld.msg
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
```

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#

#
# Copyright (c) 1995, 2010, Oracle and/or its affiliates. All rights reserved.
#

#
# Copyright (c) 2012, Joyent, Inc.  All rights reserved.
#

@ _START_

# Message file for cmd/sgs/libld.

@ MSG_ID_LIBLD

#
# TRANSLATION_NOTE -- Beginning of USAGE message
#       The following messages are the usage messages for the ld command.
#       Tab characters (\t) are used to align the messages.
#
#       Each usage message starts with \t, and if the message has more than one
#               line, the following messages are aligned by 3 tab characters.
#               When you see \n\t\t\t, the first \n is used to change the line,
#               and following 3 tab characters are used to align the line.
#
#       Each usage message option is surrounded by [ and ].  Then the
#       description of the option follows. The descriptions should be aligned,
#       so tab characters are padded as needed after the closing bracket ].
#
#       How to align the messages are up to the translators and the
#       localization engineers.
#
#       In C locale, the first 3 messages would look like the following:
#
#       usage: ld [-6:abc:.....] file(s)
#               [-a]            create an absolute file
#               [-b]            do not do special PIC relocations in a.out
#               [-c file]       record configuration 'file'
```



```
#
@ MSG_ARG_USAGE         "usage: ld [-%s] file(s)\n"
@ MSG_ARG_DETAIL_3      "\t[-32]\t\tenforce a 32-bit link-edit\n"
@ MSG_ARG_DETAIL_6      "\t[-64]\t\tenforce a 64-bit link-edit\n"
@ MSG_ARG_DETAIL_A      "\t[-a]\t\tcreate an absolute file\n"
@ MSG_ARG_DETAIL_B      "\t[-b]\t\tdo not do special PIC relocations in a.out\n"
@ MSG_ARG_DETAIL_CBDR   "\t[-B direct | nodirect]\n\
                         \t\t\testablish direct bindings, or inhibit direct \
                         binding\n\
                         \t\t\tto, the object being created\n"
@ MSG_ARG_DETAIL_CBDY   "\t[-B dynamic | static]\n\
                         \t\t\tsearch for shared libraries|archives\n"
@ MSG_ARG_DETAIL_CBE    "\t[-B eliminate]\teliminate unqualified global \
                         symbols from the\n\t\t\tsymbol table\n"
@ MSG_ARG_DETAIL_CBG    "\t[-B group]\trelocate object from within group\n"
@ MSG_ARG_DETAIL_CBL    "\t[-B local]\treduce unqualified global symbols to \
                         local\n"
@ MSG_ARG_DETAIL_CBR    "\t[-B reduce]\tprocess symbol reductions\n"
@ MSG_ARG_DETAIL_CBS    "\t[-B symbolic]\tbind external references to \
                         definitions when creating\n\
                         \t\t\tshared objects\n"
@ MSG_ARG_DETAIL_C      "\t[-c name]\trecord configuration file 'name'\n"
@ MSG_ARG_DETAIL_CC     "\t[-C]\t\tdemangle C++ symbol name diagnostics\n"
@ MSG_ARG_DETAIL_D      "\t[-d y | n]\toperate in dynamic|static mode\n"
@ MSG_ARG_DETAIL_CD     "\t[-D token,...]\tprint diagnostic messages\n"
@ MSG_ARG_DETAIL_E      "\t[-e epsym], [--entry epsym]\n\
                         \t\t\tuse 'epsym' as entry point address\n"
@ MSG_ARG_DETAIL_F      "\t[-f name], [--auxiliary name]\n\
                         \t\t\tspecify library for which this file is an \
                         auxiliary\n\t\t\tfilter\n"
@ MSG_ARG_DETAIL_CF     "\t[-F name], [--filter name]\n\
                         \t\t\tspecify library for which this file is a filter\n
@ MSG_ARG_DETAIL_CG     "\t[-G], [-shared]\n\
                         \t\t\tcreate a shared object\n"
@ MSG_ARG_DETAIL_H      "\t[-h name], [--soname name]\n\
                         \t\t\tuse 'name' as internal shared object identifier\n
@ MSG_ARG_DETAIL_I      "\t[-i]\t\tignore LD_LIBRARY_PATH setting\n"
@ MSG_ARG_DETAIL_CI     "\t[-I name]\tuse 'name' as path of interpreter\n"
@ MSG_ARG_DETAIL_L      "\t[-l x], [--library x]\n\
                         \t\t\tsearch for libx.so or libx.a\n"
@ MSG_ARG_DETAIL_CL     "\t[-L path], [--library-path path]\n\
                         \t\t\tsearch for libraries in directory 'path'\n"
@ MSG_ARG_DETAIL_M      "\t[-m]\t\tprint memory map\n"
@ MSG_ARG_DETAIL_CM     "\t[-M mapfile]\tuse processing directives contained \
                         in 'mapfile'\n"
@ MSG_ARG_DETAIL_CN     "\t[-N string]\tcreate a dynamic dependency for \
                         'string'\n"
@ MSG_ARG_DETAIL_O      "\t[-o outfile], [--output outfile]\n\
                         \t\t\tname the output file 'outfile'\n"
@ MSG_ARG_DETAIL_P      "\t[-p auditlib]\tidentify audit library to accompany \
                         this object\n"
@ MSG_ARG_DETAIL_CP     "\t[-P auditlib]\tidentify audit library for \
                         processing the dependencies\n\
                         \t\t\tof this object\n"
@ MSG_ARG_DETAIL_CQ     "\t[-Q y | n]\tdo|do not place version information in \
                         output file\n"
@ MSG_ARG_DETAIL_R      "\t[-r], [--relocatable]\n\
                         \t\t\tcreate a relocatable object\n"
@ MSG_ARG_DETAIL_CR     "\t[-R path], [-rpath path]\n\
                         \t\t\tspecify a library search path to be used at run \
                         time\n"
@ MSG_ARG_DETAIL_S      "\t[-s], [--strip-all]\n\
                         \t\t\tstrip any symbol and debugging information\n"
@ MSG_ARG_DETAIL_CS     "\t[-S supportlib]\n\
                         \t\t\tspecify a link-edit support library\n"
@ MSG_ARG_DETAIL_T      "\t[-t]\t\tdo not warn of multiply-defined symbols \
```

```
                        that have\n\t\t\tdifferent sizes or alignments\n"
@ MSG_ARG_DETAIL_U      "\t[-u symname], [--undefined symname]\n\
                         \t\t\tcreate an undefined symbol 'symname'\n"
@ MSG_ARG_DETAIL_CV     "\t[-V], [--version]\n\
                         \t\t\tprint version information\n"
@ MSG_ARG_DETAIL_CY     "\t[-Y P,dirlist]\tuse 'dirlist' as a default path \
                         when searching for\n\
                         \t\t\tlibraries\n"
@ MSG_ARG_DETAIL_ZA     "\t[-z absexec]\twhen building an executable absolute \
                         symbols\n \
                         \t\t\treferenced in dynamic objects are promoted to\n \
                         \t\t\tthe executable\n"
@ MSG_ARG_DETAIL_ZAE    "\t[-z allextract | defaultextract | weakextract],\n\
                         \t[--whole-archive | --no-whole-archive]\n\
                         \t\t\textract all member files, only members that \
                         resolve\n\
                         \t\t\tundefined or tentative symbols, or \
                         allow extraction of\n\
                         \t\t\tarchive members to resolve weak references from \
                         \n\t\t\t\tarchive files\n"
@ MSG_ARG_DETAIL_ZAL    "\t[-z altexec64]\texecute the 64-bit link-editor\n"
@ MSG_ARG_DETAIL_ZADLIB "\t[-z assert-deflib]\n\
                        \t\t\tenables warnings for linking with libraries in \
                        the \n\t\t\tdefault search path\n\
                        \t[-z assert-deflib=libname]\n\
                        \t\t\tenables warnings for linking with libraries in \
                        the \n\t\t\tdefault search path, but 'libname' is exempt
@ MSG_ARG_DETAIL_ZC     "\t[-z combreloc | nocombreloc]\n\
                         \t\t\tcombine|do not combine multiple relocation \
                         sections\n"
@ MSG_ARG_DETAIL_ZNC    "\t[-z nocompstrtab]\n\t\t\tdisable compression of \
                         string tables\n"
@ MSG_ARG_DETAIL_ZDEF   "\t[-z deferred | nodeferred]\n\
                         \t\t\tenable|disable deferred identification of \
                         shared object\n\t\t\tdependencies\n"
@ MSG_ARG_DETAIL_ZDFS   "\t[-z defs], [--no-undefined]\n\
                         \t\t\tdisallow undefined symbol references\n"
@ MSG_ARG_DETAIL_ZDRS   "\t[-z direct | nodirect]\n\
                         \t\t\tenable|disable direct binding to shared object\n\
                         \t\t\tdependencies\n"
@ MSG_ARG_DETAIL_ZE     "\t[-z endfiltee]\tmarks a filtee such that it will \
                         terminate a filters\n\t\t\tsearch\n"
@ MSG_ARG_DETAIL_ZFATW  "\t[-z fatal-warnings | nofatal-warnings],\n\
                         \t[--fatal-warnings | --no-fatal-warnings]\n\
                         \t\tenable|disable treatment of warnings as fatal\n"
@ MSG_ARG_DETAIL_ZFA    "\t[-z finiarray=function]\n\
                         \t\t\tname of function to be appended to the \
                         .fini_array\n"
@ MSG_ARG_DETAIL_ZGP    "\t[-z groupperm | nogroupperm]\n\
                         \t\t\tenable|disable setting of group permissions\n\
                         \t\t\ton dynamic dependencies\n"
@ MSG_ARG_DETAIL_ZGUIDE "\t[-z guidance | -z guidance=item1,item2,...]\n\
                         \t\t\tenable guidance warnings. items: \
                         noall, nodefs,\n\
                         \t\t\tnodirect, nolazyload, nomapfile, notext, \
                         nounused\n"
@ MSG_ARG_DETAIL_ZH     "\t[-z help], [--help]\n\
                         \t\t\tprint this usage message\n"
@ MSG_ARG_DETAIL_ZIG    "\t[-z ignore | record]\n\
                         \t\t\tignore|record unused dynamic dependencies\n"
@ MSG_ARG_DETAIL_ZINA   "\t[-z initarray=function]\n\
                         \t\t\tname of function to be appended to the \
                         .init_array\n"
@ MSG_ARG_DETAIL_ZINI   "\t[-z initfirst]\tmark object to indicate that its \
                         .init section should\n\
                         \t\t\tbe executed before the .init section of any \
```



```
                         other\n\t\t\tobjects\n"
@ MSG_ARG_DETAIL_ZINT   "\t[-z interpose]\
                         \tdynamic object is to be an 'interposer' on direct\n\
                         \t\t\tbindings\n"
@ MSG_ARG_DETAIL_ZLAZY  "\t[-z lazyload | nolazyload]\n\
                         \t\tenable|disable delayed loading of shared \
                         object\n\t\tdependencies\n"
@ MSG_ARG_DETAIL_ZLD32  "\t[-z ld32=arg1,arg2,...]\n\
                         \t\tdefine arguments applicable to the \
                         32-bit class of ld(1)\n"
@ MSG_ARG_DETAIL_ZLD64  "\t[-z ld64=arg1,arg2,...]\n\
                         \t\tdefine arguments applicable to the \
                         64-bit class of ld(1)\n"
@ MSG_ARG_DETAIL_ZLO    "\t[-z loadfltr]\tmark filter as requiring immediate \
                         loading of its\n\
                         \t\tfiltees at runtime\n"
@ MSG_ARG_DETAIL_ZM     "\t[-z muldefs], [--allow-multiple-definition]\n\
                         \t\tallow multiply-defined symbols\n"
@ MSG_ARG_DETAIL_ZNDFS  "\t[-z nodefs]\tallow undefined symbol references\n"
@ MSG_ARG_DETAIL_ZNDEF  "\t[-z nodefaultlib]\n\
                         \t\tmark object to ignore any default library \
                         search path\n"
@ MSG_ARG_DETAIL_ZNDEL  "\t[-z nodelete]\tmark object as non-deletable\n"
@ MSG_ARG_DETAIL_ZNDLO  "\t[-z nodlopen]\tmark object as non-dlopen()'able\n"
@ MSG_ARG_DETAIL_ZNDU   "\t[-z nodump]\tmark object as non-dldump()'able\n"
@ MSG_ARG_DETAIL_ZNLD   "\t[-z noldynsym]\tdo not add a .SUNW_ldynsym section\n"
@ MSG_ARG_DETAIL_ZNPA   "\t[-z nopartial]\texpand any partially initialized \
                         symbols\n"
@ MSG_ARG_DETAIL_ZNV    "\t[-z noversion]\tdo not record any version sections\n"
@ MSG_ARG_DETAIL_ZNOW   "\t[-z now]\tmark object as requiring non-lazy \
                         binding\n"
@ MSG_ARG_DETAIL_ZO     "\t[-z origin]\tmark object as requiring $ORIGIN \
                         processing\n"
@ MSG_ARG_DETAIL_ZPIA   "\t[-z preinitarray=function]\n\
                         \t\tname of function to be appended to the \
                         .preinit_array\n"
@ MSG_ARG_DETAIL_ZRL    "\t[-z redlocsym]\treduce local syms in .symtab to \
                         a minimum\n"
@ MSG_ARG_DETAIL_ZRREL  "\t[-z relaxreloc]\trelax rules used for relocations \
                         against COMDAT sections\n"
@ MSG_ARG_DETAIL_ZRS    "\t[-z rescan]\tafter processing all arguments, rescan \
                         archive list\n\
                         \t\tuntil no further member extraction occurs\n"
@ MSG_ARG_DETAIL_ZRSN   "\t[-z rescan-now]\timmediately rescan archive list \
                         until\n\
                         \t\tno further member extraction occurs\n"
@ MSG_ARG_DETAIL_ZRSGRP "\t[-z rescan-start archives... -z rescan-end],\n\
                         \t[--start-group archives... --end-group], \
                         [-( archives... -)]\n\
                         \t\trescan specified archive group upon reaching\n\
                         \t\tthe end of the group, until no further\n\
                         \t\tmember extraction occurs\n"
@ MSG_ARG_DETAIL_ZSCAP  "\t[-z symbolcap]\tconvert object capabilities to \
                         symbol capabilities\n"
@ MSG_ARG_DETAIL_ZTARG  "\t[-z target=platform]\n\
                         \t\ttarget machine for cross linking\n"
@ MSG_ARG_DETAIL_ZT     "\t[-z text]\tdisallow output relocations against \
                         text\n"
@ MSG_ARG_DETAIL_ZTO    "\t[-z textoff]\tallow output relocations against \
                         text\n"
@ MSG_ARG_DETAIL_ZTW    "\t[-z textwarn]\twarn if there are relocations \
                         against text\n"
@ MSG_ARG_DETAIL_ZWRAP  "\t[-z wrap=symbol], [-wrap=symbol], [--wrap=symbol]\n\
                         \t\twrap symbol references\n"
@ MSG_ARG_DETAIL_ZVER   "\t[-z verbose]\t\
                         generate warnings for suspicious processings\n"
```

```
258 #
259 # TRANSLATION_NOTE -- End of USAGE message
260 #
261 @ MSG_GRP_INVALNDX      "file %s: group section [%u]%s: entry %d: \
262                          invalid section index: %d"
263 @ MSG_GRP_INVALSYM      "file %s: group section [%u]%s: invalid group symbol %s"

265 # Relocation processing messages (some of these are required to satisfy
266 # do_reloc(), which is common code used by cmd/sgs/rtld - make sure both
267 # message files remain consistent).

269 @ MSG_REL_NOFIT         "relocation error: %s: file %s: symbol %s: \
270                          value 0x%llx does not fit"
271 @ MSG_REL_NONALIGN      "relocation error: %s: file %s: symbol %s: \
272                          offset 0x%llx is non-aligned"
273 @ MSG_REL_NULL          "relocation error: file %s: section [%u]%s: \
274                          skipping null relocation record"
275 @ MSG_REL_NOTSUP        "relocation error: %s: file %s: section [%u]%s: \
276                          relocation not currently supported"
277 @ MSG_REL_PICREDLOC     "relocation error: %s: file %s symbol %s: \
278                          -z redlocsym may not be used for pic code"
279 @ MSG_REL_TLSLE         "relocation error: %s: file %s: symbol %s: \
280                          relocation illegal when building a shared object"
281 @ MSG_REL_TLSBND        "relocation error: %s: file %s: symbol %s: \
282                          bound to: %s: relocation illegal when not bound \
283                          to object being created"
284 @ MSG_REL_TLSSTAT       "relocation error: %s: file %s: symbol %s: \
285                          relocation illegal when building a static object"
286 @ MSG_REL_TLSBADSYM     "relocation error: %s: file %s: symbol %s: \
287                          bad symbol type %s: symbol type must be TLS"
288 @ MSG_REL_BADTLS        "relocation error: %s: file %s: symbol %s: \
289                          relocation illegal for TLS symbol"
290 @ MSG_REL_BADGOTBASED   "relocation error: %s: file %s: symbol %s: a GOT \
291                          relative relocation must reference a local symbol"
292 @ MSG_REL_UNKNWSYM      "relocation error: %s: file %s: section [%u]%s: \
293                          attempt to relocate with respect to unknown \
294                          symbol %s: offset 0x%llx, symbol index %d"
295 @ MSG_REL_UNSUPSZ       "relocation error: %s: file %s: symbol %s: \
296                          offset size (%d bytes) is not supported"
297 @ MSG_REL_INVALOFFSET   "relocation error: %s: file %s section [%u]%s: \
298                          invalid offset symbol '%s': offset 0x%llx"
299 @ MSG_REL_INVALRELT     "relocation error: file %s: section [%u]%s: \
300                          invalid relocation type: 0x%x"
301 @ MSG_REL_EMPTYSEC      "relocation error: %s: file %s: symbol %s: \
302                          attempted against empty section [%u]%s"
303 @ MSG_REL_EXTERNSYM     "relocation error: %s: file %s: symbol %s: \
304                          external symbolic relocation against non-allocatable \
305                          section %s; cannot be processed at runtime: \
306                          relocation ignored"
307 @ MSG_REL_UNEXPREL      "relocation error: %s: file %s: symbol %s: \
308                          unexpected relocation; generic processing performed"
309 @ MSG_REL_UNEXPSYM      "relocation error: %s: file %s: symbol %s: \
310                          unexpected symbol referenced from file %s"
311 @ MSG_REL_SYMDISC       "relocation error: %s: file %s: section [%u]%s: \
312                          symbol %s: symbol has been discarded with discarded \
313                          section: [%u]%s"
314 @ MSG_REL_NOSYMBOL      "relocation error: %s: file %s: section: [%u]%s: \
315                          offset: 0x%llx: relocation requires reference symbol"
316 @ MSG_REL_DISPREL1      "relocation error: %s: file %s: symbol %s: \
317                          displacement relocation applied to the symbol \
318                          %s at 0x%llx: symbol %s is a copy relocated symbol"
319 @ MSG_REL_UNSUPSIZE     "relocation error: %s: file %s: section [%u]%s: \
320                          relocation against section symbol unsupported"

322 @ MSG_REL_DISPREL2      "relocation warning: %s: file %s: symbol %s: \
```



```
323                          may contain displacement relocation"
324 @ MSG_REL_DISPREL3      "relocation warning: %s: file %s: symbol %s: \
325                          displacement relocation applied to the symbol \
326                          %s: at 0x%llx: displacement relocation will not be \
327                          visible in output image"
328 @ MSG_REL_DISPREL4      "relocation warning: %s: file %s: symbol %s: \
329                          displacement relocation to be applied to the symbol \
330                          %s: at 0x%llx: displacement relocation will be \
331                          visible in output image"
332 @ MSG_REL_COPY          "relocation warning: %s: file %s: symbol %s: \
333                          relocation bound to a symbol with STV_PROTECTED \
334                          visibility"
335 @ MSG_RELINVSEC         "relocation warning: %s: file %s: section: [%u]%s: \
336                          against suspicious section [%u]%s; relocation ignored"
337 @ MSG_REL_TLSIE         "relocation warning: %s: file %s: symbol %s: \
338                          relocation has restricted use when building a shared \
339                          object"

341 @ MSG_REL_SLOPCDATNONAM "relocation warning: %s: file %s: section [%u]%s: \
342                          relocation against discarded COMDAT section [%u]%s: \
343                          redirected to file %s"
344 @ MSG_REL_SLOPCDATNAM   "relocation warning: %s: file %s: section [%u]%s: \
345                          symbol %s: relocation against discarded COMDAT \
346                          section [%u]%s: redirected to file %s"
347 @ MSG_REL_SLOPCDATNOSYM "relocation warning: %s: file %s: section [%u]%s: \
348                          symbol %s: relocation against discarded COMDAT \
349                          section [%u]%s: symbol not found, relocation ignored"

351 @ MSG_REL_NOREG         "relocation error: REGISTER relocation not supported \
352                          on target architecture"

354 #
355 # TRANSLATION_NOTE
356 #       The following 7 messages are the message to print the
357 #       following example messages.
358 #
359 #Text relocation remains                         referenced
360 #    against symbol                offset      in file
361 #str                                0x14        main.o
362 #printf                             0x1c        main.o
363 #
364 #       The first two lines are the header, and the next msgid
365 #       is the format string for the header.
366 #       Tabs and spaces are used for alignment.
367 #       The first and third %s are for: "Text relocation remains against symbol"
368 #       The second %s and fourth %s are for: "referenced in file"
369 #       The third %s is for: "offset"
370 #
371 @ MSG_REL_REMAIN_FMT_1  "%-40s\t%s\n    %s\t\t    %s\t%s"
372 #
373 # TRANSLATION_NOTE
374 #       The next two msdid make a sentence. So translate:
375 #               "Text relocation remain against symbol"
376 #       And separate them into two msgstr considering the proper
377 #       alignment.
378 @ MSG_REL_RMN_ITM_11    "Text relocation remains"
379 @ MSG_REL_RMN_ITM_12    "against symbol"
380 @ MSG_REL_RMN_ITM_13    "warning: Text relocation remains"

382 @ MSG_REL_RMN_ITM_2     "offset"

384 #
385 # TRANSLATION_NOTE
386 #       The next two msdid make a sentence. So translate:
387 #               "referenced in file"
388 #       And separate them into two msgstr considering the proper
```

```
389 #       alignment.
390 @ MSG_REL_RMN_ITM_31    "referenced"
391 @ MSG_REL_RMN_ITM_32    "in file"
392 @ MSG_REL_REMAIN_2      "%-35s 0x%-8llx\t%s"
393 @ MSG_REL_REMAIN_3      "relocations remain against allocatable but \
394                          non-writable sections"

396 # Files processing messages

398 @ MSG_FIL_MULINC_1      "file %s: attempted multiple inclusion of file"
399 @ MSG_FIL_MULINC_2      "file %s: linked to %s: attempted multiple inclusion \
400                          of file"
401 @ MSG_FIL_SOINSTAT      "input of shared object '%s' in static mode"
402 @ MSG_FIL_INVALSEC      "file %s: section [%u]%s has invalid type %s"
403 @ MSG_FIL_NOTFOUND      "file %s: required by %s, not found"
404 @ MSG_FIL_MALSTR        "file %s: section [%u]%s: malformed string table, \
405                          initial or final byte"
406 @ MSG_FIL_PTHTOLONG     "'%s/%s' pathname too long"
407 @ MSG_FIL_EXCLUDE       "file %s: section [%u]%s contains both SHF_EXCLUDE and \
408                          SHF_ALLOC flags: SHF_EXCLUDE ignored"
409 @ MSG_FIL_INTERRUPT     "file %s: creation interrupted: %s"
410 @ MSG_FIL_INVRELOC1     "file %s: section [%u]%s: relocations can not be \
411                          applied against section [%u]%s"
412 @ MSG_FIL_INVSHINFO     "file %s: section [%u]%s: has invalid sh_info: %lld"
413 @ MSG_FIL_INVSHLINK     "file %s: section [%u]%s: has invalid sh_link: %lld"
414 @ MSG_FIL_INVSHENTSIZE  "file %s: section [%u]%s: has invalid sh_entsize: %lld"
415 @ MSG_FIL_NOSTRTABLE    "file %s: section [%u]%s: symbol[%d]: specifies string \
416                          table offset 0x%llx: no string table is available"
417 @ MSG_FIL_EXCSTRTABLE   "file %s: section [%u]%s: symbol[%d]: specifies string \
418                          table offset 0x%llx: exceeds string table %s: \
419                          size 0x%llx"
420 @ MSG_FIL_NONAMESYM     "file %s: section [%u]%s: symbol[%d]: global symbol has
421                          no name"
422 @ MSG_FIL_UNKCAP        "file %s: section [%u]%s: unknown capability tag: %d"
423 @ MSG_FIL_BADSF1        "file %s: section [%u]%s: unknown software \
424                          capabilities: 0x%llx; ignored"
425 @ MSG_FIL_INADDR32SF1   "file %s: section [%u]%s: software capability ADDR32: is
426                          ineffective when building 32-bit object; ignored"
427 @ MSG_FIL_EXADDR32SF1   "file %s: section [%u]%s: software capability ADDR32: \
428                          requires executable be built with ADDR32 capability"

430 @ MSG_FIL_BADORDREF     "file %s: section [%u]%s: contains illegal reference \
431                          to discarded section: [%u]%s"

433 # Recording name conflicts

435 @ MSG_REC_OPTCNFLT      "recording name conflict: file '%s' and %s provide \
436                          identical dependency names: %s"
437 @ MSG_REC_OBJCNFLT      "recording name conflict: file '%s' and file '%s' \
438                          provide identical dependency names: %s  %s"
439 @ MSG_REC_CNFLTHINT     "(possible multiple inclusion of the same file)"

441 # System call messages

443 @ MSG_SYS_OPEN          "file %s: open failed: %s"
444 @ MSG_SYS_UNLINK        "file %s: unlink failed: %s"
445 @ MSG_SYS_MMAPANON      "mmap anon failed: %s"
446 @ MSG_SYS_MALLOC        "malloc failed: %s"


449 # Messages related to platform support

451 @ MSG_TARG_UNSUPPORTED  "unsupported ELF machine type: %s"


454 # ELF processing messages
```



```
456 @ MSG_ELF_LIBELF        "libelf: version not supported: %d"

458 @ MSG_ELF_ARMEM         "file %s: unable to locate archive member;\n\t\
459                          offset=%x, symbol=%s"

461 @ MSG_ELF_ARSYM         "file %s ignored: unable to locate archive symbol table"

463 @ MSG_ELF_VERSYM        "file %s: version symbol section entry mismatch:\n\t\
464                          (section [%u]%s entries=%d; section [%u]%s entries=%d)"

466 @ MSG_ELF_NOGROUPSECT   "file %s: section [%u]%s: SHF_GROUP flag set, but no \
467                          corresponding SHT_GROUP section found"

469 # Section processing errors

471 @ MSG_SCN_NONALLOC      "%s: non-allocatable section '%s' directed to a \
472                          loadable segment: %s"

474 @ MSG_SCN_MULTICOMDAT   "file %s: section [%u]%s: cannot be susceptible to multi
475                          COMDAT mechanisms: %s"

477 @ MSG_SCN_DWFOVRFLW     "%s: section %s: encoded DWARF data exceeds \
478                          section size"
479 @ MSG_SCN_DWFBADENC     "%s: section %s: invalid DWARF encoding: %#x"

481 # Symbol processing errors

483 @ MSG_SYM_NOSECDEF      "symbol '%s' in file %s has no section definition"
484 @ MSG_SYM_INVSEC        "symbol '%s' in file %s associated with invalid \
485                          section[%lld]"
486 @ MSG_SYM_TLS           "symbol '%s' in file %s (STT_TLS), is defined \
487                          in a non-SHF_TLS section"
488 @ MSG_SYM_BADADDR       "symbol '%s' in file %s: section [%u]%s: size %#llx: \
489                          symbol (address %#llx, size %#llx) lies outside \
490                          of containing section"
491 @ MSG_SYM_BADADDR_ROTXT "symbol '%s' in file %s: readonly text section \
492                          [%u]%s: size %#llx: symbol (address %#llx, \
493                          size %#llx) lies outside of containing section"
494 @ MSG_SYM_MULDEF        "symbol '%s' is multiply-defined:"
495 @ MSG_SYM_CONFVIS       "symbol '%s' has conflicting visibilities:"
496 @ MSG_SYM_DIFFTYPE      "symbol '%s' has differing types:"
497 @ MSG_SYM_DIFFATTR      "symbol '%s' has differing %s:\n\
498                          \t(file %s value=0x%llx; file %s value=0x%llx);"
499 @ MSG_SYM_FILETYPES     "\t(file %s type=%s; file %s type=%s);"
500 @ MSG_SYM_VISTYPES      "\t(file %s visibility=%s; file %s visibility=%s);"
501 @ MSG_SYM_DEFTAKEN      "\t%s definition taken"
502 @ MSG_SYM_DEFUPDATE     "\t%s definition taken and updated with larger size"
503 @ MSG_SYM_LARGER        "\tlargest value applied"
504 @ MSG_SYM_TENTERR       "\ttentative symbol cannot override defined symbol \
505                          of smaller size"

507 @ MSG_SYM_INVSHNDX      "symbol %s has invalid section index; \
508                          ignored:\n\t(file %s value=%s);"
509 @ MSG_SYM_NONGLOB       "global symbol %s has non-global binding:\n\
510                          \t(file %s value=%s);"
511 @ MSG_SYM_RESERVE       "reserved symbol '%s' already defined in file %s"
512 @ MSG_SYM_NOTNULL       "undefined symbol '%s' with non-zero value encountered \
513                          from file %s"
514 @ MSG_SYM_DUPSORTADDR   "section %s: symbol '%s' and symbol '%s' have the \
515                          same address: %#llx: remove duplicate with \
516                          NOSORTSYM mapfile directive"

518 @ MSG_PSYM_INVMINFO1    "file %s: section [%u]%s: entry[%d] has invalid m_info:
519                          0x%llx for symbol index"
520 @ MSG_PSYM_INVMINFO2    "file %s: section [%u]%s: entry[%d] has invalid m_info:
```

```
521                          0x%llx for size"
522 @ MSG_PSYM_INVMREPEAT   "file %s: section [%u]%s: entry[%d] has invalid m_repeat
523                          0x%llx"
524 @ MSG_PSYM_CANNOTEXPND  "file %s: section [%u]%s: entry[%d] can not be expanded:
525                          associated symbol size is unknown %s"
526 @ MSG_PSYM_NOSTATIC     "and partial initialization cannot be deferred to \
527                          a static object"
528 @ MSG_MOVE_OVERLAP      "file %s: section [%u]%s: symbol '%s' overlapping move \
529                          initialization: start=0x%llx, length=0x%llx: \
530                          start=0x%llx, length=0x%llx"
531 @ MSG_PSYM_EXPREASON1   "output file is static object"
532 @ MSG_PSYM_EXPREASON2   "-z nopartial option in effect"
533 @ MSG_PSYM_EXPREASON3   "move infrastructure size is greater than move data"

535 #
536 # Support library failures
537 #
538 @ MSG_SUP_NOLOAD        "dlopen() of support library (%s) failed with \
539                          error: %s"
540 @ MSG_SUP_BADVERSION    "initialization of support library (%s) failed with \
541                          bad version.  supported: %d returned: %d"


544 #
545 # TRANSLATION_NOTE
546 #       The following 7 messages are the message to print the
547 #       following example messages.
548 #
549 #Undefined                      first referenced
550 # symbol                            in file
551 #inquire                           halt_hold.o
552 #
553 @ MSG_SYM_FMT_UNDEF     "%s\t\t\t%s\
554                          \n %s  \t\t\t    %s"

556 #
557 # TRANSLATION_NOTE
558 #       The next two msdid make a sentence. So translate:
559 #               "Undefined symbol"
560 #       And separate them into two msgstr considering the proper
561 #       alignment.
562 @ MSG_SYM_UNDEF_ITM_11  "Undefined"
563 @ MSG_SYM_UNDEF_ITM_12  "symbol"
564 #
565 # TRANSLATION_NOTE
566 #       The next two msdid make a sentence. So translate:
567 #               "first referenced in file"
568 #       And separate them into two msgstr considering the proper
569 #       alignment.
570 @ MSG_SYM_UNDEF_ITM_21  "first referenced"
571 @ MSG_SYM_UNDEF_ITM_22  "in file"
572 #

574 @ MSG_SYM_UND_UNDEF     "%-35s %s"
575 @ MSG_SYM_UND_NOVER     "%-35s %s  (symbol has no version assigned)"
576 @ MSG_SYM_UND_IMPL      "%-35s %s  (symbol belongs to implicit dependency %s)"
577 @ MSG_SYM_UND_NOTA      "%-35s %s  (symbol belongs to unavailable version %s \
578                          (%s))"
579 @ MSG_SYM_UND_BNDLOCAL  "%-35s %s  (symbol scope specifies local binding)"

581 @ MSG_SYM_ENTRY         "entry point"
582 @ MSG_SYM_UNDEF         "%s symbol '%s' is undefined"
583 @ MSG_SYM_EXTERN        "%s symbol '%s' is undefined  (symbol belongs to \
584                          dependency %s)"
585 @ MSG_SYM_NOCRT         "symbol '%s' not found, but %s section exists - \
586                          possible link-edit without using the compiler driver"
```



```
588 # Output file update messages

590 @ MSG_UPD_NOREADSEG     "No read-only segments found.  Setting '_etext' to 0"
591 @ MSG_UPD_NORDWRSEG     "No read-write segments found.  Setting '_edata' to 0"
592 @ MSG_UPD_NOSEG         "Setting 'end' and '_end' to 0"

594 @ MSG_UPD_SEGOVERLAP    "%s: segment address overlap;\n\
595                          \tprevious segment ending at address 0x%llx overlaps\n\
596                          \tuser defined segment '%s' starting at address 0x%llx"
597 @ MSG_UPD_LARGSIZE      "%s: segment %s calculated size 0x%llx\n\
598                          \tis larger than user-defined size 0x%llx"

600 @ MSG_UPD_NOBITS        "NOBITS section found before end of initialized data"
601 @ MSG_SEG_FIRNOTLOAD    "First segment has type %s, PT_LOAD required: %s"
602 @ MSG_UPD_MULEHFRAME    "file %s; section [%u]%s and file %s; section [%u]%s \
603                          have incompatibile attributes and cannot \
604                          be merged into a single output section"


607 # Version processing messages

609 @ MSG_VER_HIGHER        "file %s: version revision %d is higher than \
610                          expected %d"
611 @ MSG_VER_NOEXIST       "file %s: version '%s' does not exist:\n\
612                          \trequired by file %s"
613 @ MSG_VER_UNDEF         "version '%s' undefined, referenced by version '%s':\n\
614                          \trequired by file %s"
615 @ MSG_VER_UNAVAIL       "file %s: version '%s' is unavailable:\n\
616                          \trequired by file %s"
617 @ MSG_VER_DEFINED       "version symbol '%s' already defined in file %s"
618 @ MSG_VER_INVALNDX      "version symbol '%s' from file %s has an invalid \
619                          version index (%d)"
620 @ MSG_VER_ADDVERS       "unused $ADDVERS specification from file '%s' \
621                          for object '%s'\nversion(s):"
622 @ MSG_VER_ADDVER        "\t%s"
623 @ MSG_VER_CYCLIC        "following versions generate cyclic dependency:"

625 # Capabilities messages

627 @ MSG_CAP_MULDEF        "capabilities symbol '%s' has multiply-defined members:"
628 @ MSG_CAP_MULDEFSYMS    "\t(file %s symbol '%s'; file %s symbol '%s');"
629 @ MSG_CAP_REDUNDANT     "file %s: section [%u]%s: symbol capabilities \
630                          redundant, as object capabilities are more restrictive"
631 @ MSG_CAP_NOSYMSFOUND   "no global symbols have been found that are associated \
632                          with capabilities identified relocatable objects: \
633                          -z symbolcap has no effect"

635 @ MSG_CAPINFO_INVALSYM  "file %s: capabilities info section [%u]%s: index %d: \
636                          family member symbol '%s': invalid"
637 @ MSG_CAPINFO_INVALLEAD "file %s: capabilities info section [%u]%s: index %d: \
638                          family lead symbol '%s': invalid symbol index %d"

640 # Basic strings

642 @ MSG_STR_ALIGNMENTS    "alignments"
643 @ MSG_STR_COMMAND       "(command line)"
644 @ MSG_STR_TLSREL        "(internal TLS relocation requirement)"
645 @ MSG_STR_SIZES         "sizes"
646 @ MSG_STR_UNKNOWN       "<unknown>"
647 @ MSG_STR_SECTION       "%s (section)"
648 @ MSG_STR_SECTION_MSTR  "%s (merged string section)"

650 #
651 # TRANSLATION_NOTE
652 #       The elf_ function name represents a man page reference and should not
```

```
653 #       be translated.
654 @ MSG_ELF_BEGIN         "file %s: elf_begin"
655 @ MSG_ELF_CNTL          "file %s: elf_cntl"
656 @ MSG_ELF_GETARHDR      "file %s: elf_getarhdr"
657 @ MSG_ELF_GETARSYM      "file %s: elf_getarsym"
658 @ MSG_ELF_GETDATA       "file %s: elf_getdata"
659 @ MSG_ELF_GETEHDR       "file %s: elf_getehdr"
660 @ MSG_ELF_GETPHDR       "file %s: elf_getphdr"
661 @ MSG_ELF_GETSCN        "file %s: elf_getscn: scnndx: %d"
662 @ MSG_ELF_GETSHDR       "file %s: elf_getshdr"
663 @ MSG_ELF_MEMORY        "file %s: elf_memory"
664 @ MSG_ELF_NDXSCN        "file %s: elf_ndxscn"
665 @ MSG_ELF_NEWDATA       "file %s: elf_newdata"
666 @ MSG_ELF_NEWEHDR       "file %s: elf_newehdr"
667 @ MSG_ELF_NEWSCN        "file %s: elf_newscn"
668 @ MSG_ELF_NEWPHDR       "file %s: elf_newphdr"
669 @ MSG_ELF_STRPTR        "file %s: elf_strptr"
670 @ MSG_ELF_UPDATE        "file %s: elf_update"
671 @ MSG_ELF_SWAP_WRIMAGE  "file %s: _elf_swap_wrimage"


674 @ MSG_REJ_MACH          "file %s: wrong ELF machine type: %s"
675 @ MSG_REJ_CLASS         "file %s: wrong ELF class: %s"
676 @ MSG_REJ_DATA          "file %s: wrong ELF data format: %s"
677 @ MSG_REJ_TYPE          "file %s: bad ELF type: %s"
678 @ MSG_REJ_BADFLAG       "file %s: bad ELF flags value: %s"
679 @ MSG_REJ_MISFLAG       "file %s: mismatched ELF flags value: %s"
680 @ MSG_REJ_VERSION       "file %s: mismatched ELF/lib version: %s"
681 @ MSG_REJ_HAL           "file %s: HAL R1 extensions required"
682 @ MSG_REJ_US3           "file %s: Sun UltraSPARC III extensions required"
683 @ MSG_REJ_STR           "file %s: %s"
684 @ MSG_REJ_UNKFILE       "file %s: unknown file type"
685 @ MSG_REJ_UNKCAP        "file=%s; unknown capability: %d"
686 @ MSG_REJ_HWCAP_1       "file %s: hardware capability (CA_SUNW_HW_1) \
687                          unsupported: %s"
688 @ MSG_REJ_SFCAP_1       "file %s: software capability (CA_SUNW_SF_1) \
689                          unsupported: %s"
690 @ MSG_REJ_MACHCAP       "file %s: machine capability (CA_SUNW_MACH) \
691                          unsupported: %s"
692 @ MSG_REJ_PLATCAP       "file %s: platform capability (CA_SUNW_PLAT) \
693                          unsupported: %s"
694 @ MSG_REJ_HWCAP_2       "file %s: hardware capability (CA_SUNW_HW_2) \
695                          unsupported: %s"
696 @ MSG_REJ_ARCHIVE       "file %s: invalid archive use"

698 # Guidance messages
699 @ MSG_GUIDE_SUMMARY     "see ld(1) -z guidance for more information"
700 @ MSG_GUIDE_DEFS        "-z defs option recommended for shared objects"
701 @ MSG_GUIDE_DIRECT      "-B direct or -z direct option recommended before \
702                          first dependency"
703 @ MSG_GUIDE_LAZYLOAD    "-z lazyload option recommended before \
704                          first dependency"
705 @ MSG_GUIDE_MAPFILE     "version 2 mapfile syntax recommended: %s"
706 @ MSG_GUIDE_TEXT        "position independent (PIC) code recommended for \
707                          shared objects"
708 @ MSG_GUIDE_UNUSED      "removal of unused dependency recommended: %s"

710 @ _END_


713 # The following strings represent reserved names.  Reference to these strings
714 # is via the MSG_ORIG() macro, and thus translations are not required.

716 @ MSG_STR_EOF           "<eof>"
717 @ MSG_STR_ERROR         "<error>"
718 @ MSG_STR_EMPTY         ""
```



```
719 @ MSG_QSTR_BANG         "'!'"
720 @ MSG_STR_COLON         ":"
721 @ MSG_QSTR_COLON        "':'"
722 @ MSG_QSTR_SEMICOLON    "';'"
723 @ MSG_QSTR_EQUAL        "'='"
724 @ MSG_QSTR_PLUSEQ       "'+='"
725 @ MSG_QSTR_MINUSEQ      "'-='"
726 @ MSG_QSTR_ATSIGN       "'@'"
727 @ MSG_QSTR_DASH         "'-'"
728 @ MSG_QSTR_LEFTBKT      "'{'"
729 @ MSG_QSTR_RIGHTBKT     "'}'"
730 @ MSG_QSTR_PIPE         "'|'"
731 @ MSG_QSTR_STAR         "'*'"
732 @ MSG_STR_DOT           "."
733 @ MSG_STR_SLASH         "/"
734 @ MSG_STR_DYNAMIC       "(.dynamic)"
735 @ MSG_STR_ORIGIN        "$ORIGIN"
736 @ MSG_STR_MACHINE       "$MACHINE"
737 @ MSG_STR_PLATFORM      "$PLATFORM"
738 @ MSG_STR_ISALIST       "$ISALIST"
739 @ MSG_STR_OSNAME        "$OSNAME"
740 @ MSG_STR_OSREL         "$OSREL"
741 @ MSG_STR_UU_REAL_U     "__real_"
742 @ MSG_STR_UU_WRAP_U     "__wrap_"
743 @ MSG_STR_UELF32        "_ELF32"
744 @ MSG_STR_UELF64        "_ELF64"
745 @ MSG_STR_USPARC        "_sparc"
746 @ MSG_STR_UX86          "_x86"
747 @ MSG_STR_TRUE          "true"

749 @ MSG_STR_CDIR_ADD      "$add"
750 @ MSG_STR_CDIR_CLEAR    "$clear"
751 @ MSG_STR_CDIR_ERROR    "$error"
752 @ MSG_STR_CDIR_MFVER    "$mapfile_version"
753 @ MSG_STR_CDIR_IF       "$if"
754 @ MSG_STR_CDIR_ELIF     "$elif"
755 @ MSG_STR_CDIR_ELSE     "$else"
756 @ MSG_STR_CDIR_ENDIF    "$endif"

758 @ MSG_STR_GROUP         "GROUP"
759 @ MSG_STR_SUNW_COMDAT   "SUNW_COMDAT"

761 @ MSG_FMT_ARMEM         "%s(%s)"
762 @ MSG_FMT_COLPATH       "%s:%s"
763 @ MSG_FMT_SYMNAM        "'%s'"
764 @ MSG_FMT_NULLSYMNAM    "%s[%d]"
765 @ MSG_FMT_STRCAT        "%s%s"

767 @ MSG_PTH_RTLD          "/usr/lib/ld.so.1"

769 @ MSG_SUNW_OST_SGS      "SUNW_OST_SGS"


772 # Section strings

774 @ MSG_SCN_BSS           ".bss"
775 @ MSG_SCN_DATA          ".data"
776 @ MSG_SCN_COMMENT       ".comment"
777 @ MSG_SCN_DEBUG         ".debug"
778 @ MSG_SCN_DEBUG_INFO    ".debug_info"
779 @ MSG_SCN_DYNAMIC       ".dynamic"
780 @ MSG_SCN_DYNSYMSORT    ".SUNW_dynsymsort"
781 @ MSG_SCN_DYNTLSSORT    ".SUNW_dyntlssort"
782 @ MSG_SCN_DYNSTR        ".dynstr"
783 @ MSG_SCN_DYNSYM        ".dynsym"
784 @ MSG_SCN_DYNSYM_SHNDX  ".dynsym_shndx"
```

```
785 @ MSG_SCN_LDYNSYM       ".SUNW_ldynsym"
786 @ MSG_SCN_LDYNSYM_SHNDX ".SUNW_ldynsym_shndx"
787 @ MSG_SCN_EX_SHARED     ".ex_shared"
788 @ MSG_SCN_EX_RANGES     ".exception_ranges"
789 @ MSG_SCN_EXCL          ".excl"
790 @ MSG_SCN_FINI          ".fini"
791 @ MSG_SCN_FINIARRAY     ".fini_array"
792 @ MSG_SCN_GOT           ".got"
793 @ MSG_SCN_GNU_LINKONCE  ".gnu.linkonce."
794 @ MSG_SCN_HASH          ".hash"
795 @ MSG_SCN_INDEX         ".index"
796 @ MSG_SCN_INIT          ".init"
797 @ MSG_SCN_INITARRAY     ".init_array"
798 @ MSG_SCN_INTERP        ".interp"
799 @ MSG_SCN_LBSS          ".lbss"
800 @ MSG_SCN_LDATA         ".ldata"
801 @ MSG_SCN_LINE          ".line"
802 @ MSG_SCN_LRODATA       ".lrodata"
803 @ MSG_SCN_PLT           ".plt"
804 @ MSG_SCN_PREINITARRAY  ".preinit_array"
805 @ MSG_SCN_REL           ".rel"
806 @ MSG_SCN_RELA          ".rela"
807 @ MSG_SCN_RODATA        ".rodata"
808 @ MSG_SCN_SBSS          ".sbss"
809 @ MSG_SCN_SBSS2         ".sbss2"
810 @ MSG_SCN_SDATA         ".sdata"
811 @ MSG_SCN_SDATA2        ".sdata2"
812 @ MSG_SCN_SHSTRTAB      ".shstrtab"
813 @ MSG_SCN_STAB          ".stab"
814 @ MSG_SCN_STABEXCL      ".stab.exclstr"
815 @ MSG_SCN_STRTAB        ".strtab"
816 @ MSG_SCN_SUNWMOVE      ".SUNW_move"
817 @ MSG_SCN_SUNWRELOC     ".SUNW_reloc"
818 @ MSG_SCN_SUNWSYMINFO   ".SUNW_syminfo"
819 @ MSG_SCN_SUNWVERSION   ".SUNW_version"
820 @ MSG_SCN_SUNWVERSYM    ".SUNW_versym"
821 @ MSG_SCN_SUNWCAP       ".SUNW_cap"
822 @ MSG_SCN_SUNWCAPINFO   ".SUNW_capinfo"
823 @ MSG_SCN_SUNWCAPCHAIN  ".SUNW_capchain"
824 @ MSG_SCN_SYMTAB        ".symtab"
825 @ MSG_SCN_SYMTAB_SHNDX  ".symtab_shndx"
826 @ MSG_SCN_TBSS          ".tbss"
827 @ MSG_SCN_TDATA         ".tdata"
828 @ MSG_SCN_TEXT          ".text"

830 @ MSG_SYM_FINIARRAY     "finiarray"
831 @ MSG_SYM_INITARRAY     "initarray"
832 @ MSG_SYM_PREINITARRAY  "preinitarray"

834 #
835 # GNU section names
836 #
837 @ MSG_SCN_CTORS         ".ctors"
838 @ MSG_SCN_DTORS         ".dtors"
839 @ MSG_SCN_EHFRAME       ".eh_frame"
840 @ MSG_SCN_EHFRAME_HDR   ".eh_frame_hdr"
841 @ MSG_SCN_GCC_X_TBL     ".gcc_except_table"
842 @ MSG_SCN_JCR           ".jcr"

844 # Segment names for segments referenced by entrance criteria

846 @ MSG_ENT_BSS           "bss"
847 @ MSG_ENT_DATA          "data"
848 @ MSG_ENT_EXTRA         "extra"
849 @ MSG_ENT_LDATA         "ldata"
850 @ MSG_ENT_LRODATA       "lrodata"
```



```
851 @ MSG_ENT_NOTE          "note"
852 @ MSG_ENT_TEXT          "text"

854 # Symbol names

856 @ MSG_SYM_START         "_start"
857 @ MSG_SYM_MAIN          "main"

859 @ MSG_SYM_FINI_U        "_fini"
860 @ MSG_SYM_INIT_U        "_init"
861 @ MSG_SYM_DYNAMIC       "DYNAMIC"
862 @ MSG_SYM_DYNAMIC_U     "_DYNAMIC"
863 @ MSG_SYM_EDATA         "edata"
864 @ MSG_SYM_EDATA_U       "_edata"
865 @ MSG_SYM_END           "end"
866 @ MSG_SYM_END_U         "_end"
867 @ MSG_SYM_ETEXT         "etext"
868 @ MSG_SYM_ETEXT_U       "_etext"
869 @ MSG_SYM_GOFTBL        "GLOBAL_OFFSET_TABLE_"
870 @ MSG_SYM_GOFTBL_U      "_GLOBAL_OFFSET_TABLE_"
871 @ MSG_SYM_PLKTBL        "PROCEDURE_LINKAGE_TABLE_"
872 @ MSG_SYM_PLKTBL_U      "_PROCEDURE_LINKAGE_TABLE_"
873 @ MSG_SYM_TLSGETADDR_U  "__tls_get_addr"
874 @ MSG_SYM_TLSGETADDR_UU "___tls_get_addr"

876 @ MSG_SYM_L_END         "END_"
877 @ MSG_SYM_L_END_U       "_END_"
878 @ MSG_SYM_L_START       "START_"
879 @ MSG_SYM_L_START_U     "_START_"

881 # Support functions

883 @ MSG_SUP_VERSION       "ld_version"
884 @ MSG_SUP_INPUT_DONE    "ld_input_done"

886 @ MSG_SUP_START_64      "ld_start64"
887 @ MSG_SUP_ATEXIT_64     "ld_atexit64"
888 @ MSG_SUP_OPEN_64       "ld_open64"
889 @ MSG_SUP_FILE_64       "ld_file64"
890 @ MSG_SUP_INSEC_64      "ld_input_section64"
891 @ MSG_SUP_SEC_64        "ld_section64"

893 @ MSG_SUP_START         "ld_start"
894 @ MSG_SUP_ATEXIT        "ld_atexit"
895 @ MSG_SUP_OPEN          "ld_open"
896 @ MSG_SUP_FILE          "ld_file"
897 @ MSG_SUP_INSEC         "ld_input_section"
898 @ MSG_SUP_SEC           "ld_section"

900 #
901 # Message previously in 'ld'
902 #
903 #
904 @ _START_

906 # System error messages

908 @ MSG_SYS_STAT          "file %s: stat failed: %s"
909 @ MSG_SYS_READ          "file %s: read failed: %s"
910 @ MSG_SYS_NOTREG        "file %s: is not a regular file"

912 # Argument processing messages

914 @ MSG_ARG_DY_INCOMP     "%s option is incompatible with building a dynamic \
915                          executable"
916 @ MSG_MARG_DY_INCOMP    "%s is incompatible with building a dynamic \
```

```
917                          executable"
918 @ MSG_ARG_ST_INCOMP     "%s option is incompatible with building a static \
919                          object (-dn, -r, --relocatable)"
920 @ MSG_MARG_ST_INCOMP    "%s is incompatible with building a static \
921                          object (-dn, -r, --relocatable)"
922 @ MSG_MARG_ST_ONLYAVL   "%s is only available when building a shared object"
923 @ MSG_ARG_INCOMP        "option %s and %s are incompatible"
924 @ MSG_MARG_INCOMP       "%s and %s are incompatible"
925 @ MSG_ARG_MTONCE        "option %s appears more than once, first setting taken"
926 @ MSG_MARG_MTONCE       "%s appears more than once, first setting taken"
927 @ MSG_ARG_ILLEGAL       "option %s has illegal argument '%s'"
928 @ MSG_ARG_YP            "option -YP and -Y%c may not be specified concurrently"
929 @ MSG_ARG_STRIP         "%s specified with %s; only debugging \
930                          information stripped"
931 @ MSG_ARG_NOFILES       "no files on input command line"
932 @ MSG_ARG_NOFLTR        "option %s is only meaningful when building a filter"
933 @ MSG_ARG_NODEFLIB      "the default library search path has been suppressed, \
934                          but no runpaths have been specified via %s"
935 @ MSG_ARG_NOENTRY       "entry point symbol '%s' is undefined"
936 @ MSG_ARG_UNSUPPORTED   "option %s is no longer supported; ignored"
937 @ MSG_MARG_ONLY         "option %s can only be used with a %s"
938 @ MSG_ARG_UNKNOWN       "unrecognized option '-%c'"
939 @ MSG_ARG_LONG_UNKNOWN  "unrecognized option '%s'"
940 @ MSG_ARG_USEHELP       "use the -z help option for usage information"


943 @ MSG_ARG_FLAGS         "flags processing errors"
944 @ MSG_ARG_FILES         "file processing errors. No output written to %s"
945 @ MSG_ARG_SYM_WARN      "symbol referencing errors"
946 @ MSG_ARG_SYM_FATAL     "symbol referencing errors. No output written to %s"
947 @ MSG_ARG_AR_GRP_OLAP   "%s cannot be nested"
948 @ MSG_ARG_AR_GRP_BAD    "%s used without corresponding %s"


951 # Messages used to refer to options where there is more than
952 # one name accepted.

954 @ MSG_MARG_AR_GRPS      "archive rescan groups \
955                          (-z rescan-start, -(, --start-group)"
956 @ MSG_MARG_AR_GRP_END   "archive rescan group end option \
957                          (-z rescan-end, -), --end-group)"
958 @ MSG_MARG_AR_GRP_START "archive rescan group start option \
959                          (-z rescan-start, -(, --start-group)"
960 @ MSG_MARG_ENTRY        "entry point option (-e, --entry)"
961 @ MSG_MARG_FILTER_AUX   "auxiliary filter option (-f, --auxiliary)"
962 @ MSG_MARG_FILTER       "filter option (-F, --filter)"
963 @ MSG_MARG_OUTFILE      "output object option (-o, --output)"
964 @ MSG_MARG_REL          "relocatable object option (-r, --relocatable)"
965 @ MSG_MARG_RPATH        "runpath option (-R, -rpath)"
966 @ MSG_MARG_SO           "shared object option (-G, -shared)"
967 @ MSG_MARG_SONAME       "soname option (-h, --soname)"
968 @ MSG_MARG_STRIP        "strip option (-s, --strip-all)"

970 # Entrance criteria messages

972 @ MSG_ENT_MAP_FMT_TIL_1 "\t\t%s\n\n"
973 @ MSG_ENT_MAP_TITLE_1   "LINK EDITOR MEMORY MAP"

975 #
976 # TRANSLATION_NOTE -- Entry map header
977 #
978 # The next message is a format string for a title. The title is composed of
979 # two lines. In C locale, it would look like:
980 #
981 #       output          input           new
982 #       section         section         displacement    size
```



```
983 #
984 # The \t characters are used for alignment.  (output section), (input section),
985 # and (new displacement) have to be aligned.
986 #
987 @ MSG_ENT_MAP_FMT_TIL_2 "\n%s\t\t%s\t\t%s\n%s\t\t%s\t\t%s\t%s\n\n"
988 @ MSG_ENT_MAP_FMT_TIL_3 "\n%s\t\t%s\t\t%s\n%s\t\t%s\t\t%s\t\t%s\n\n"
989 @ MSG_ENT_ITM_OUTPUT    "output"
990 @ MSG_ENT_ITM_INPUT     "input"
991 @ MSG_ENT_ITM_NEW       "new"
992 @ MSG_ENT_ITM_SECTION   "section"
993 @ MSG_ENT_ITM_DISPMNT   "displacement"
994 @ MSG_ENT_ITM_SIZE      "size"
995 @ MSG_ENT_ITM_VIRTUAL   "virtual"
996 @ MSG_ENT_ITM_ADDRESS   "address"

998 @ MSG_ENT_MAP_ENTRY_1   "%-8.8s\t\t\t%08.2llx\t%08.2llx\n"
999 @ MSG_ENT_MAP_ENTRY_2   "\t\t%-8.8s\t%08.2llx\t%08.2llx %s\n"

1001 #
1002 # TRANSLATION_NOTE -- multiple defined symbol table header
1003 #
1004 # In C locale, an example output is:
1005 #
1006 #               MULTIPLY DEFINED SYMBOLS
1007 #
1008 #
1009 #symbol                          definition used     also defined in
1010 #
1011 #variable1                          main.o
1012 #                                                      ./libfred.so
1013 @ MSG_ENT_MUL_FMT_TIL_0 "\n\n\t\t%s\n\n\n"
1014 @ MSG_ENT_MUL_TIL_0     "MULTIPLY DEFINED SYMBOLS"

1016 #
1017 # TRANSLATION_NOTE -- This is the format string for:
1018 #
1019 #symbol                          definition used     also defined in
1020 #
1021 @ MSG_ENT_MUL_FMT_TIL_1 "%s\t\t\t\t  %s     %s\n\n"
1022 @ MSG_ENT_MUL_ITM_SYM   "symbol"
1023 @ MSG_ENT_MUL_ITM_DEF_0 "definition used"
1024 @ MSG_ENT_MUL_ITM_DEF_1 "also defined in"

1026 #
1027 # TRANSLATION_NOTE -- This is the format string for the second item:
1028 #
1029 @ MSG_ENT_MUL_ENTRY_1   "%-35s %s\n"

1031 #
1032 # TRANSLATION_NOTE -- This is the format string for the third item:
1033 #
1034 @ MSG_ENT_MUL_ENTRY_2   "\t\t\t\t\t\t\t%s\n"

1036 @ MSG_ENT_NOSEC_1       "mapfile: %s segment: section '%s' does not appear \
1037                          in mapfile specified input file(s)"
1038 @ MSG_ENT_NOSEC_2       "mapfile: %s segment: section '%s' does not appear \
1039                          in any input file"

1041 # Library messages

1043 @ MSG_LIB_NOTFOUND      "library -l%s: not found"
1044 @ MSG_LIB_MALFORM       "LD_LIBRARY_PATH malformed"
1045 @ MSG_LIB_BADYP         "-YP library path malformed"


1048 # Mapfile processing messages
```

```
1050 @ MSG_MAP_BADAUTORED    "%s: %llu: auto-reduction ('*') can only be used in \
1051                          hidden/local, or eliminate scope"
1052 @ MSG_MAP_BADFLAG       "%s: %llu: badly formed section flags '%s'"
1053 @ MSG_MAP_BADBNAME      "%s: %llu: basename cannot contain path \
1054                          separator ('/'): %s"
1055 @ MSG_MAP_BADONAME      "%s: %llu: object name cannot contain path \
1056                          separator ('/'): %s"
1057 @ MSG_MAP_REDEFATT      "%s: %llu: redefining %s attribute for '%s'"
1058 @ MSG_MAP_PREMEOF       "%s: %llu: premature EOF"
1059 @ MSG_MAP_ILLCHAR       "%s: %llu: illegal character '\\%03o'"
1060 @ MSG_MAP_MALFORM       "%s: %llu: malformed entry"
1061 @ MSG_MAP_NONLOAD       "%s: %llu: %s not allowed on non-LOAD segments"
1062 @ MSG_MAP_NOSTACK1      "%s: %llu: %s not allowed on STACK segment"
1063 @ MSG_MAP_MOREONCE      "%s: %llu: %s set more than once on same line"
1064 @ MSG_MAP_NOTERM        "%s: %llu: unterminated quoted string: %s"
1065 @ MSG_MAP_SECINSEG      "%s: %llu: section within segment ordering done on \
1066                          a non-existent segment '%s'"
1067 @ MSG_MAP_UNEXINHERIT   "%s: %llu: unnamed version cannot inherit from other \
1068                          versions: %s"
1069 @ MSG_MAP_UNEXTOK       "%s: %llu: unexpected occurrence of '%c' token"

1071 @ MSG_MAP_SEGEMPLOAD    "%s: %llu: empty segment must be of type LOAD or NULL"
1072 @ MSG_MAP_SEGEMPEXE     "%s: %llu: a LOAD empty segment definition is only \
1073                          allowed when creating a dynamic executable"
1074 @ MSG_MAP_SEGEMPATT     "%s: %llu: a LOAD empty segment must have an address \
1075                          and size"
1076 @ MSG_MAP_SEGEMPNOATT   "%s: %llu: a NULL empty segment must not have an \
1077                          address or size"
1078 @ MSG_MAP_SEGEMPSEC     "%s: %llu: empty segment can not have sections \
1079                          assigned to it"
1080 @ MSG_MAP_SEGEMNOPERM   "%s: %llu: empty segment must not have \
1081                          p_flags set: 0x%x"

1083 @ MSG_MAP_CNTADDRORDER  "%s: %llu: segment cannot have an explicit address \
1084                          and also be in the SEGMENT_ORDER list: %s"
1085 @ MSG_MAP_CNTDISSEG     "%s: %llu: segment cannot be disabled: %s"
1086 @ MSG_MAP_DUPNAMENT     "%s: %llu: cannot redefine entrance criteria: %s"
1087 @ MSG_MAP_DUPORDSEG     "%s: %llu: segment is already in %s list: %s"
1088 @ MSG_MAP_DUP_OS_ORD    "%s: %llu: section is already in OS_ORDER list: %s"
1089 @ MSG_MAP_DUP_IS_ORD    "%s: %llu: entrance criteria is already in \
1090                          IS_ORDER list: %s"
1091 @ MSG_MAP_UNKENT        "%s: %llu: unknown entrance criteria \
1092                          (ASSIGN_SECTION): %s"
1093 @ MSG_MAP_UNKSEG        "%s: %llu: unknown segment: %s"
1094 @ MSG_MAP_UNKSYMDEF     "%s: %llu: unknown symbol definition: %s"
1095 @ MSG_MAP_UNKSEGTYP     "%s: %llu: unknown internal segment type %d"
1096 @ MSG_MAP_UNKSOTYP      "%s: %llu: unknown shared object type: %s"
1097 @ MSG_MAP_UNKSEGATT     "%s: %llu: unknown segment attribute: %s"
1098 @ MSG_MAP_UNKSEGFLG     "%s: %llu: unknown segment flag: ?%c"
1099 @ MSG_MAP_UNKSECTYP     "%s: %llu: unknown section type: %s"

1101 @ MSG_MAP_SEGSIZE       "%s: %lld: existing segment size symbols cannot \
1102                          be reset: %s"
1103 @ MSG_MAP_SEGADDR       "%s: %llu: segment address or length '%s' %s"
1104 @ MSG_MAP_BADCAPVAL     "%s: %llu: bad capability value: %s"
1105 @ MSG_MAP_UNKCAPATTR    "%s: %llu: unknown capability attribute '%s'"
1106 @ MSG_MAP_EMPTYCAP      "%s: %llu: empty capability definition; ignored"

1108 @ MSG_MAP_SYMDEF1       "%s: %llu: symbol '%s' is already defined in file: \
1109                          %s: %s"
1110 @ MSG_MAP_SYMDEF2       "%s: %llu: symbol '%s': %s"

1112 @ MSG_MAP_EXPSCOL       "%s: %llu: expected a ';'"
1113 @ MSG_MAP_EXPEQU        "%s: %llu: expected a '=', ':', '|', or '@'"
1114 @ MSG_MAP_EXPSEGATT     "%s: %llu: expected one or more segment attributes \
```



```
1115                          after an '='"
1116 @ MSG_MAP_EXPSEGNAM     "%s: %llu: expected a segment name at the beginning \
1117                          of a line"
1118 @ MSG_MAP_EXPSEGTYPE    "%s: %llu: %s segment cannot be used with %s \
1119                          directive: %s"
1120 @ MSG_MAP_EXPSYM_1      "%s: %llu: expected a symbol name after '@'"
1121 @ MSG_MAP_EXPSYM_2      "%s: %llu: expected a symbol name after '{'"
1122 @ MSG_MAP_EXPSEC        "%s: %llu: expected a section name after '|'"
1123 @ MSG_MAP_EXPSO         "%s: %llu: expected a shared object definition \
1124                          after '-'"
1125 @ MSG_MAP_MULTFILTEE    "%s: %llu: multiple filtee definitions are unsupported"
1126 @ MSG_MAP_NOFILTER      "%s: %llu: filtee definition required"
1127 @ MSG_MAP_BADSF1        "%s: %llu: unknown software capabilities: 0x%llx; \
1128                          ignored"
1129 @ MSG_MAP_INADDR32SF1   "%s: %llu: software capability ADDR32: is ineffective \
1130                          when building 32-bit object: ignored"
1131 @ MSG_MAP_NOINTPOSE     "%s: %llu: interposition symbols can only be defined \
1132                          when building a dynamic executable"
1133 @ MSG_MAP_NOEXVLSZ      "%s: %llu: value and size attributes are incompatible \
1134                          with extern or parent symbols"
1135 @ MSG_MAP_FLTR_ONLYAVL  "%s: %llu: symbol filtering is only available when \
1136                          building a shared object"

1138 @ MSG_MAP_SEGSAME       "segments '%s' and '%s' have the same assigned \
1139                          virtual address"
1140 @ MSG_MAP_EXCLIMIT      "exceeds internal limit"
1141 @ MSG_MAP_NOBADFRM      "number is badly formed"

1143 @ MSG_MAP_SEGTYP        "segment type"
1144 @ MSG_MAP_SEGVADDR      "segment virtual address"
1145 @ MSG_MAP_SEGPHYS       "segment physical address"
1146 @ MSG_MAP_SEGLEN        "segment length"
1147 @ MSG_MAP_SEGFLAG       "segment flags"
1148 @ MSG_MAP_SEGALIGN      "segment alignment"
1149 @ MSG_MAP_SEGROUND      "segment rounding"

1151 @ MSG_MAP_SECTYP        "section type"
1152 @ MSG_MAP_SECFLAG       "section flags"
1153 @ MSG_MAP_SECNAME       "section name"

1155 @ MSG_MAP_SYMVAL        "symbol value"
1156 @ MSG_MAP_SYMSIZE       "symbol size"

1158 @ MSG_MAP_DIFF_SYMVAL   "symbol values differ"
1159 @ MSG_MAP_DIFF_SYMSZ    "symbol sizes differ"
1160 @ MSG_MAP_DIFF_SYMTYP   "symbol types differ"
1161 @ MSG_MAP_DIFF_SYMNDX   "symbol indexes differ"
1162 @ MSG_MAP_DIFF_SYMLCL   "symbol scope conflict against local and non-local"
1163 @ MSG_MAP_DIFF_SYMGLOB  "symbol scope conflict against singleton/exported"
1164 @ MSG_MAP_DIFF_SYMPROT  "symbol scope conflict against protected"
1165 @ MSG_MAP_DIFF_SYMVER   "symbol version conflict"
1166 @ MSG_MAP_DIFF_SYMMUL   "symbol multiple definition"
1167 @ MSG_MAP_DIFF_SNGLDIR  "singleton scope and direct declaration are \
1168                          incompatible"
1169 @ MSG_MAP_DIFF_PROTNDIR "protected scope and no-direct declaration \
1170                          are incompatible"


1173 @ MSG_MAP_SECORDER      "section ordering requested, but no matching section \
1174                          found: segment: %s section: %s"


1177 # Mapfile Directives

1179 @ MSG_MAP_EXP_ATTR      "%s: %llu: expected attribute name (%s), or \
1180                          terminator (';', '}'): %s"
```

```
1181 @ MSG_MAP_EXP_CAPMASK   "%s: %llu: expected capability name, integer value, or \
1182                          terminator (';', '}'): %s"
1183 @ MSG_MAP_EXP_CAPNAME   "%s: %llu: expected name, or terminator (';', '}'): %s"
1184 @ MSG_MAP_EXP_CAPID     "%s: %llu: expected name, or '{' following %s: %s"
1185 @ MSG_MAP_EXP_CAPHW     "%s: %llu: expected hardware capability, or \
1186                          terminator (';', '}'): %s"
1187 @ MSG_MAP_EXP_CAPSF     "%s: %llu: expected software capability, or \
1188                          terminator (';', '}'): %s"
1189 @ MSG_MAP_EXP_EQ        "%s: %llu: expected '=' following %s: %s"
1190 @ MSG_MAP_EXP_EQ_ALL    "%s: %llu: expected '=', '+=', or '-=' following %s: %s"
1191 @ MSG_MAP_EXP_EQ_PEQ    "%s: %llu: expected '=' following %s: %s"
1192 @ MSG_MAP_EXP_DIR       "%s: %llu: expected mapfile directive (%s): %s"
1193 @ MSG_MAP_SFLG_EXBANG   "%s: %llu: '!' appears without corresponding flag"
1194 @ MSG_MAP_EXP_FILNAM    "%s: %llu: expected file name following %s: %s"
1195 @ MSG_MAP_EXP_FILPATH   "%s: %llu: expected file path following %s: %s"
1196 @ MSG_MAP_EXP_INT       "%s: %llu: expected integer value following %s: %s"
1197 @ MSG_MAP_EXP_LBKT      "%s: %llu: expected '{' following %s: %s"
1198 @ MSG_MAP_EXP_OBJNAM    "%s: %llu: expected object name following %s: %s"
1199 @ MSG_MAP_SFLG_ONEBANG  "%s: %llu: '!' can only be specified once per flag"
1200 @ MSG_MAP_EXP_SECFLAG   "%s: %llu: expected section flag (%s), '!', or \
1201                          terminator (';', '}'): %s"
1202 @ MSG_MAP_EXP_SECNAM    "%s: %llu: expected section name following %s: %s"
1203 @ MSG_MAP_EXP_SEGFLAG   "%s: %llu: expected segment flag (%s), or \
1204                          terminator (';', '}'): %s"
1205 @ MSG_MAP_EXP_ECNAM     "%s: %llu: expected entrance criteria (ASSIGN_SECTION) \
1206                          name, or terminator (';', '}'): %s"
1207 @ MSG_MAP_EXP_SEGNAM    "%s: %llu: expected segment name following %s: %s"
1208 @ MSG_MAP_EXP_SEM       "%s: %llu: expected ';' to terminate %s: %s"
1209 @ MSG_MAP_EXP_SEMLBKT   "%s: %llu: expected ';' or '{' following %s: %s"
1210 @ MSG_MAP_EXP_SEMRBKT   "%s: %llu: expected ';' or '}' to terminate %s: %s"
1211 @ MSG_MAP_EXP_SHTYPE    "%s: %llu: expected section type: %s"
1212 @ MSG_MAP_EXP_SYM       "%s: %llu: expected symbol name, symbol scope, \
1213                          or '*': %s"
1214 @ MSG_MAP_EXP_SYMEND    "%s: %llu: expected inherited version name, or \
1215                          terminator (';'): %s"
1216 @ MSG_MAP_EXP_SYMDELIM  "%s: %llu: expected one of ':', ';', or '{': %s"
1217 @ MSG_MAP_EXP_SYMFLAG   "%s: %llu: expected symbol flag (%s), or \
1218                          terminator (';', '}'): %s"
1219 @ MSG_MAP_EXP_SYMNAM    "%s: %llu: expected symbol name following %s: %s"
1220 @ MSG_MAP_EXP_SYMSCOPE  "%s: %llu: expected symbol scope (%s): %s"
1221 @ MSG_MAP_EXP_SYMTYPE   "%s: %llu: expected symbol type (%s): %s"
1222 @ MSG_MAP_EXP_VERSION   "%s: %llu: expected version name following %s: %s"
1223 @ MSG_MAP_BADEXTRA      "%s: %llu: unexpected text found following %s directive"
1224 @ MSG_MAP_VALUELIMIT    "%s: %llu: numeric value exceeds word size: %s"
1225 @ MSG_MAP_MALVALUE      "%s: %llu: malformed numeric value: %s"
1226 @ MSG_MAP_BADVALUETAIL  "%s: %llu: unexpected characters following numeric \
1227                          constant: %s"
1228 @ MSG_MAP_WSNEEDED      "%s: %llu: whitespace needed before token: %s"
1229 @ MSG_MAP_BADCHAR       "%s: %llu: unexpected text: %s"
1230 @ MSG_MAP_BADKWQUOTE    "%s: %llu: mapfile keywords should not be quoted: %s"
1231 @ MSG_MAP_CDIR_NOTBOL   "%s: %llu: mapfile control directive not at start of \
1232                          line: %s"
1233 @ MSG_MAP_NOATTR        "%s: %llu: %s specified no attributes (empty {})"
1234 @ MSG_MAP_NOVALUES      "%s: %llu: %s specified without values"
1235 @ MSG_MAP_INTERR        "<internal error>"
1236 @ MSG_MAP_ISORDVER      "%s: %llu: version 0 mapfile ?O flag and version 1 \
1237                          segment IS_ORDER attribute are mutually exclusive: %s"
1238 @ MSG_MAP_SYMATTR       "symbol attributes";

1240 # Mapfile Control Directives

1242 @ MSG_MAP_CDIR_BADVDIR  "%s: %llu: $mapfile_version directive must specify \
1243                          version 2 or higher: %d"
1244 @ MSG_MAP_CDIR_BADVER   "%s: %llu: unknown mapfile version: %d"
1245 @ MSG_MAP_CDIR_REPVER   "%s: %llu: $mapfile_version must be first directive \
1246                          in file"
```



```
1247 @ MSG_MAP_CDIR_REQARG   "%s: %llu: %s directive requires an argument"
1248 @ MSG_MAP_CDIR_REQNOARG "%s: %llu: %s directive does not accept arguments"
1249 @ MSG_MAP_CDIR_BAD      "%s: %llu: unrecognized mapfile control directive"
1250 @ MSG_MAP_CDIR_NOIF     "%s: %llu: %s directive used without opening $if"
1251 @ MSG_MAP_CDIR_ELSE     "%s: %llu: %s directive preceded by $else on line %d"
1252 @ MSG_MAP_CDIR_NOEND    "%s: %llu: EOF encountered without closing $endif \
1253                          for $if on line %d"
1254 @ MSG_MAP_CDIR_ERROR    "%s: %llu: error: %s"


1257 # Mapfile Conditional Expressions

1259 @ MSG_MAP_CEXP_TOKERR   "%s: %llu: syntax error in conditional expression at: %s
1260 @ MSG_MAP_CEXP_SEMERR   "%s: %llu: malformed conditional expression"
1261 @ MSG_MAP_CEXP_BADOPUSE "%s: %llu: invalid operator use in conditional \
1262                          expression"
1263 @ MSG_MAP_CEXP_UNBALPAR "%s: %llu: unbalanced parenthesis in conditional \
1264                          expression"
1265 @ MSG_MAP_BADCESC       "%s: %llu: unrecognized escape in double quoted \
1266                          token: \\%c\n"

1268 # Generic error diagnostic labels

1270 @ MSG_STR_NULL          "(null)"

1272 @ MSG_DBG_DFLT_FMT      "debug: "
1273 @ MSG_DBG_AOUT_FMT      "debug: a.out: "
1274 @ MSG_DBG_NAME_FMT      "debug: %s: "

1276 # -z assert-deflib strings

1278 @ MSG_ARG_ASSDEFLIB_MALFORMED   "library name malformed: %s"
1279 @ MSG_ARG_ASSDEFLIB_FOUND       "dynamic library found on default search path \
1280                                 (%s): lib%s.so"

1282 @ _END_


1285 # Software identification.  Note, the SGU strings is historic, and has
1286 # little relevance.  It is preserved as applications have used this
1287 # string to identify the Solaris link-editor.

1289 @ MSG_SGS_ID            "ld: Software Generation Utilities - \
1290                          Solaris Link Editors: "

1292 # The following strings represent reserved words, files, pathnames and symbols.
1293 # Reference to this strings is via the MSG_ORIG() macro, and thus no message
1294 # translation is required.

1296 @ MSG_DBG_FOPEN_MODE    "w"

1298 @ MSG_DBG_CLS32_FMT     "32: "
1299 @ MSG_DBG_CLS64_FMT     "64: "

1301 @ MSG_STR_PATHTOK       ";:"
1302 @ MSG_STR_AOUT          "a.out"

1304 @ MSG_STR_LIB_A         "%s/lib%s.a"
1305 @ MSG_STR_LIB_SO        "%s/lib%s.so"
1306 @ MSG_STR_PATH          "%s/%s"
1307 @ MSG_STR_STRNL         "%s\n"
1308 @ MSG_STR_NL            "\n"
1309 @ MSG_STR_CAPGROUPID    "CAP_GROUP_%d"

1311 @ MSG_STR_LD_DYNAMIC    "dynamic"
1312 @ MSG_STR_SYMBOLIC      "symbolic"
```

```
@ MSG_STR_ELIMINATE     "eliminate"
@ MSG_STR_LOCAL         "local"
@ MSG_STR_PROGBITS      "progbits"
@ MSG_STR_SYMTAB        "symtab"
@ MSG_STR_DYNSYM        "dynsym"
@ MSG_STR_REL           "rel"
@ MSG_STR_RELA          "rela"
@ MSG_STR_STRTAB        "strtab"
@ MSG_STR_HASH          "hash"
@ MSG_STR_LIB           "lib"
@ MSG_STR_NOTE          "note"
@ MSG_STR_NOBITS        "nobits"
@ MSG_STR_HWCAP_1       "hwcap_1"
@ MSG_STR_SFCAP_1       "sfcap_1"
@ MSG_STR_SOEXT         ".so"

@ MSG_STR_OPTIONS       "3:6:abc:d:e:f:h:il:mo:p:rstu:z:B:CD:F:GI:L:M:N:P:Q:R:\
                         S:VW:Y:?"

# Argument processing strings

@ MSG_ARG_3             "-3"
@ MSG_ARG_6             "-6"
@ MSG_ARG_A             "-a"
@ MSG_ARG_B             "-b"
@ MSG_ARG_CB            "-B"
@ MSG_ARG_BDIRECT       "-Bdirect"
@ MSG_ARG_BDYNAMIC      "-Bdynamic"
@ MSG_ARG_BELIMINATE    "-Beliminate"
@ MSG_ARG_BGROUP        "-Bgroup"
@ MSG_ARG_BLOCAL        "-Blocal"
@ MSG_ARG_BNODIRECT     "-Bnodirect"
@ MSG_ARG_BSYMBOLIC     "-Bsymbolic"
@ MSG_ARG_BTRANSLATOR   "-Btranslator"
@ MSG_ARG_C             "-c"
@ MSG_ARG_D             "-d"
@ MSG_ARG_DY            "-dy"
@ MSG_ARG_CI            "-I"
@ MSG_ARG_CN            "-N"
@ MSG_ARG_P             "-p"
@ MSG_ARG_CP            "-P"
@ MSG_ARG_CQ            "-Q"
@ MSG_ARG_CY            "-Y"
@ MSG_ARG_CYL           "-YL"
@ MSG_ARG_CYP           "-YP"
@ MSG_ARG_CYU           "-YU"
@ MSG_ARG_Z             "-z"
@ MSG_ARG_ZDEFNODEF     "-z[defs|nodefs]"
@ MSG_ARG_ZASLR         "-zaslr"
#endif /* ! codereview */
@ MSG_ARG_ZGUIDE        "-zguidance"
@ MSG_ARG_ZNODEF        "-znodefs"
@ MSG_ARG_ZNOINTERP     "-znointerp"
@ MSG_ARG_ZRELAXRELOC   "-zrelaxreloc"
@ MSG_ARG_ZNORELAXRELOC "-znorelaxreloc"
@ MSG_ARG_ZTEXT         "-ztext"
@ MSG_ARG_ZTEXTOFF      "-ztextoff"
@ MSG_ARG_ZTEXTWARN     "-ztextwarn"
@ MSG_ARG_ZTEXTALL      "-z[text|textwarn|textoff]"
@ MSG_ARG_ZLOADFLTR     "-zloadfltr"
@ MSG_ARG_ZCOMBRELOC    "-zcombreloc"
@ MSG_ARG_ZSYMBOLCAP    "-zsymbolcap"
@ MSG_ARG_ZFATWNOFATW   "-z[fatal-warnings|nofatalwarnings]"

@ MSG_ARG_ABSEXEC       "absexec"
@ MSG_ARG_ALTEXEC64     "altexec64"
```



```
@ MSG_ARG_ASLR          "aslr"
#endif /* ! codereview */
@ MSG_ARG_NOCOMPSTRTAB  "nocompstrtab"
@ MSG_ARG_GROUPPERM     "groupperm"
@ MSG_ARG_NOGROUPPERM   "nogroupperm"
@ MSG_ARG_LAZYLOAD      "lazyload"
@ MSG_ARG_NOLAZYLOAD    "nolazyload"
@ MSG_ARG_INTERPOSE     "interpose"
@ MSG_ARG_DIRECT        "direct"
@ MSG_ARG_NODIRECT      "nodirect"
@ MSG_ARG_IGNORE        "ignore"
@ MSG_ARG_RECORD        "record"
@ MSG_ARG_INITFIRST     "initfirst"
@ MSG_ARG_INITARRAY     "initarray="
@ MSG_ARG_FINIARRAY     "finiarray="
@ MSG_ARG_PREINITARRAY  "preinitarray="
@ MSG_ARG_RTLDINFO      "rtldinfo="
@ MSG_ARG_DTRACE        "dtrace="
@ MSG_ARG_TRANSLATOR    "translator"
@ MSG_ARG_NOOPEN        "nodlopen"
@ MSG_ARG_NOW           "now"
@ MSG_ARG_ORIGIN        "origin"
@ MSG_ARG_DEFS          "defs"
@ MSG_ARG_NODEFS        "nodefs"
@ MSG_ARG_NODUMP        "nodump"
@ MSG_ARG_NOVERSION     "noversion"
@ MSG_ARG_TEXT          "text"
@ MSG_ARG_TEXTOFF       "textoff"
@ MSG_ARG_TEXTWARN      "textwarn"
@ MSG_ARG_MULDEFS       "muldefs"
@ MSG_ARG_NODELETE      "nodelete"
@ MSG_ARG_NOINTERP      "nointerp"
@ MSG_ARG_NOPARTIAL     "nopartial"
@ MSG_ARG_NORELOC       "noreloc"
@ MSG_ARG_REDLOCSYM     "redlocsym"
@ MSG_ARG_VERBOSE       "verbose"
@ MSG_ARG_WEAKEXT       "weakextract"
@ MSG_ARG_LOADFLTR      "loadfltr"
@ MSG_ARG_ALLEXTRT      "allextract"
@ MSG_ARG_DFLEXTRT      "defaultextract"
@ MSG_ARG_COMBRELOC     "combreloc"
@ MSG_ARG_NOCOMBRELOC   "nocombreloc"
@ MSG_ARG_NODEFAULTLIB  "nodefaultlib"
@ MSG_ARG_ENDFILTEE     "endfiltee"
@ MSG_ARG_LD32          "ld32="
@ MSG_ARG_LD64          "ld64="
@ MSG_ARG_RESCAN        "rescan"
@ MSG_ARG_RESCAN_NOW    "rescan-now"
@ MSG_ARG_RESCAN_START  "rescan-start"
@ MSG_ARG_RESCAN_END    "rescan-end"
@ MSG_ARG_GUIDE         "guidance"
@ MSG_ARG_NOLDYNSYM     "noldynsym"
@ MSG_ARG_RELAXRELOC    "relaxreloc"
@ MSG_ARG_NORELAXRELOC  "norelaxreloc"
@ MSG_ARG_NOSIGHANDLER  "nosighandler"
@ MSG_ARG_GLOBAUDIT     "globalaudit"
@ MSG_ARG_TARGET        "target="
@ MSG_ARG_WRAP          "wrap="
@ MSG_ARG_FATWARN       "fatal-warnings"
@ MSG_ARG_NOFATWARN     "nofatal-warnings"
@ MSG_ARG_HELP          "help"
@ MSG_ARG_GROUP         "group"
@ MSG_ARG_REDUCE        "reduce"
@ MSG_ARG_STATIC        "static"
@ MSG_ARG_SYMBOLCAP     "symbolcap"
@ MSG_ARG_DEFERRED      "deferred"
```

```
@ MSG_ARG_NODEFERRED    "nodeferred"
@ MSG_ARG_ASSDEFLIB     "assert-deflib"

@ MSG_ARG_LCOM          "L,"
@ MSG_ARG_PCOM          "P,"
@ MSG_ARG_UCOM          "U,"

@ MSG_ARG_T_RPATH       "rpath"
@ MSG_ARG_T_SHARED      "shared"
@ MSG_ARG_T_SONAME      "soname"
@ MSG_ARG_T_WL          "l,-"

@ MSG_ARG_T_AUXFLTR     "-auxiliary"
@ MSG_ARG_T_MULDEFS     "-allow-multiple-definition"
@ MSG_ARG_T_INTERP      "-dynamic-linker"
@ MSG_ARG_T_ENDGROUP    "-end-group"
@ MSG_ARG_T_ENTRY       "-entry"
@ MSG_ARG_T_STDFLTR     "-filter"
@ MSG_ARG_T_FATWARN     "-fatal-warnings"
@ MSG_ARG_T_NOFATWARN   "-no-fatal-warnings"
@ MSG_ARG_T_HELP        "-help"
@ MSG_ARG_T_LIBRARY     "-library"
@ MSG_ARG_T_LIBPATH     "-library-path"
@ MSG_ARG_T_NOUNDEF     "-no-undefined"
@ MSG_ARG_T_NOWHOLEARC  "-no-whole-archive"
@ MSG_ARG_T_OUTPUT      "-output"
@ MSG_ARG_T_RELOCATABLE "-relocatable"
@ MSG_ARG_T_STARTGROUP  "-start-group"
@ MSG_ARG_T_STRIP       "-strip-all"
@ MSG_ARG_T_UNDEF       "-undefined"
@ MSG_ARG_T_VERSION     "-version"
@ MSG_ARG_T_WHOLEARC    "-whole-archive"
@ MSG_ARG_T_WRAP        "-wrap"
@ MSG_ARG_T_OPAR        "("
@ MSG_ARG_T_CPAR        ")"

@ MSG_ARG_ENABLED       "enabled"
@ MSG_ARG_DISABLED      "disabled"

#endif /* ! codereview */
# -z guidance=item strings
@ MSG_ARG_GUIDE_DELIM           ",: \t"
@ MSG_ARG_GUIDE_NO_ALL          "noall"
@ MSG_ARG_GUIDE_NO_DEFS         "nodefs"
@ MSG_ARG_GUIDE_NO_DIRECT       "nodirect"
@ MSG_ARG_GUIDE_NO_LAZYLOAD     "nolazyload"
@ MSG_ARG_GUIDE_NO_MAPFILE      "nomapfile"
@ MSG_ARG_GUIDE_NO_TEXT         "notext"
@ MSG_ARG_GUIDE_NO_UNUSED       "nounused"

# Environment variable strings

@ MSG_LD_RUN_PATH       "LD_RUN_PATH"
@ MSG_LD_LIBPATH_32     "LD_LIBRARY_PATH_32"
@ MSG_LD_LIBPATH_64     "LD_LIBRARY_PATH_64"
@ MSG_LD_LIBPATH        "LD_LIBRARY_PATH"

@ MSG_LD_NOVERSION_32   "LD_NOVERSION_32"
@ MSG_LD_NOVERSION_64   "LD_NOVERSION_64"
@ MSG_LD_NOVERSION      "LD_NOVERSION"

@ MSG_SGS_SUPPORT_32    "SGS_SUPPORT_32"
@ MSG_SGS_SUPPORT_64    "SGS_SUPPORT_64"
@ MSG_SGS_SUPPORT       "SGS_SUPPORT"
```



```
# Symbol names

@ MSG_SYM_LIBVER_U      "_lib_version"


# Mapfile tokens

@ MSG_MAP_LOAD          "load"
@ MSG_MAP_NOTE          "note"
@ MSG_MAP_NULL          "null"
@ MSG_MAP_STACK         "stack"
@ MSG_MAP_ADDVERS       "addvers"
@ MSG_MAP_FUNCTION      "function"
@ MSG_MAP_DATA          "data"
@ MSG_MAP_COMMON        "common"
@ MSG_MAP_PARENT        "parent"
@ MSG_MAP_EXTERN        "extern"
@ MSG_MAP_DIRECT        "direct"
@ MSG_MAP_NODIRECT      "nodirect"
@ MSG_MAP_FILTER        "filter"
@ MSG_MAP_AUXILIARY     "auxiliary"
@ MSG_MAP_OVERRIDE      "override"
@ MSG_MAP_INTERPOSE     "interpose"
@ MSG_MAP_DYNSORT       "dynsort"
@ MSG_MAP_NODYNSORT     "nodynsort"

@ MSG_MAPKW_ALIGN               "ALIGN"
@ MSG_MAPKW_ALLOC               "ALLOC"
@ MSG_MAPKW_ALLOW               "ALLOW"
@ MSG_MAPKW_AMD64_LARGE         "AMD64_LARGE"
@ MSG_MAPKW_ASSIGN_SECTION      "ASSIGN_SECTION"
@ MSG_MAPKW_AUX                 "AUXILIARY"
@ MSG_MAPKW_CAPABILITY          "CAPABILITY"
@ MSG_MAPKW_COMMON              "COMMON"
@ MSG_MAPKW_DATA                "DATA"
@ MSG_MAPKW_DEFAULT             "DEFAULT"
@ MSG_MAPKW_DEPEND_VERSIONS     "DEPEND_VERSIONS"
@ MSG_MAPKW_DIRECT              "DIRECT"
@ MSG_MAPKW_DISABLE             "DISABLE"
@ MSG_MAPKW_DYNSORT             "DYNSORT"
@ MSG_MAPKW_ELIMINATE           "ELIMINATE"
@ MSG_MAPKW_EXECUTE             "EXECUTE"
@ MSG_MAPKW_EXPORTED            "EXPORTED"
@ MSG_MAPKW_EXTERN              "EXTERN"
@ MSG_MAPKW_FILTER              "FILTER"
@ MSG_MAPKW_FILE_BASENAME       "FILE_BASENAME"
@ MSG_MAPKW_FILE_PATH           "FILE_PATH"
@ MSG_MAPKW_FILE_OBJNAME        "FILE_OBJNAME"
@ MSG_MAPKW_FUNCTION            "FUNCTION"
@ MSG_MAPKW_FLAGS               "FLAGS"
@ MSG_MAPKW_GLOBAL              "GLOBAL"
@ MSG_MAPKW_INTERPOSE           "INTERPOSE"
@ MSG_MAPKW_HIDDEN              "HIDDEN"
@ MSG_MAPKW_HDR_NOALLOC         "HDR_NOALLOC"
@ MSG_MAPKW_HW                  "HW"
@ MSG_MAPKW_HW_1                "HW_1"
@ MSG_MAPKW_HW_2                "HW_2"
@ MSG_MAPKW_IS_NAME             "IS_NAME"
@ MSG_MAPKW_IS_ORDER            "IS_ORDER"
@ MSG_MAPKW_LOAD_SEGMENT        "LOAD_SEGMENT"
@ MSG_MAPKW_LOCAL               "LOCAL"
@ MSG_MAPKW_MACHINE             "MACHINE"
@ MSG_MAPKW_MAX_SIZE            "MAX_SIZE"
@ MSG_MAPKW_NOHDR               "NOHDR"
@ MSG_MAPKW_NODIRECT            "NODIRECT"
@ MSG_MAPKW_NODYNSORT           "NODYNSORT"
```

```
1577 @ MSG_MAPKW_NOTE_SEGMENT        "NOTE_SEGMENT"
1578 @ MSG_MAPKW_NULL_SEGMENT        "NULL_SEGMENT"
1579 @ MSG_MAPKW_OS_ORDER            "OS_ORDER"
1580 @ MSG_MAPKW_PADDR               "PADDR"
1581 @ MSG_MAPKW_PARENT              "PARENT"
1582 @ MSG_MAPKW_PHDR_ADD_NULL       "PHDR_ADD_NULL"
1583 @ MSG_MAPKW_PLATFORM            "PLATFORM"
1584 @ MSG_MAPKW_PROTECTED           "PROTECTED"
1585 @ MSG_MAPKW_READ                "READ"
1586 @ MSG_MAPKW_ROUND               "ROUND"
1587 @ MSG_MAPKW_REQUIRE             "REQUIRE"
1588 @ MSG_MAPKW_SEGMENT_ORDER       "SEGMENT_ORDER"
1589 @ MSG_MAPKW_SF                  "SF"
1590 @ MSG_MAPKW_SF_1                "SF_1"
1591 @ MSG_MAPKW_SINGLETON           "SINGLETON"
1592 @ MSG_MAPKW_SIZE                "SIZE"
1593 @ MSG_MAPKW_SIZE_SYMBOL         "SIZE_SYMBOL"
1594 @ MSG_MAPKW_STACK               "STACK"
1595 @ MSG_MAPKW_SYMBOL_SCOPE        "SYMBOL_SCOPE"
1596 @ MSG_MAPKW_SYMBOL_VERSION      "SYMBOL_VERSION"
1597 @ MSG_MAPKW_SYMBOLIC            "SYMBOLIC"
1598 @ MSG_MAPKW_TYPE                "TYPE"
1599 @ MSG_MAPKW_VADDR               "VADDR"
1600 @ MSG_MAPKW_VALUE               "VALUE"
1601 @ MSG_MAPKW_WRITE               "WRITE"


1604 @ MSG_STR_DTRACE        "PT_SUNWDTRACE"
```

```
**********************************************************
   96444 Wed May 27 19:49:05 2015
new/usr/src/cmd/sgs/libld/common/sections.c
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
______unchanged_portion_omitted_

 927 /*
 928  * Make the dynamic section.  Calculate the size of any strings referenced
 929  * within this structure, they will be added to the global string table
 930  * (.dynstr).  This routine should be called before make_dynstr().
 931  *
 932  * This routine must be maintained in parallel with update_odynamic()
 933  * in update.c
 934  */
 935 static uintptr_t
 936 make_dynamic(Ofl_desc *ofl)
 937 {
 938         Shdr            *shdr;
 939         Os_desc         *osp;
 940         Elf_Data        *data;
 941         Is_desc         *isec;
 942         size_t          cnt = 0;
 943         Aliste          idx;
 944         Ifl_desc        *ifl;
 945         Sym_desc        *sdp;
 946         size_t          size;
 947         Str_tbl         *strtbl;
 948         ofl_flag_t      flags = ofl->ofl_flags;
 949         int             not_relobj = !(flags & FLG_OF_RELOBJ);
 950         int             unused = 0;

 952         /*
 953          * Select the required string table.
 954          */
 955         if (OFL_IS_STATIC_OBJ(ofl))
 956                 strtbl = ofl->ofl_strtab;
 957         else
 958                 strtbl = ofl->ofl_dynstrtab;

 960         /*
 961          * Only a limited subset of DT_ entries apply to relocatable
 962          * objects. See the comment at the head of update_odynamic() in
 963          * update.c for details.
 964          */
 965         if (new_section(ofl, SHT_DYNAMIC, MSG_ORIG(MSG_SCN_DYNAMIC), 0,
 966             &isec, &shdr, &data) == S_ERROR)
 967                 return (S_ERROR);

 969         /*
 970          * new_section() does not set SHF_ALLOC.  If we're building anything
 971          * besides a relocatable object, then the .dynamic section should
 972          * reside in allocatable memory.
 973          */
 974         if (not_relobj)
 975                 shdr->sh_flags |= SHF_ALLOC;

 977         /*
 978          * new_section() does not set SHF_WRITE.  If we're building an object
 979          * that specifies an interpretor, then a DT_DEBUG entry is created,
 980          * which is initialized to the applications link-map list at runtime.
 981          */
 982         if (ofl->ofl_osinterp)
```



```
 983                 shdr->sh_flags |= SHF_WRITE;

 985         osp = ofl->ofl_osdynamic =
 986             ld_place_section(ofl, isec, NULL, ld_targ.t_id.id_dynamic, NULL);

 988         /*
 989          * Reserve entries for any needed dependencies.
 990          */
 991         for (APLIST_TRAVERSE(ofl->ofl_sos, idx, ifl)) {
 992                 if (!(ifl->ifl_flags & (FLG_IF_NEEDED | FLG_IF_NEEDSTR)))
 993                         continue;

 995                 /*
 996                  * If this dependency didn't satisfy any symbol references,
 997                  * generate a debugging diagnostic (ld(1) -Dunused can be used
 998                  * to display these).  If this is a standard needed dependency,
 999                  * and -z ignore is in effect, drop the dependency.  Explicitly
1000                  * defined dependencies (i.e., -N dep) don't get dropped, and
1001                  * are flagged as being required to simplify update_odynamic()
1002                  * processing.
1003                  */
1004                 if ((ifl->ifl_flags & FLG_IF_NEEDSTR) ||
1005                     ((ifl->ifl_flags & FLG_IF_DEPREQD) == 0)) {
1006                         if (unused++ == 0)
1007                                 DBG_CALL(Dbg_util_nl(ofl->ofl_lml, DBG_NL_STD));
1008                         DBG_CALL(Dbg_unused_file(ofl->ofl_lml, ifl->ifl_soname,
1009                             (ifl->ifl_flags & FLG_IF_NEEDSTR), 0));

1011                         /*
1012                          * Guidance: Remove unused dependency.
1013                          *
1014                          * If -z ignore is in effect, this warning is not
1015                          * needed because we will quietly remove the unused
1016                          * dependency.
1017                          */
1018                         if (OFL_GUIDANCE(ofl, FLG_OFG_NO_UNUSED) &&
1019                             ((ifl->ifl_flags & FLG_IF_IGNORE) == 0))
1020                                 ld_eprintf(ofl, ERR_GUIDANCE,
1021                                     MSG_INTL(MSG_GUIDE_UNUSED),
1022                                     ifl->ifl_soname);

1024                         if (ifl->ifl_flags & FLG_IF_NEEDSTR)
1025                                 ifl->ifl_flags |= FLG_IF_DEPREQD;
1026                         else if (ifl->ifl_flags & FLG_IF_IGNORE)
1027                                 continue;
1028                 }

1030                 /*
1031                  * If this object requires a DT_POSFLAG_1 entry, reserve it.
1032                  */
1033                 if ((ifl->ifl_flags & MSK_IF_POSFLAG1) && not_relobj)
1034                         cnt++;

1036                 if (st_insert(strtbl, ifl->ifl_soname) == -1)
1037                         return (S_ERROR);
1038                 cnt++;

1040                 /*
1041                  * If the needed entry contains the $ORIGIN token make sure
1042                  * the associated DT_1_FLAGS entry is created.
1043                  */
1044                 if (strstr(ifl->ifl_soname, MSG_ORIG(MSG_STR_ORIGIN))) {
1045                         ofl->ofl_dtflags_1 |= DF_1_ORIGIN;
1046                         ofl->ofl_dtflags |= DF_ORIGIN;
1047                 }
1048         }
```

```
1050         if (unused)
1051                 DBG_CALL(Dbg_util_nl(ofl->ofl_lml, DBG_NL_STD));
1052
1053         if (not_relobj) {
1054                 /*
1055                  * Reserve entries for any per-symbol auxiliary/filter strings.
1056                  */
1057                 cnt += alist_nitems(ofl->ofl_dtsfltrs);
1058
1059                 /*
1060                  * Reserve entries for _init() and _fini() section addresses.
1061                  */
1062                 if (((sdp = ld_sym_find(MSG_ORIG(MSG_SYM_INIT_U),
1063                     SYM_NOHASH, NULL, ofl)) != NULL) &&
1064                     (sdp->sd_ref == REF_REL_NEED) &&
1065                     (sdp->sd_sym->st_shndx != SHN_UNDEF)) {
1066                         sdp->sd_flags |= FLG_SY_UPREQD;
1067                         cnt++;
1068                 }
1069                 if (((sdp = ld_sym_find(MSG_ORIG(MSG_SYM_FINI_U),
1070                     SYM_NOHASH, NULL, ofl)) != NULL) &&
1071                     (sdp->sd_ref == REF_REL_NEED) &&
1072                     (sdp->sd_sym->st_shndx != SHN_UNDEF)) {
1073                         sdp->sd_flags |= FLG_SY_UPREQD;
1074                         cnt++;
1075                 }
1076
1077                 /*
1078                  * Reserve entries for any soname, filter name (shared libs
1079                  * only), run-path pointers, cache names and audit requirements.
1080                  */
1081                 if (ofl->ofl_soname) {
1082                         cnt++;
1083                         if (st_insert(strtbl, ofl->ofl_soname) == -1)
1084                                 return (S_ERROR);
1085                 }
1086                 if (ofl->ofl_filtees) {
1087                         cnt++;
1088                         if (st_insert(strtbl, ofl->ofl_filtees) == -1)
1089                                 return (S_ERROR);
1090
1091                         /*
1092                          * If the filtees entry contains the $ORIGIN token
1093                          * make sure the associated DT_1_FLAGS entry is created.
1094                          */
1095                         if (strstr(ofl->ofl_filtees,
1096                             MSG_ORIG(MSG_STR_ORIGIN))) {
1097                                 ofl->ofl_dtflags_1 |= DF_1_ORIGIN;
1098                                 ofl->ofl_dtflags |= DF_ORIGIN;
1099                         }
1100                 }
1101         }
1102
1103         if (ofl->ofl_rpath) {
1104                 cnt += 2;       /* DT_RPATH & DT_RUNPATH */
1105                 if (st_insert(strtbl, ofl->ofl_rpath) == -1)
1106                         return (S_ERROR);
1107
1108                 /*
1109                  * If the rpath entry contains the $ORIGIN token make sure
1110                  * the associated DT_1_FLAGS entry is created.
1111                  */
1112                 if (strstr(ofl->ofl_rpath, MSG_ORIG(MSG_STR_ORIGIN))) {
1113                         ofl->ofl_dtflags_1 |= DF_1_ORIGIN;
1114                         ofl->ofl_dtflags |= DF_ORIGIN;
```



```
1115                 }
1116         }
1117
1118         if (not_relobj) {
1119                 Aliste  idx;
1120                 Sg_desc *sgp;
1121
1122                 if (ofl->ofl_config) {
1123                         cnt++;
1124                         if (st_insert(strtbl, ofl->ofl_config) == -1)
1125                                 return (S_ERROR);
1126
1127                         /*
1128                          * If the config entry contains the $ORIGIN token
1129                          * make sure the associated DT_1_FLAGS entry is created.
1130                          */
1131                         if (strstr(ofl->ofl_config, MSG_ORIG(MSG_STR_ORIGIN))) {
1132                                 ofl->ofl_dtflags_1 |= DF_1_ORIGIN;
1133                                 ofl->ofl_dtflags |= DF_ORIGIN;
1134                         }
1135                 }
1136                 if (ofl->ofl_depaudit) {
1137                         cnt++;
1138                         if (st_insert(strtbl, ofl->ofl_depaudit) == -1)
1139                                 return (S_ERROR);
1140                 }
1141                 if (ofl->ofl_audit) {
1142                         cnt++;
1143                         if (st_insert(strtbl, ofl->ofl_audit) == -1)
1144                                 return (S_ERROR);
1145                 }
1146
1147                 /*
1148                  * Reserve entries for the DT_HASH, DT_STRTAB, DT_STRSZ,
1149                  * DT_SYMTAB, DT_SYMENT, and DT_CHECKSUM.
1150                  */
1151                 cnt += 6;
1152
1153                 /*
1154                  * If we are including local functions at the head of
1155                  * the dynsym, then also reserve entries for DT_SUNW_SYMTAB
1156                  * and DT_SUNW_SYMSZ.
1157                  */
1158                 if (OFL_ALLOW_LDYNSYM(ofl))
1159                         cnt += 2;
1160
1161                 if ((ofl->ofl_dynsymsortcnt > 0) ||
1162                     (ofl->ofl_dyntlssortcnt > 0))
1163                         cnt++;          /* DT_SUNW_SORTENT */
1164
1165                 if (ofl->ofl_dynsymsortcnt > 0)
1166                         cnt += 2;       /* DT_SUNW_[SYMSORT|SYMSORTSZ] */
1167
1168                 if (ofl->ofl_dyntlssortcnt > 0)
1169                         cnt += 2;       /* DT_SUNW_[TLSSORT|TLSSORTSZ] */
1170
1171                 if ((flags & (FLG_OF_VERDEF | FLG_OF_NOVERSEC)) ==
1172                     FLG_OF_VERDEF)
1173                         cnt += 2;               /* DT_VERDEF & DT_VERDEFNUM */
1174
1175                 if ((flags & (FLG_OF_VERNEED | FLG_OF_NOVERSEC)) ==
1176                     FLG_OF_VERNEED)
1177                         cnt += 2;               /* DT_VERNEED & DT_VERNEEDNUM */
1178
1179                 if ((flags & FLG_OF_COMREL) && ofl->ofl_relocrelcnt)
1180                         cnt++;                  /* DT_RELACOUNT */
```

```
1182                if (flags & FLG_OF_TEXTREL)     /* DT_TEXTREL */
1183                        cnt++;
1184
1185                if (ofl->ofl_osfiniarray)       /* DT_FINI_ARRAY */
1186                        cnt += 2;               /*    DT_FINI_ARRAYSZ */
1187
1188                if (ofl->ofl_osinitarray)       /* DT_INIT_ARRAY */
1189                        cnt += 2;               /*    DT_INIT_ARRAYSZ */
1190
1191                if (ofl->ofl_ospreinitarray)    /* DT_PREINIT_ARRAY & */
1192                        cnt += 2;               /*    DT_PREINIT_ARRAYSZ */
1193
1194                /*
1195                 * If we have plt's reserve a DT_PLTRELSZ, DT_PLTREL and
1196                 * DT_JMPREL.
1197                 */
1198                if (ofl->ofl_pltcnt)
1199                        cnt += 3;
1200
1201                /*
1202                 * If plt padding is needed (Sparcv9).
1203                 */
1204                if (ofl->ofl_pltpad)
1205                        cnt += 2;               /* DT_PLTPAD & DT_PLTPADSZ */
1206
1207                /*
1208                 * If we have any relocations reserve a DT_REL, DT_RELSZ and
1209                 * DT_RELENT entry.
1210                 */
1211                if (ofl->ofl_relocsz)
1212                        cnt += 3;
1213
1214                /*
1215                 * If a syminfo section is required create DT_SYMINFO,
1216                 * DT_SYMINSZ, and DT_SYMINENT entries.
1217                 */
1218                if (flags & FLG_OF_SYMINFO)
1219                        cnt += 3;
1220
1221                /*
1222                 * If there are any partially initialized sections allocate
1223                 * DT_MOVETAB, DT_MOVESZ and DT_MOVEENT.
1224                 */
1225                if (ofl->ofl_osmove)
1226                        cnt += 3;
1227
1228                /*
1229                 * Allocate one DT_REGISTER entry for every register symbol.
1230                 */
1231                cnt += ofl->ofl_regsymcnt;
1232
1233                /*
1234                 * Reserve a entry for each '-zrtldinfo=...' specified
1235                 * on the command line.
1236                 */
1237                for (APLIST_TRAVERSE(ofl->ofl_rtldinfo, idx, sdp))
1238                        cnt++;
1239
1240                /*
1241                 * The following entry should only be placed in a segment that
1242                 * is writable.
1243                 */
1244                if (((sgp = osp->os_sgdesc) != NULL) &&
1245                    (sgp->sg_phdr.p_flags & PF_W) && ofl->ofl_osinterp)
1246                        cnt++;          /* DT_DEBUG */
```



```
1248                /*
1249                 * Capabilities require a .dynamic entry for the .SUNW_cap
1250                 * section.
1251                 */
1252                if (ofl->ofl_oscap)
1253                        cnt++;                  /* DT_SUNW_CAP */
1254
1255                /*
1256                 * Symbol capabilities require a .dynamic entry for the
1257                 * .SUNW_capinfo section.
1258                 */
1259                if (ofl->ofl_oscapinfo)
1260                        cnt++;                  /* DT_SUNW_CAPINFO */
1261
1262                /*
1263                 * Capabilities chain information requires a .SUNW_capchain
1264                 * entry (DT_SUNW_CAPCHAIN), entry size (DT_SUNW_CAPCHAINENT),
1265                 * and total size (DT_SUNW_CAPCHAINSZ).
1266                 */
1267                if (ofl->ofl_oscapchain)
1268                        cnt += 3;
1269
1270                if (flags & FLG_OF_SYMBOLIC)
1271                        cnt++;                  /* DT_SYMBOLIC */
1272
1273                if (ofl->ofl_aslr != 0)         /* DT_SUNW_ASLR */
1274                        cnt++;
1275 #endif /* ! codereview */
1276        }
1277
1278        /*
1279         * Account for Architecture dependent .dynamic entries, and defaults.
1280         */
1281        (*ld_targ.t_mr.mr_mach_make_dynamic)(ofl, &cnt);
1282
1283        /*
1284         * DT_FLAGS, DT_FLAGS_1, DT_SUNW_STRPAD, and DT_NULL. Also,
1285         * allow room for the unused extra DT_NULLs. These are included
1286         * to allow an ELF editor room to add items later.
1287         */
1288        cnt += 4 + DYNAMIC_EXTRA_ELTS;
1289
1290        /*
1291         * DT_SUNW_LDMACH. Used to hold the ELF machine code of the
1292         * linker that produced the output object. This information
1293         * allows us to determine whether a given object was linked
1294         * natively, or by a linker running on a different type of
1295         * system. This information can be valuable if one suspects
1296         * that a problem might be due to alignment or byte order issues.
1297         */
1298        cnt++;
1299
1300        /*
1301         * Determine the size of the section from the number of entries.
1302         */
1303        size = cnt * (size_t)shdr->sh_entsize;
1304
1305        shdr->sh_size = (Xword)size;
1306        data->d_size = size;
1307
1308        /*
1309         * There are several tags that are specific to the Solaris osabi
1310         * range which we unconditionally put into any dynamic section
1311         * we create (e.g. DT_SUNW_STRPAD or DT_SUNW_LDMACH). As such,
1312         * any Solaris object with a dynamic section should be tagged as
```

```
1313          * ELFOSABI_SOLARIS.
1314          */
1315         ofl->ofl_flags |= FLG_OF_OSABI;

1317         return ((uintptr_t)ofl->ofl_osdynamic);
1318 }

1320 /*
1321  * Build the GOT section and its associated relocation entries.
1322  */
1323 uintptr_t
1324 ld_make_got(Ofl_desc *ofl)
1325 {
1326         Elf_Data        *data;
1327         Shdr    *shdr;
1328         Is_desc *isec;
1329         size_t  size = (size_t)ofl->ofl_gotcnt * ld_targ.t_m.m_got_entsize;
1330         size_t  rsize = (size_t)ofl->ofl_relocgotsz;

1332         if (new_section(ofl, SHT_PROGBITS, MSG_ORIG(MSG_SCN_GOT), 0,
1333             &isec, &shdr, &data) == S_ERROR)
1334                 return (S_ERROR);

1336         data->d_size = size;

1338         shdr->sh_flags |= SHF_WRITE;
1339         shdr->sh_size = (Xword)size;
1340         shdr->sh_entsize = ld_targ.t_m.m_got_entsize;

1342         ofl->ofl_osgot = ld_place_section(ofl, isec, NULL,
1343             ld_targ.t_id.id_got, NULL);
1344         if (ofl->ofl_osgot == (Os_desc *)S_ERROR)
1345                 return (S_ERROR);

1347         ofl->ofl_osgot->os_szoutrels = (Xword)rsize;

1349         return (1);
1350 }

1352 /*
1353  * Build an interpreter section.
1354  */
1355 static uintptr_t
1356 make_interp(Ofl_desc *ofl)
1357 {
1358         Shdr            *shdr;
1359         Elf_Data        *data;
1360         Is_desc         *isec;
1361         const char      *iname = ofl->ofl_interp;
1362         size_t          size;

1364         /*
1365          * If -z nointerp is in effect, don't create an interpreter section.
1366          */
1367         if (ofl->ofl_flags1 & FLG_OF1_NOINTRP)
1368                 return (1);

1370         /*
1371          * An .interp section is always created for a dynamic executable.
1372          * A user can define the interpreter to use.  This definition overrides
1373          * the default that would be recorded in an executable, and triggers
1374          * the creation of an .interp section in any other object.  Presumably
1375          * the user knows what they are doing.  Refer to the generic ELF ABI
1376          * section 5-4, and the ld(1) -I option.
1377          */
1378         if (((ofl->ofl_flags & (FLG_OF_DYNAMIC | FLG_OF_EXEC |
```



```
1379             FLG_OF_RELOBJ)) != (FLG_OF_DYNAMIC | FLG_OF_EXEC)) && !iname)
1380                 return (1);

1382         /*
1383          * In the case of a dynamic executable, supply a default interpreter
1384          * if the user has not specified their own.
1385          */
1386         if (iname == NULL)
1387                 iname = ofl->ofl_interp = ld_targ.t_m.m_def_interp;

1389         size = strlen(iname) + 1;

1391         if (new_section(ofl, SHT_PROGBITS, MSG_ORIG(MSG_SCN_INTERP), 0,
1392             &isec, &shdr, &data) == S_ERROR)
1393                 return (S_ERROR);

1395         data->d_size = size;
1396         shdr->sh_size = (Xword)size;
1397         data->d_align = shdr->sh_addralign = 1;

1399         ofl->ofl_osinterp =
1400             ld_place_section(ofl, isec, NULL, ld_targ.t_id.id_interp, NULL);
1401         return ((uintptr_t)ofl->ofl_osinterp);
1402 }

1404 /*
1405  * Common function used to build the SHT_SUNW_versym section, SHT_SUNW_syminfo
1406  * section, and SHT_SUNW_capinfo section.  Each of these sections provide
1407  * additional symbol information, and their size parallels the associated
1408  * symbol table.
1409  */
1410 static Os_desc *
1411 make_sym_sec(Ofl_desc *ofl, const char *sectname, Word stype, int ident)
1412 {
1413         Shdr            *shdr;
1414         Elf_Data        *data;
1415         Is_desc         *isec;

1417         /*
1418          * We don't know the size of this section yet, so set it to 0.  The
1419          * size gets filled in after the associated symbol table is sized.
1420          */
1421         if (new_section(ofl, stype, sectname, 0, &isec, &shdr, &data) ==
1422             S_ERROR)
1423                 return ((Os_desc *)S_ERROR);

1425         return (ld_place_section(ofl, isec, NULL, ident, NULL));
1426 }

1428 /*
1429  * Determine whether a symbol capability is redundant because the object
1430  * capabilities are more restrictive.
1431  */
1432 inline static int
1433 is_cap_redundant(Objcapset *ocapset, Objcapset *scapset)
1434 {
1435         Alist           *oalp, *salp;
1436         elfcap_mask_t   omsk, smsk;

1438         /*
1439          * Inspect any platform capabilities.  If the object defines platform
1440          * capabilities, then the object will only be loaded for those
1441          * platforms.  A symbol capability set that doesn't define the same
1442          * platforms is redundant, and a symbol capability that does not provide
1443          * at least one platform name that matches a platform name in the object
1444          * capabilities will never execute (as the object wouldn't have been
```

```
1445          * loaded).
1446          */
1447         oalp = ocapset->oc_plat.cl_val;
1448         salp = scapset->oc_plat.cl_val;
1449         if (oalp && ((salp == NULL) || cap_names_match(oalp, salp)))
1450                 return (1);

1452         /*
1453          * If the symbol capability set defines platforms, and the object
1454          * doesn't, then the symbol set is more restrictive.
1455          */
1456         if (salp && (oalp == NULL))
1457                 return (0);

1459         /*
1460          * Next, inspect any machine name capabilities.  If the object defines
1461          * machine name capabilities, then the object will only be loaded for
1462          * those machines.  A symbol capability set that doesn't define the same
1463          * machine names is redundant, and a symbol capability that does not
1464          * provide at least one machine name that matches a machine name in the
1465          * object capabilities will never execute (as the object wouldn't have
1466          * been loaded).
1467          */
1468         oalp = ocapset->oc_plat.cl_val;
1469         salp = scapset->oc_plat.cl_val;
1470         if (oalp && ((salp == NULL) || cap_names_match(oalp, salp)))
1471                 return (1);

1473         /*
1474          * If the symbol capability set defines machine names, and the object
1475          * doesn't, then the symbol set is more restrictive.
1476          */
1477         if (salp && (oalp == NULL))
1478                 return (0);

1480         /*
1481          * Next, inspect any hardware capabilities.  If the objects hardware
1482          * capabilities are greater than or equal to that of the symbols
1483          * capabilities, then the symbol capability set is redundant.  If the
1484          * symbols hardware capabilities are greater that the objects, then the
1485          * symbol set is more restrictive.
1486          *
1487          * Note that this is a somewhat arbitrary definition, as each capability
1488          * bit is independent of the others, and some of the higher order bits
1489          * could be considered to be less important than lower ones.  However,
1490          * this is the only reasonable non-subjective definition.
1491          */
1492         omsk = ocapset->oc_hw_2.cm_val;
1493         smsk = scapset->oc_hw_2.cm_val;
1494         if ((omsk > smsk) || (omsk && (omsk == smsk)))
1495                 return (1);
1496         if (omsk < smsk)
1497                 return (0);

1499         /*
1500          * Finally, inspect the remaining hardware capabilities.
1501          */
1502         omsk = ocapset->oc_hw_1.cm_val;
1503         smsk = scapset->oc_hw_1.cm_val;
1504         if ((omsk > smsk) || (omsk && (omsk == smsk)))
1505                 return (1);

1507         return (0);
1508 }

1510 /*
```



```
1511  * Capabilities values might have been assigned excluded values.  These
1512  * excluded values should be removed before calculating any capabilities
1513  * sections size.
1514  */
1515 static void
1516 capmask_value(Lm_list *lml, Word type, Capmask *capmask, int *title)
1517 {
1518         /*
1519          * First determine whether any bits should be excluded.
1520          */
1521         if ((capmask->cm_val & capmask->cm_exc) == 0)
1522                 return;

1524         DBG_CALL(Dbg_cap_post_title(lml, title));

1526         DBG_CALL(Dbg_cap_val_entry(lml, DBG_STATE_CURRENT, type,
1527             capmask->cm_val, ld_targ.t_m.m_mach));
1528         DBG_CALL(Dbg_cap_val_entry(lml, DBG_STATE_EXCLUDE, type,
1529             capmask->cm_exc, ld_targ.t_m.m_mach));

1531         capmask->cm_val &= ~capmask->cm_exc;

1533         DBG_CALL(Dbg_cap_val_entry(lml, DBG_STATE_RESOLVED, type,
1534             capmask->cm_val, ld_targ.t_m.m_mach));
1535 }

1537 static void
1538 capstr_value(Lm_list *lml, Word type, Caplist *caplist, int *title)
1539 {
1540         Aliste  idx1, idx2;
1541         char    *estr;
1542         Capstr  *capstr;
1543         Boolean found = FALSE;

1545         /*
1546          * First determine whether any strings should be excluded.
1547          */
1548         for (APLIST_TRAVERSE(caplist->cl_exc, idx1, estr)) {
1549                 for (ALIST_TRAVERSE(caplist->cl_val, idx2, capstr)) {
1550                         if (strcmp(estr, capstr->cs_str) == 0) {
1551                                 found = TRUE;
1552                                 break;
1553                         }
1554                 }
1555         }

1557         if (found == FALSE)
1558                 return;

1560         /*
1561          * Traverse the current strings, then delete the excluded strings,
1562          * and finally display the resolved strings.
1563          */
1564         if (DBG_ENABLED) {
1565                 Dbg_cap_post_title(lml, title);
1566                 for (ALIST_TRAVERSE(caplist->cl_val, idx2, capstr)) {
1567                         Dbg_cap_ptr_entry(lml, DBG_STATE_CURRENT, type,
1568                             capstr->cs_str);
1569                 }
1570         }
1571         for (APLIST_TRAVERSE(caplist->cl_exc, idx1, estr)) {
1572                 for (ALIST_TRAVERSE(caplist->cl_val, idx2, capstr)) {
1573                         if (strcmp(estr, capstr->cs_str) == 0) {
1574                                 DBG_CALL(Dbg_cap_ptr_entry(lml,
1575                                     DBG_STATE_EXCLUDE, type, capstr->cs_str));
1576                                 alist_delete(caplist->cl_val, &idx2);
```

```
1577                                    break;
1578                            }
1579                    }
1580            }
1581            if (DBG_ENABLED) {
1582                    for (ALIST_TRAVERSE(caplist->cl_val, idx2, capstr)) {
1583                            Dbg_cap_ptr_entry(lml, DBG_STATE_RESOLVED, type,
1584                                capstr->cs_str);
1585                    }
1586            }
1587    }

1589    /*
1590     * Build a capabilities section.
1591     */
1592    #define CAP_UPDATE(cap, capndx, tag, val)       \
1593            cap->c_tag = tag; \
1594            cap->c_un.c_val = val; \
1595            cap++, capndx++;

1597    static uintptr_t
1598    make_cap(Ofl_desc *ofl, Word shtype, const char *shname, int ident)
1599    {
1600            Shdr            *shdr;
1601            Elf_Data        *data;
1602            Is_desc         *isec;
1603            Cap             *cap;
1604            size_t          size = 0;
1605            Word            capndx = 0;
1606            Str_tbl         *strtbl;
1607            Objcapset       *ocapset = &ofl->ofl_ocapset;
1608            Aliste          idx1;
1609            Capstr          *capstr;
1610            int             title = 0;

1612            /*
1613             * Determine which string table to use for any CA_SUNW_MACH,
1614             * CA_SUNW_PLAT, or CA_SUNW_ID strings.
1615             */
1616            if (OFL_IS_STATIC_OBJ(ofl))
1617                    strtbl = ofl->ofl_strtab;
1618            else
1619                    strtbl = ofl->ofl_dynstrtab;

1621            /*
1622             * If symbol capabilities have been requested, but none have been
1623             * created, warn the user.  This scenario can occur if none of the
1624             * input relocatable objects defined any object capabilities.
1625             */
1626            if ((ofl->ofl_flags & FLG_OF_OTOSCAP) && (ofl->ofl_capsymcnt == 0))
1627                    ld_eprintf(ofl, ERR_WARNING, MSG_INTL(MSG_CAP_NOSYMSFOUND));

1629            /*
1630             * If symbol capabilities have been collected, but no symbols are left
1631             * referencing these capabilities, promote the capability groups back
1632             * to an object capability definition.
1633             */
1634            if ((ofl->ofl_flags & FLG_OF_OTOSCAP) && ofl->ofl_capsymcnt &&
1635                (ofl->ofl_capfamilies == NULL)) {
1636                    ld_eprintf(ofl, ERR_WARNING, MSG_INTL(MSG_CAP_NOSYMSFOUND));
1637                    ld_cap_move_symtoobj(ofl);
1638                    ofl->ofl_capsymcnt = 0;
1639                    ofl->ofl_capgroups = NULL;
1640                    ofl->ofl_flags &= ~FLG_OF_OTOSCAP;
1641            }
```



```
1643            /*
1644             * Remove any excluded capabilities.
1645             */
1646            capstr_value(ofl->ofl_lml, CA_SUNW_PLAT, &ocapset->oc_plat, &title);
1647            capstr_value(ofl->ofl_lml, CA_SUNW_MACH, &ocapset->oc_mach, &title);
1648            capmask_value(ofl->ofl_lml, CA_SUNW_HW_2, &ocapset->oc_hw_2, &title);
1649            capmask_value(ofl->ofl_lml, CA_SUNW_HW_1, &ocapset->oc_hw_1, &title);
1650            capmask_value(ofl->ofl_lml, CA_SUNW_SF_1, &ocapset->oc_sf_1, &title);

1652            /*
1653             * Determine how many entries are required for any object capabilities.
1654             */
1655            size += alist_nitems(ocapset->oc_plat.cl_val);
1656            size += alist_nitems(ocapset->oc_mach.cl_val);
1657            if (ocapset->oc_hw_2.cm_val)
1658                    size++;
1659            if (ocapset->oc_hw_1.cm_val)
1660                    size++;
1661            if (ocapset->oc_sf_1.cm_val)
1662                    size++;

1664            /*
1665             * Only identify a capabilities group if the group has content.  If a
1666             * capabilities identifier exists, and no other capabilities have been
1667             * supplied, remove the identifier.  This scenario could exist if a
1668             * user mistakenly defined a lone identifier, or if an identified group
1669             * was overridden so as to clear the existing capabilities and the
1670             * identifier was not also cleared.
1671             */
1672            if (ocapset->oc_id.cs_str) {
1673                    if (size)
1674                            size++;
1675                    else
1676                            ocapset->oc_id.cs_str = NULL;
1677            }
1678            if (size)
1679                    size++;                 /* Add CA_SUNW_NULL */

1681            /*
1682             * Determine how many entries are required for any symbol capabilities.
1683             */
1684            if (ofl->ofl_capsymcnt) {
1685                    /*
1686                     * If there are no object capabilities, a CA_SUNW_NULL entry
1687                     * is required before any symbol capabilities.
1688                     */
1689                    if (size == 0)
1690                            size++;
1691                    size += ofl->ofl_capsymcnt;
1692            }

1694            if (size == 0)
1695                    return (NULL);

1697            if (new_section(ofl, shtype, shname, size, &isec,
1698                &shdr, &data) == S_ERROR)
1699                    return (S_ERROR);

1701            if ((data->d_buf = libld_malloc(shdr->sh_size)) == NULL)
1702                    return (S_ERROR);

1704            cap = (Cap *)data->d_buf;

1706            /*
1707             * Fill in any object capabilities.  If there is an identifier, then the
1708             * identifier comes first.  The remaining items follow in precedence
```

```
1709          * order, although the order isn't important for runtime verification.
1710          */
1711         if (ocapset->oc_id.cs_str) {
1712                 ofl->ofl_flags |= FLG_OF_CAPSTRS;
1713                 if (st_insert(strtbl, ocapset->oc_id.cs_str) == -1)
1714                         return (S_ERROR);
1715                 ocapset->oc_id.cs_ndx = capndx;
1716                 CAP_UPDATE(cap, capndx, CA_SUNW_ID, 0);
1717         }
1718         if (ocapset->oc_plat.cl_val) {
1719                 ofl->ofl_flags |= (FLG_OF_PTCAP | FLG_OF_CAPSTRS);

1721                 /*
1722                  * Insert any platform name strings in the appropriate string
1723                  * table.  The capability value can't be filled in yet, as the
1724                  * final offset of the strings isn't known until later.
1725                  */
1726                 for (ALIST_TRAVERSE(ocapset->oc_plat.cl_val, idx1, capstr)) {
1727                         if (st_insert(strtbl, capstr->cs_str) == -1)
1728                                 return (S_ERROR);
1729                         capstr->cs_ndx = capndx;
1730                         CAP_UPDATE(cap, capndx, CA_SUNW_PLAT, 0);
1731                 }
1732         }
1733         if (ocapset->oc_mach.cl_val) {
1734                 ofl->ofl_flags |= (FLG_OF_PTCAP | FLG_OF_CAPSTRS);

1736                 /*
1737                  * Insert the machine name strings in the appropriate string
1738                  * table.  The capability value can't be filled in yet, as the
1739                  * final offset of the strings isn't known until later.
1740                  */
1741                 for (ALIST_TRAVERSE(ocapset->oc_mach.cl_val, idx1, capstr)) {
1742                         if (st_insert(strtbl, capstr->cs_str) == -1)
1743                                 return (S_ERROR);
1744                         capstr->cs_ndx = capndx;
1745                         CAP_UPDATE(cap, capndx, CA_SUNW_MACH, 0);
1746                 }
1747         }
1748         if (ocapset->oc_hw_2.cm_val) {
1749                 ofl->ofl_flags |= FLG_OF_PTCAP;
1750                 CAP_UPDATE(cap, capndx, CA_SUNW_HW_2, ocapset->oc_hw_2.cm_val);
1751         }
1752         if (ocapset->oc_hw_1.cm_val) {
1753                 ofl->ofl_flags |= FLG_OF_PTCAP;
1754                 CAP_UPDATE(cap, capndx, CA_SUNW_HW_1, ocapset->oc_hw_1.cm_val);
1755         }
1756         if (ocapset->oc_sf_1.cm_val) {
1757                 ofl->ofl_flags |= FLG_OF_PTCAP;
1758                 CAP_UPDATE(cap, capndx, CA_SUNW_SF_1, ocapset->oc_sf_1.cm_val);
1759         }
1760         CAP_UPDATE(cap, capndx, CA_SUNW_NULL, 0);

1762         /*
1763          * Fill in any symbol capabilities.
1764          */
1765         if (ofl->ofl_capgroups) {
1766                 Cap_group       *cgp;

1768                 for (APLIST_TRAVERSE(ofl->ofl_capgroups, idx1, cgp)) {
1769                         Objcapset       *scapset = &cgp->cg_set;
1770                         Aliste          idx2;
1771                         Is_desc         *isp;

1773                         cgp->cg_ndx = capndx;
```



```
1775                         if (scapset->oc_id.cs_str) {
1776                                 ofl->ofl_flags |= FLG_OF_CAPSTRS;
1777                                 /*
1778                                  * Insert the identifier string in the
1779                                  * appropriate string table.  The capability
1780                                  * value can't be filled in yet, as the final
1781                                  * offset of the string isn't known until later.
1782                                  */
1783                                 if (st_insert(strtbl,
1784                                     scapset->oc_id.cs_str) == -1)
1785                                         return (S_ERROR);
1786                                 scapset->oc_id.cs_ndx = capndx;
1787                                 CAP_UPDATE(cap, capndx, CA_SUNW_ID, 0);
1788                         }

1790                         if (scapset->oc_plat.cl_val) {
1791                                 ofl->ofl_flags |= FLG_OF_CAPSTRS;

1793                                 /*
1794                                  * Insert the platform name string in the
1795                                  * appropriate string table.  The capability
1796                                  * value can't be filled in yet, as the final
1797                                  * offset of the string isn't known until later.
1798                                  */
1799                                 for (ALIST_TRAVERSE(scapset->oc_plat.cl_val,
1800                                     idx2, capstr)) {
1801                                         if (st_insert(strtbl,
1802                                             capstr->cs_str) == -1)
1803                                                 return (S_ERROR);
1804                                         capstr->cs_ndx = capndx;
1805                                         CAP_UPDATE(cap, capndx,
1806                                             CA_SUNW_PLAT, 0);
1807                                 }
1808                         }
1809                         if (scapset->oc_mach.cl_val) {
1810                                 ofl->ofl_flags |= FLG_OF_CAPSTRS;

1812                                 /*
1813                                  * Insert the machine name string in the
1814                                  * appropriate string table.  The capability
1815                                  * value can't be filled in yet, as the final
1816                                  * offset of the string isn't known until later.
1817                                  */
1818                                 for (ALIST_TRAVERSE(scapset->oc_mach.cl_val,
1819                                     idx2, capstr)) {
1820                                         if (st_insert(strtbl,
1821                                             capstr->cs_str) == -1)
1822                                                 return (S_ERROR);
1823                                         capstr->cs_ndx = capndx;
1824                                         CAP_UPDATE(cap, capndx,
1825                                             CA_SUNW_MACH, 0);
1826                                 }
1827                         }
1828                         if (scapset->oc_hw_2.cm_val) {
1829                                 CAP_UPDATE(cap, capndx, CA_SUNW_HW_2,
1830                                     scapset->oc_hw_2.cm_val);
1831                         }
1832                         if (scapset->oc_hw_1.cm_val) {
1833                                 CAP_UPDATE(cap, capndx, CA_SUNW_HW_1,
1834                                     scapset->oc_hw_1.cm_val);
1835                         }
1836                         if (scapset->oc_sf_1.cm_val) {
1837                                 CAP_UPDATE(cap, capndx, CA_SUNW_SF_1,
1838                                     scapset->oc_sf_1.cm_val);
1839                         }
1840                         CAP_UPDATE(cap, capndx, CA_SUNW_NULL, 0);
```

```
1842                        /*
1843                         * If any object capabilities are available, determine
1844                         * whether these symbol capabilities are less
1845                         * restrictive, and hence redundant.
1846                         */
1847                        if (((ofl->ofl_flags & FLG_OF_PTCAP) == 0) ||
1848                            (is_cap_redundant(ocapset, scapset) == 0))
1849                                continue;

1851                        /*
1852                         * Indicate any files that provide redundant symbol
1853                         * capabilities.
1854                         */
1855                        for (APLIST_TRAVERSE(cgp->cg_secs, idx2, isp)) {
1856                                ld_eprintf(ofl, ERR_WARNING,
1857                                    MSG_INTL(MSG_CAP_REDUNDANT),
1858                                    isp->is_file->ifl_name,
1859                                    EC_WORD(isp->is_scnndx), isp->is_name);
1860                        }
1861                }
1862        }

1864        /*
1865         * If capabilities strings are required, the sh_info field of the
1866         * section header will be set to the associated string table.
1867         */
1868        if (ofl->ofl_flags & FLG_OF_CAPSTRS)
1869                shdr->sh_flags |= SHF_INFO_LINK;

1871        /*
1872         * Place these capabilities in the output file.
1873         */
1874        if ((ofl->ofl_oscap = ld_place_section(ofl, isec,
1875            NULL, ident, NULL)) == (Os_desc *)S_ERROR)
1876                return (S_ERROR);

1878        /*
1879         * If symbol capabilities are required, then a .SUNW_capinfo section is
1880         * also created.  This table will eventually be sized to match the
1881         * associated symbol table.
1882         */
1883        if (ofl->ofl_capfamilies) {
1884                if ((ofl->ofl_oscapinfo = make_sym_sec(ofl,
1885                    MSG_ORIG(MSG_SCN_SUNWCAPINFO), SHT_SUNW_capinfo,
1886                    ld_targ.t_id.id_capinfo)) == (Os_desc *)S_ERROR)
1887                        return (S_ERROR);

1889                /*
1890                 * If we're generating a dynamic object, capabilities family
1891                 * members are maintained in a .SUNW_capchain section.
1892                 */
1893                if (ofl->ofl_capchaincnt &&
1894                    ((ofl->ofl_flags & FLG_OF_RELOBJ) == 0)) {
1895                        if (new_section(ofl, SHT_SUNW_capchain,
1896                            MSG_ORIG(MSG_SCN_SUNWCAPCHAIN),
1897                            ofl->ofl_capchaincnt, &isec, &shdr,
1898                            &data) == S_ERROR)
1899                                return (S_ERROR);

1901                        ofl->ofl_oscapchain = ld_place_section(ofl, isec,
1902                            NULL, ld_targ.t_id.id_capchain, NULL);
1903                        if (ofl->ofl_oscapchain == (Os_desc *)S_ERROR)
1904                                return (S_ERROR);

1906                }
```



```
1907        }
1908        return (1);
1909 }
1910 #undef  CAP_UPDATE

1912 /*
1913  * Build the PLT section and its associated relocation entries.
1914  */
1915 static uintptr_t
1916 make_plt(Ofl_desc *ofl)
1917 {
1918        Shdr            *shdr;
1919        Elf_Data        *data;
1920        Is_desc         *isec;
1921        size_t          size = ld_targ.t_m.m_plt_reservsz +
1922            (((size_t)ofl->ofl_pltcnt + (size_t)ofl->ofl_pltpad) *
1923            ld_targ.t_m.m_plt_entsize);
1924        size_t          rsize = (size_t)ofl->ofl_relocpltsz;

1926        /*
1927         * On sparc, account for the NOP at the end of the plt.
1928         */
1929        if (ld_targ.t_m.m_mach == LD_TARG_BYCLASS(EM_SPARC, EM_SPARCV9))
1930                size += sizeof (Word);

1932        if (new_section(ofl, SHT_PROGBITS, MSG_ORIG(MSG_SCN_PLT), 0,
1933            &isec, &shdr, &data) == S_ERROR)
1934                return (S_ERROR);

1936        data->d_size = size;
1937        data->d_align = ld_targ.t_m.m_plt_align;

1939        shdr->sh_flags = ld_targ.t_m.m_plt_shf_flags;
1940        shdr->sh_size = (Xword)size;
1941        shdr->sh_addralign = ld_targ.t_m.m_plt_align;
1942        shdr->sh_entsize = ld_targ.t_m.m_plt_entsize;

1944        ofl->ofl_osplt = ld_place_section(ofl, isec, NULL,
1945            ld_targ.t_id.id_plt, NULL);
1946        if (ofl->ofl_osplt == (Os_desc *)S_ERROR)
1947                return (S_ERROR);

1949        ofl->ofl_osplt->os_szoutrels = (Xword)rsize;

1951        return (1);
1952 }

1954 /*
1955  * Make the hash table.  Only built for dynamic executables and shared
1956  * libraries, and provides hashed lookup into the global symbol table
1957  * (.dynsym) for the run-time linker to resolve symbol lookups.
1958  */
1959 static uintptr_t
1960 make_hash(Ofl_desc *ofl)
1961 {
1962        Shdr            *shdr;
1963        Elf_Data        *data;
1964        Is_desc         *isec;
1965        size_t          size;
1966        Word            nsyms = ofl->ofl_globcnt;
1967        size_t          cnt;

1969        /*
1970         * Allocate section header structures. We set entcnt to 0
1971         * because it's going to change after we place this section.
1972         */
```

```
1973         if (new_section(ofl, SHT_HASH, MSG_ORIG(MSG_SCN_HASH), 0,
1974             &isec, &shdr, &data) == S_ERROR)
1975                 return (S_ERROR);

1977         /*
1978          * Place the section first since it will affect the local symbol
1979          * count.
1980          */
1981         ofl->ofl_oshash =
1982             ld_place_section(ofl, isec, NULL, ld_targ.t_id.id_hash, NULL);
1983         if (ofl->ofl_oshash == (Os_desc *)S_ERROR)
1984                 return (S_ERROR);

1986         /*
1987          * Calculate the number of output hash buckets.
1988          */
1989         ofl->ofl_hashbkts = findprime(nsyms);

1991         /*
1992          * The size of the hash table is determined by
1993          *
1994          *      i.      the initial nbucket and nchain entries (2)
1995          *      ii.     the number of buckets (calculated above)
1996          *      iii.    the number of chains (this is based on the number of
1997          *              symbols in the .dynsym array).
1998          */
1999         cnt = 2 + ofl->ofl_hashbkts + DYNSYM_ALL_CNT(ofl);
2000         size = cnt * shdr->sh_entsize;

2002         /*
2003          * Finalize the section header and data buffer initialization.
2004          */
2005         if ((data->d_buf = libld_calloc(size, 1)) == NULL)
2006                 return (S_ERROR);
2007         data->d_size = size;
2008         shdr->sh_size = (Xword)size;

2010         return (1);
2011 }

2013 /*
2014  * Generate the standard symbol table.  Contains all locals and globals,
2015  * and resides in a non-allocatable section (ie. it can be stripped).
2016  */
2017 static uintptr_t
2018 make_symtab(Ofl_desc *ofl)
2019 {
2020         Shdr            *shdr;
2021         Elf_Data        *data;
2022         Is_desc         *isec;
2023         Is_desc         *xisec = 0;
2024         size_t          size;
2025         Word            symcnt;

2027         /*
2028          * Create the section headers. Note that we supply an ent_cnt
2029          * of 0. We won't know the count until the section has been placed.
2030          */
2031         if (new_section(ofl, SHT_SYMTAB, MSG_ORIG(MSG_SCN_SYMTAB), 0,
2032             &isec, &shdr, &data) == S_ERROR)
2033                 return (S_ERROR);

2035         /*
2036          * Place the section first since it will affect the local symbol
2037          * count.
2038          */
```



```
2039         if ((ofl->ofl_ossymtab = ld_place_section(ofl, isec, NULL,
2040             ld_targ.t_id.id_symtab, NULL)) == (Os_desc *)S_ERROR)
2041                 return (S_ERROR);

2043         /*
2044          * At this point we've created all but the 'shstrtab' section.
2045          * Determine if we have to use 'Extended Sections'.  If so - then
2046          * also create a SHT_SYMTAB_SHNDX section.
2047          */
2048         if ((ofl->ofl_shdrcnt + 1) >= SHN_LORESERVE) {
2049                 Shdr            *xshdr;
2050                 Elf_Data        *xdata;

2052                 if (new_section(ofl, SHT_SYMTAB_SHNDX,
2053                     MSG_ORIG(MSG_SCN_SYMTAB_SHNDX), 0, &xisec,
2054                     &xshdr, &xdata) == S_ERROR)
2055                         return (S_ERROR);

2057                 if ((ofl->ofl_ossymshndx = ld_place_section(ofl, xisec, NULL,
2058                     ld_targ.t_id.id_symtab_ndx, NULL)) == (Os_desc *)S_ERROR)
2059                         return (S_ERROR);
2060         }

2062         /*
2063          * Calculated number of symbols, which need to be augmented by
2064          * the (yet to be created) .shstrtab entry.
2065          */
2066         symcnt = (size_t)(1 + SYMTAB_ALL_CNT(ofl));
2067         size = symcnt * shdr->sh_entsize;

2069         /*
2070          * Finalize the section header and data buffer initialization.
2071          */
2072         data->d_size = size;
2073         shdr->sh_size = (Xword)size;

2075         /*
2076          * If we created a SHT_SYMTAB_SHNDX - then set it's sizes too.
2077          */
2078         if (xisec) {
2079                 size_t  xsize = symcnt * sizeof (Word);

2081                 xisec->is_indata->d_size = xsize;
2082                 xisec->is_shdr->sh_size = (Xword)xsize;
2083         }

2085         return (1);
2086 }

2088 /*
2089  * Build a dynamic symbol table. These tables reside in the text
2090  * segment of a dynamic executable or shared library.
2091  *
2092  *      .SUNW_ldynsym contains local function symbols
2093  *      .dynsym contains only globals symbols
2094  *
2095  * The two tables are created adjacent to each other, with .SUNW_ldynsym
2096  * coming first.
2097  */
2098 static uintptr_t
2099 make_dynsym(Ofl_desc *ofl)
2100 {
2101         Shdr            *shdr, *lshdr;
2102         Elf_Data        *data, *ldata;
2103         Is_desc         *isec, *lisec;
2104         size_t          size;
```

```
2105         Xword           cnt;
2106         int             allow_ldynsym;
2107
2108         /*
2109          * Unless explicitly disabled, always produce a .SUNW_ldynsym section
2110          * when it is allowed by the file type, even if the resulting
2111          * table only ends up with a single STT_FILE in it. There are
2112          * two reasons: (1) It causes the generation of the DT_SUNW_SYMTAB
2113          * entry in the .dynamic section, which is something we would
2114          * like to encourage, and (2) Without it, we cannot generate
2115          * the associated .SUNW_dyn[sym|tls]sort sections, which are of
2116          * value to DTrace.
2117          *
2118          * In practice, it is extremely rare for an object not to have
2119          * local symbols for .SUNW_ldynsym, so 99% of the time, we'd be
2120          * doing it anyway.
2121          */
2122         allow_ldynsym = OFL_ALLOW_LDYNSYM(ofl);
2123
2124         /*
2125          * Create the section headers. Note that we supply an ent_cnt
2126          * of 0. We won't know the count until the section has been placed.
2127          */
2128         if (allow_ldynsym && new_section(ofl, SHT_SUNW_LDYNSYM,
2129             MSG_ORIG(MSG_SCN_LDYNSYM), 0, &lisec, &lshdr, &ldata) == S_ERROR)
2130                 return (S_ERROR);
2131
2132         if (new_section(ofl, SHT_DYNSYM, MSG_ORIG(MSG_SCN_DYNSYM), 0,
2133             &isec, &shdr, &data) == S_ERROR)
2134                 return (S_ERROR);
2135
2136         /*
2137          * Place the section(s) first since it will affect the local symbol
2138          * count.
2139          */
2140         if (allow_ldynsym &&
2141             ((ofl->ofl_osldynsym = ld_place_section(ofl, lisec, NULL,
2142             ld_targ.t_id.id_ldynsym, NULL)) == (Os_desc *)S_ERROR))
2143                 return (S_ERROR);
2144         ofl->ofl_osdynsym =
2145             ld_place_section(ofl, isec, NULL, ld_targ.t_id.id_dynsym, NULL);
2146         if (ofl->ofl_osdynsym == (Os_desc *)S_ERROR)
2147                 return (S_ERROR);
2148
2149         cnt = DYNSYM_ALL_CNT(ofl);
2150         size = (size_t)cnt * shdr->sh_entsize;
2151
2152         /*
2153          * Finalize the section header and data buffer initialization.
2154          */
2155         data->d_size = size;
2156         shdr->sh_size = (Xword)size;
2157
2158         /*
2159          * An ldynsym contains local function symbols. It is not
2160          * used for linking, but if present, serves to allow better
2161          * stack traces to be generated in contexts where the symtab
2162          * is not available. (dladdr(), or stripped executable/library files).
2163          */
2164         if (allow_ldynsym) {
2165                 cnt = 1 + ofl->ofl_dynlocscnt + ofl->ofl_dynscopecnt;
2166                 size = (size_t)cnt * shdr->sh_entsize;
2167
2168                 ldata->d_size = size;
2169                 lshdr->sh_size = (Xword)size;
2170         }
```



```
2171
2172         return (1);
2173 }
2174
2175 /*
2176  * Build .SUNW_dynsymsort and/or .SUNW_dyntlssort sections. These are
2177  * index sections for the .SUNW_ldynsym/.dynsym pair that present data
2178  * and function symbols sorted by address.
2179  */
2180 static uintptr_t
2181 make_dynsort(Ofl_desc *ofl)
2182 {
2183         Shdr            *shdr;
2184         Elf_Data        *data;
2185         Is_desc         *isec;
2186
2187         /* Only do it if the .SUNW_ldynsym section is present */
2188         if (!OFL_ALLOW_LDYNSYM(ofl))
2189                 return (1);
2190
2191         /* .SUNW_dynsymsort */
2192         if (ofl->ofl_dynsymsortcnt > 0) {
2193                 if (new_section(ofl, SHT_SUNW_symsort,
2194                     MSG_ORIG(MSG_SCN_DYNSYMSORT), ofl->ofl_dynsymsortcnt,
2195                     &isec, &shdr, &data) == S_ERROR)
2196                 return (S_ERROR);
2197
2198                 if ((ofl->ofl_osdynsymsort = ld_place_section(ofl, isec, NULL,
2199                     ld_targ.t_id.id_dynsort, NULL)) == (Os_desc *)S_ERROR)
2200                         return (S_ERROR);
2201         }
2202
2203         /* .SUNW_dyntlssort */
2204         if (ofl->ofl_dyntlssortcnt > 0) {
2205                 if (new_section(ofl, SHT_SUNW_tlssort,
2206                     MSG_ORIG(MSG_SCN_DYNTLSSORT),
2207                     ofl->ofl_dyntlssortcnt, &isec, &shdr, &data) == S_ERROR)
2208                 return (S_ERROR);
2209
2210                 if ((ofl->ofl_osdyntlssort = ld_place_section(ofl, isec, NULL,
2211                     ld_targ.t_id.id_dynsort, NULL)) == (Os_desc *)S_ERROR)
2212                         return (S_ERROR);
2213         }
2214
2215         return (1);
2216 }
2217
2218 /*
2219  * Helper routine for make_dynsym_shndx. Builds a
2220  * a SHT_SYMTAB_SHNDX for .dynsym or .SUNW_ldynsym, without knowing
2221  * which one it is.
2222  */
2223 static uintptr_t
2224 make_dyn_shndx(Ofl_desc *ofl, const char *shname, Os_desc *symtab,
2225     Os_desc **ret_os)
2226 {
2227         Is_desc         *isec;
2228         Is_desc         *dynsymisp;
2229         Shdr            *shdr, *dynshdr;
2230         Elf_Data        *data;
2231
2232         dynsymisp = ld_os_first_isdesc(symtab);
2233         dynshdr = dynsymisp->is_shdr;
2234
2235         if (new_section(ofl, SHT_SYMTAB_SHNDX, shname,
2236             (dynshdr->sh_size / dynshdr->sh_entsize),
```

```
2237             &isec, &shdr, &data) == S_ERROR)
2238                 return (S_ERROR);

2240         if ((*ret_os = ld_place_section(ofl, isec, NULL,
2241             ld_targ.t_id.id_dynsym_ndx, NULL)) == (Os_desc *)S_ERROR)
2242                 return (S_ERROR);

2244         assert(*ret_os);

2246         return (1);
2247 }

2249 /*
2250  * Build a SHT_SYMTAB_SHNDX for the .dynsym, and .SUNW_ldynsym
2251  */
2252 static uintptr_t
2253 make_dynsym_shndx(Ofl_desc *ofl)
2254 {
2255         /*
2256          * If there is a .SUNW_ldynsym, generate a section for its extended
2257          * index section as well.
2258          */
2259         if (OFL_ALLOW_LDYNSYM(ofl)) {
2260                 if (make_dyn_shndx(ofl, MSG_ORIG(MSG_SCN_LDYNSYM_SHNDX),
2261                     ofl->ofl_osldynsym, &ofl->ofl_osldynshndx) == S_ERROR)
2262                         return (S_ERROR);
2263         }

2265         /* The Generate a section for the dynsym */
2266         if (make_dyn_shndx(ofl, MSG_ORIG(MSG_SCN_DYNSYM_SHNDX),
2267             ofl->ofl_osdynsym, &ofl->ofl_osdynshndx) == S_ERROR)
2268                 return (S_ERROR);

2270         return (1);
2271 }


2274 /*
2275  * Build a string table for the section headers.
2276  */
2277 static uintptr_t
2278 make_shstrtab(Ofl_desc *ofl)
2279 {
2280         Shdr            *shdr;
2281         Elf_Data        *data;
2282         Is_desc         *isec;
2283         size_t          size;

2285         if (new_section(ofl, SHT_STRTAB, MSG_ORIG(MSG_SCN_SHSTRTAB),
2286             0, &isec, &shdr, &data) == S_ERROR)
2287                 return (S_ERROR);

2289         /*
2290          * Place the section first, as it may effect the number of section
2291          * headers to account for.
2292          */
2293         ofl->ofl_osshstrtab =
2294             ld_place_section(ofl, isec, NULL, ld_targ.t_id.id_note, NULL);
2295         if (ofl->ofl_osshstrtab == (Os_desc *)S_ERROR)
2296                 return (S_ERROR);

2298         size = st_getstrtab_sz(ofl->ofl_shdrsttab);
2299         assert(size > 0);

2301         data->d_size = size;
2302         shdr->sh_size = (Xword)size;
```



```
2304         return (1);
2305 }

2307 /*
2308  * Build a string section for the standard symbol table.
2309  */
2310 static uintptr_t
2311 make_strtab(Ofl_desc *ofl)
2312 {
2313         Shdr            *shdr;
2314         Elf_Data        *data;
2315         Is_desc         *isec;
2316         size_t          size;

2318         /*
2319          * This string table consists of all the global and local symbols.
2320          * Account for null bytes at end of the file name and the beginning
2321          * of section.
2322          */
2323         if (st_insert(ofl->ofl_strtab, ofl->ofl_name) == -1)
2324                 return (S_ERROR);

2326         size = st_getstrtab_sz(ofl->ofl_strtab);
2327         assert(size > 0);

2329         if (new_section(ofl, SHT_STRTAB, MSG_ORIG(MSG_SCN_STRTAB),
2330             0, &isec, &shdr, &data) == S_ERROR)
2331                 return (S_ERROR);

2333         /* Set the size of the data area */
2334         data->d_size = size;
2335         shdr->sh_size = (Xword)size;

2337         ofl->ofl_osstrtab =
2338             ld_place_section(ofl, isec, NULL, ld_targ.t_id.id_strtab, NULL);
2339         return ((uintptr_t)ofl->ofl_osstrtab);
2340 }

2342 /*
2343  * Build a string table for the dynamic symbol table.
2344  */
2345 static uintptr_t
2346 make_dynstr(Ofl_desc *ofl)
2347 {
2348         Shdr            *shdr;
2349         Elf_Data        *data;
2350         Is_desc         *isec;
2351         size_t          size;

2353         /*
2354          * If producing a .SUNW_ldynsym, account for the initial STT_FILE
2355          * symbol that precedes the scope reduced global symbols.
2356          */
2357         if (OFL_ALLOW_LDYNSYM(ofl)) {
2358                 if (st_insert(ofl->ofl_dynstrtab, ofl->ofl_name) == -1)
2359                         return (S_ERROR);
2360                 ofl->ofl_dynscopecnt++;
2361         }

2363         /*
2364          * Account for any local, named register symbols.  These locals are
2365          * required for reference from DT_REGISTER .dynamic entries.
2366          */
2367         if (ofl->ofl_regsyms) {
2368                 int     ndx;
```

```
2370                 for (ndx = 0; ndx < ofl->ofl_regsymsno; ndx++) {
2371                         Sym_desc        *sdp;

2373                         if ((sdp = ofl->ofl_regsyms[ndx]) == NULL)
2374                                 continue;

2376                         if (!SYM_IS_HIDDEN(sdp) &&
2377                             (ELF_ST_BIND(sdp->sd_sym->st_info) != STB_LOCAL))
2378                                 continue;

2380                         if (sdp->sd_sym->st_name == NULL)
2381                                 continue;

2383                         if (st_insert(ofl->ofl_dynstrtab, sdp->sd_name) == -1)
2384                                 return (S_ERROR);
2385                 }
2386         }

2388         /*
2389          * Reserve entries for any per-symbol auxiliary/filter strings.
2390          */
2391         if (ofl->ofl_dtsfltrs != NULL) {
2392                 Dfltr_desc      *dftp;
2393                 Aliste          idx;

2395                 for (ALIST_TRAVERSE(ofl->ofl_dtsfltrs, idx, dftp))
2396                         if (st_insert(ofl->ofl_dynstrtab, dftp->dft_str) == -1)
2397                                 return (S_ERROR);
2398         }

2400         size = st_getstrtab_sz(ofl->ofl_dynstrtab);
2401         assert(size > 0);

2403         if (new_section(ofl, SHT_STRTAB, MSG_ORIG(MSG_SCN_DYNSTR),
2404             0, &isec, &shdr, &data) == S_ERROR)
2405                 return (S_ERROR);

2407         /* Make it allocable if necessary */
2408         if (!(ofl->ofl_flags & FLG_OF_RELOBJ))
2409                 shdr->sh_flags |= SHF_ALLOC;

2411         /* Set the size of the data area */
2412         data->d_size = size + DYNSTR_EXTRA_PAD;

2414         shdr->sh_size = (Xword)size;

2416         ofl->ofl_osdynstr =
2417             ld_place_section(ofl, isec, NULL, ld_targ.t_id.id_dynstr, NULL);
2418         return ((uintptr_t)ofl->ofl_osdynstr);
2419 }

2421 /*
2422  * Generate an output relocation section which will contain the relocation
2423  * information to be applied to the 'osp' section.
2424  *
2425  * If (osp == NULL) then we are creating the coalesced relocation section
2426  * for an executable and/or a shared object.
2427  */
2428 static uintptr_t
2429 make_reloc(Ofl_desc *ofl, Os_desc *osp)
2430 {
2431         Shdr            *shdr;
2432         Elf_Data        *data;
2433         Is_desc         *isec;
2434         size_t          size;
```



```
2435         Xword           sh_flags;
2436         char            *sectname;
2437         Os_desc         *rosp;
2438         Word            relsize;
2439         const char      *rel_prefix;

2441         /* LINTED */
2442         if (ld_targ.t_m.m_rel_sht_type == SHT_REL) {
2443                 /* REL */
2444                 relsize = sizeof (Rel);
2445                 rel_prefix = MSG_ORIG(MSG_SCN_REL);
2446         } else {
2447                 /* RELA */
2448                 relsize = sizeof (Rela);
2449                 rel_prefix = MSG_ORIG(MSG_SCN_RELA);
2450         }

2452         if (osp) {
2453                 size = osp->os_szoutrels;
2454                 sh_flags = osp->os_shdr->sh_flags;
2455                 if ((sectname = libld_malloc(strlen(rel_prefix) +
2456                     strlen(osp->os_name) + 1)) == 0)
2457                         return (S_ERROR);
2458                 (void) strcpy(sectname, rel_prefix);
2459                 (void) strcat(sectname, osp->os_name);
2460         } else if (ofl->ofl_flags & FLG_OF_COMREL) {
2461                 size = (ofl->ofl_reloccnt - ofl->ofl_reloccntsub) * relsize;
2462                 sh_flags = SHF_ALLOC;
2463                 sectname = (char *)MSG_ORIG(MSG_SCN_SUNWRELOC);
2464         } else {
2465                 size = ofl->ofl_relocrelsz;
2466                 sh_flags = SHF_ALLOC;
2467                 sectname = (char *)rel_prefix;
2468         }

2470         /*
2471          * Keep track of total size of 'output relocations' (to be stored
2472          * in .dynamic)
2473          */
2474         /* LINTED */
2475         ofl->ofl_relocsz += (Xword)size;

2477         if (new_section(ofl, ld_targ.t_m.m_rel_sht_type, sectname, 0, &isec,
2478             &shdr, &data) == S_ERROR)
2479                 return (S_ERROR);

2481         data->d_size = size;

2483         shdr->sh_size = (Xword)size;
2484         if (OFL_ALLOW_DYNSYM(ofl) && (sh_flags & SHF_ALLOC))
2485                 shdr->sh_flags = SHF_ALLOC;

2487         if (osp) {
2488                 /*
2489                  * The sh_info field of the SHT_REL* sections points to the
2490                  * section the relocations are to be applied to.
2491                  */
2492                 shdr->sh_flags |= SHF_INFO_LINK;
2493         }

2495         rosp = ld_place_section(ofl, isec, NULL, ld_targ.t_id.id_rel, NULL);
2496         if (rosp == (Os_desc *)S_ERROR)
2497                 return (S_ERROR);

2499         /*
2500          * Associate this relocation section to the section its going to
```

```
2501          * relocate.
2502          */
2503         if (osp) {
2504                 Aliste  idx;
2505                 Is_desc *risp;

2507                 /*
2508                  * This is used primarily so that we can update
2509                  * SHT_GROUP[sect_no] entries to point to the
2510                  * created output relocation sections.
2511                  */
2512                 for (APLIST_TRAVERSE(osp->os_relisdescs, idx, risp)) {
2513                         risp->is_osdesc = rosp;

2515                         /*
2516                          * If the input relocation section had the SHF_GROUP
2517                          * flag set - propagate it to the output relocation
2518                          * section.
2519                          */
2520                         if (risp->is_shdr->sh_flags & SHF_GROUP) {
2521                                 rosp->os_shdr->sh_flags |= SHF_GROUP;
2522                                 break;
2523                         }
2524                 }
2525                 osp->os_relosdesc = rosp;
2526         } else
2527                 ofl->ofl_osrel = rosp;

2529         /*
2530          * If this is the first relocation section we've encountered save it
2531          * so that the .dynamic entry can be initialized accordingly.
2532          */
2533         if (ofl->ofl_osrelhead == (Os_desc *)0)
2534                 ofl->ofl_osrelhead = rosp;

2536         return (1);
2537 }

2539 /*
2540  * Generate version needed section.
2541  */
2542 static uintptr_t
2543 make_verneed(Ofl_desc *ofl)
2544 {
2545         Shdr            *shdr;
2546         Elf_Data        *data;
2547         Is_desc         *isec;

2549         /*
2550          * verneed sections do not have a constant element size, so the
2551          * value of ent_cnt specified here (0) is meaningless.
2552          */
2553         if (new_section(ofl, SHT_SUNW_verneed, MSG_ORIG(MSG_SCN_SUNWVERSION),
2554             0, &isec, &shdr, &data) == S_ERROR)
2555                 return (S_ERROR);

2557         /* During version processing we calculated the total size. */
2558         data->d_size = ofl->ofl_verneedsz;
2559         shdr->sh_size = (Xword)ofl->ofl_verneedsz;

2561         ofl->ofl_osverneed =
2562             ld_place_section(ofl, isec, NULL, ld_targ.t_id.id_version, NULL);
2563         return ((uintptr_t)ofl->ofl_osverneed);
2564 }

2566 /*
```



```
2567  * Generate a version definition section.
2568  *
2569  *  o   the SHT_SUNW_verdef section defines the versions that exist within this
2570  *      image.
2571  */
2572 static uintptr_t
2573 make_verdef(Ofl_desc *ofl)
2574 {
2575         Shdr            *shdr;
2576         Elf_Data        *data;
2577         Is_desc         *isec;
2578         Ver_desc        *vdp;
2579         Str_tbl         *strtab;

2581         /*
2582          * Reserve a string table entry for the base version dependency (other
2583          * dependencies have symbol representations, which will already be
2584          * accounted for during symbol processing).
2585          */
2586         vdp = (Ver_desc *)ofl->ofl_verdesc->apl_data[0];

2588         if (OFL_IS_STATIC_OBJ(ofl))
2589                 strtab = ofl->ofl_strtab;
2590         else
2591                 strtab = ofl->ofl_dynstrtab;

2593         if (st_insert(strtab, vdp->vd_name) == -1)
2594                 return (S_ERROR);

2596         /*
2597          * verdef sections do not have a constant element size, so the
2598          * value of ent_cnt specified here (0) is meaningless.
2599          */
2600         if (new_section(ofl, SHT_SUNW_verdef, MSG_ORIG(MSG_SCN_SUNWVERSION),
2601             0, &isec, &shdr, &data) == S_ERROR)
2602                 return (S_ERROR);

2604         /* During version processing we calculated the total size. */
2605         data->d_size = ofl->ofl_verdefsz;
2606         shdr->sh_size = (Xword)ofl->ofl_verdefsz;

2608         ofl->ofl_osverdef =
2609             ld_place_section(ofl, isec, NULL, ld_targ.t_id.id_version, NULL);
2610         return ((uintptr_t)ofl->ofl_osverdef);
2611 }

2613 /*
2614  * This routine is called when -z nopartial is in effect.
2615  */
2616 uintptr_t
2617 ld_make_parexpn_data(Ofl_desc *ofl, size_t size, Xword align)
2618 {
2619         Shdr            *shdr;
2620         Elf_Data        *data;
2621         Is_desc         *isec;
2622         Os_desc         *osp;

2624         if (new_section(ofl, SHT_PROGBITS, MSG_ORIG(MSG_SCN_DATA), 0,
2625             &isec, &shdr, &data) == S_ERROR)
2626                 return (S_ERROR);

2628         shdr->sh_flags |= SHF_WRITE;
2629         data->d_size = size;
2630         shdr->sh_size = (Xword)size;
2631         if (align != 0) {
2632                 data->d_align = align;
```

```
2633                    shdr->sh_addralign = align;
2634            }

2636            if ((data->d_buf = libld_calloc(size, 1)) == NULL)
2637                    return (S_ERROR);

2639            /*
2640             * Retain handle to this .data input section. Variables using move
2641             * sections (partial initialization) will be redirected here when
2642             * such global references are added and '-z nopartial' is in effect.
2643             */
2644            ofl->ofl_isparexpn = isec;
2645            osp = ld_place_section(ofl, isec, NULL, ld_targ.t_id.id_data, NULL);
2646            if (osp == (Os_desc *)S_ERROR)
2647                    return (S_ERROR);

2649            if (!(osp->os_flags & FLG_OS_OUTREL)) {
2650                    ofl->ofl_dynshdrcnt++;
2651                    osp->os_flags |= FLG_OS_OUTREL;
2652            }
2653            return (1);
2654    }

2656    /*
2657     * Make .sunwmove section
2658     */
2659    uintptr_t
2660    ld_make_sunwmove(Ofl_desc *ofl, int mv_nums)
2661    {
2662            Shdr            *shdr;
2663            Elf_Data        *data;
2664            Is_desc         *isec;
2665            Aliste          idx;
2666            Sym_desc        *sdp;
2667            int             cnt = 1;


2670            if (new_section(ofl, SHT_SUNW_move, MSG_ORIG(MSG_SCN_SUNWMOVE),
2671                mv_nums, &isec, &shdr, &data) == S_ERROR)
2672                    return (S_ERROR);

2674            if ((data->d_buf = libld_calloc(data->d_size, 1)) == NULL)
2675                    return (S_ERROR);

2677            /*
2678             * Copy move entries
2679             */
2680            for (APLIST_TRAVERSE(ofl->ofl_parsyms, idx, sdp)) {
2681                    Aliste          idx2;
2682                    Mv_desc         *mdp;

2684                    if (sdp->sd_flags & FLG_SY_PAREXPN)
2685                            continue;

2687                    for (ALIST_TRAVERSE(sdp->sd_move, idx2, mdp))
2688                            mdp->md_oidx = cnt++;
2689            }

2691            if ((ofl->ofl_osmove = ld_place_section(ofl, isec, NULL, 0, NULL)) ==
2692                (Os_desc *)S_ERROR)
2693                    return (S_ERROR);

2695            return (1);
2696    }

2698    /*
```



```
2699     * Given a relocation descriptor that references a string table
2700     * input section, locate the string referenced and return a pointer
2701     * to it.
2702     */
2703    static const char *
2704    strmerge_get_reloc_str(Ofl_desc *ofl, Rel_desc *rsp)
2705    {
2706            Sym_desc *sdp = rsp->rel_sym;
2707            Xword    str_off;

2709            /*
2710             * In the case of an STT_SECTION symbol, the addend of the
2711             * relocation gives the offset into the string section. For
2712             * other symbol types, the symbol value is the offset.
2713             */

2715            if (ELF_ST_TYPE(sdp->sd_sym->st_info) != STT_SECTION) {
2716                    str_off = sdp->sd_sym->st_value;
2717            } else if ((rsp->rel_flags & FLG_REL_RELA) == FLG_REL_RELA) {
2718                    /*
2719                     * For SHT_RELA, the addend value is found in the
2720                     * rel_raddend field of the relocation.
2721                     */
2722                    str_off = rsp->rel_raddend;
2723            } else {        /* REL and STT_SECTION */
2724                    /*
2725                     * For SHT_REL, the "addend" is not part of the relocation
2726                     * record. Instead, it is found at the relocation target
2727                     * address.
2728                     */
2729                    uchar_t *addr = (uchar_t *)((uintptr_t)rsp->rel_roffset +
2730                        (uintptr_t)rsp->rel_isdesc->is_indata->d_buf);

2732                    if (ld_reloc_targval_get(ofl, rsp, addr, &str_off) == 0)
2733                            return (0);
2734            }

2736            return (str_off + (char *)sdp->sd_isc->is_indata->d_buf);
2737    }

2739    /*
2740     * First pass over the relocation records for string table merging.
2741     * Build lists of relocations and symbols that will need modification,
2742     * and insert the strings they reference into the mstrtab string table.
2743     *
2744     * entry:
2745     *      ofl, osp - As passed to ld_make_strmerge().
2746     *      mstrtab - String table to receive input strings. This table
2747     *              must be in its first (initialization) pass and not
2748     *              yet cooked (st_getstrtab_sz() not yet called).
2749     *      rel_alpp - APlist to receive pointer to any relocation
2750     *              descriptors with STT_SECTION symbols that reference
2751     *              one of the input sections being merged.
2752     *      sym_alpp - APlist to receive pointer to any symbols that reference
2753     *              one of the input sections being merged.
2754     *      rcp - Pointer to cache of relocation descriptors to examine.
2755     *              Either &ofl->ofl_actrels (active relocations)
2756     *              or &ofl->ofl_outrels (output relocations).
2757     *
2758     * exit:
2759     *      On success, rel_alpp and sym_alpp are updated, and
2760     *      any strings in the mergeable input sections referenced by
2761     *      a relocation has been entered into mstrtab. True (1) is returned.
2762     *
2763     *      On failure, False (0) is returned.
2764     */
```

```
2765 static int
2766 strmerge_pass1(Ofl_desc *ofl, Os_desc *osp, Str_tbl *mstrtab,
2767     APlist **rel_alpp, APlist **sym_alpp, Rel_cache *rcp)
2768 {
2769         Aliste          idx;
2770         Rel_cachebuf    *rcbp;
2771         Sym_desc        *sdp;
2772         Sym_desc        *last_sdp = NULL;
2773         Rel_desc        *rsp;
2774         const char      *name;
2775
2776         REL_CACHE_TRAVERSE(rcp, idx, rcbp, rsp) {
2777                 sdp = rsp->rel_sym;
2778                 if ((sdp->sd_isc == NULL) || ((sdp->sd_isc->is_flags &
2779                     (FLG_IS_DISCARD | FLG_IS_INSTRMRG)) != FLG_IS_INSTRMRG) ||
2780                     (sdp->sd_isc->is_osdesc != osp))
2781                         continue;
2782
2783                 /*
2784                  * Remember symbol for use in the third pass. There is no
2785                  * reason to save a given symbol more than once, so we take
2786                  * advantage of the fact that relocations to a given symbol
2787                  * tend to cluster in the list. If this is the same symbol
2788                  * we saved last time, don't bother.
2789                  */
2790                 if (last_sdp != sdp) {
2791                         if (aplist_append(sym_alpp, sdp, AL_CNT_STRMRGSYM) ==
2792                             NULL)
2793                                 return (0);
2794                         last_sdp = sdp;
2795                 }
2796
2797                 /* Enter the string into our new string table */
2798                 name = strmerge_get_reloc_str(ofl, rsp);
2799                 if (st_insert(mstrtab, name) == -1)
2800                         return (0);
2801
2802                 /*
2803                  * If this is an STT_SECTION symbol, then the second pass
2804                  * will need to modify this relocation, so hang on to it.
2805                  */
2806                 if ((ELF_ST_TYPE(sdp->sd_sym->st_info) == STT_SECTION) &&
2807                     (aplist_append(rel_alpp, rsp, AL_CNT_STRMRGREL) == NULL))
2808                         return (0);
2809         }
2810
2811         return (1);
2812 }
2813
2814 /*
2815  * If the output section has any SHF_MERGE|SHF_STRINGS input sections,
2816  * replace them with a single merged/compressed input section.
2817  *
2818  * entry:
2819  *      ofl - Output file descriptor
2820  *      osp - Output section descriptor
2821  *      rel_alpp, sym_alpp, - Address of 2 APlists, to be used
2822  *              for internal processing. On the initial call to
2823  *              ld_make_strmerge, these list pointers must be NULL.
2824  *              The caller is encouraged to pass the same lists back for
2825  *              successive calls to this function without freeing
2826  *              them in between calls. This causes a single pair of
2827  *              memory allocations to be reused multiple times.
2828  *
2829  * exit:
2830  *      If section merging is possible, it is done. If no errors are
```



```
2831  *      encountered, True (1) is returned. On error, S_ERROR.
2832  *
2833  *      The contents of rel_alpp and sym_alpp on exit are
2834  *      undefined. The caller can free them, or pass them back to a subsequent
2835  *      call to this routine, but should not examine their contents.
2836  */
2837 static uintptr_t
2838 ld_make_strmerge(Ofl_desc *ofl, Os_desc *osp, APlist **rel_alpp,
2839     APlist **sym_alpp)
2840 {
2841         Str_tbl         *mstrtab;       /* string table for string merge secs */
2842         Is_desc         *mstrsec;       /* Generated string merge section */
2843         Is_desc         *isp;
2844         Shdr            *mstr_shdr;
2845         Elf_Data        *mstr_data;
2846         Sym_desc        *sdp;
2847         Rel_desc        *rsp;
2848         Aliste          idx;
2849         size_t          data_size;
2850         int             st_setstring_status;
2851         size_t          stoff;
2852
2853         /* If string table compression is disabled, there's nothing to do */
2854         if ((ofl->ofl_flags1 & FLG_OF1_NCSTTAB) != 0)
2855                 return (1);
2856
2857         /*
2858          * Pass over the mergeable input sections, and if they haven't
2859          * all been discarded, create a string table.
2860          */
2861         mstrtab = NULL;
2862         for (APLIST_TRAVERSE(osp->os_mstrisdescs, idx, isp)) {
2863                 if (isdesc_discarded(isp))
2864                         continue;
2865
2866                 /*
2867                  * Input sections of 0 size are dubiously valid since they do
2868                  * not even contain the NUL string.  Ignore them.
2869                  */
2870                 if (isp->is_shdr->sh_size == 0)
2871                         continue;
2872
2873                 /*
2874                  * We have at least one non-discarded section.
2875                  * Create a string table descriptor.
2876                  */
2877                 if ((mstrtab = st_new(FLG_STNEW_COMPRESS)) == NULL)
2878                         return (S_ERROR);
2879                 break;
2880         }
2881
2882         /* If no string table was created, we have no mergeable sections */
2883         if (mstrtab == NULL)
2884                 return (1);
2885
2886         /*
2887          * This routine has to make 3 passes:
2888          *
2889          *      1) Examine all relocations, insert strings from relocations
2890          *              to the mergeable input sections into the string table.
2891          *      2) Modify the relocation values to be correct for the
2892          *              new merged section.
2893          *      3) Modify the symbols used by the relocations to reference
2894          *              the new section.
2895          *
2896          * These passes cannot be combined:
```

```
2897          *      - The string table code works in two passes, and all
2898          *              strings have to be loaded in pass one before the
2899          *              offset of any strings can be determined.
2900          *      - Multiple relocations reference a single symbol, so the
2901          *              symbol cannot be modified until all relocations are
2902          *              fixed.
2903          *
2904          * The number of relocations related to section merging is usually
2905          * a mere fraction of the overall active and output relocation lists,
2906          * and the number of symbols is usually a fraction of the number
2907          * of related relocations. We therefore build APlists for the
2908          * relocations and symbols in the first pass, and then use those
2909          * lists to accelerate the operation of pass 2 and 3.
2910          *
2911          * Reinitialize the lists to a completely empty state.
2912          */
2913         aplist_reset(*rel_alpp);
2914         aplist_reset(*sym_alpp);
2915
2916         /*
2917          * Pass 1:
2918          *
2919          * Every relocation related to this output section (and the input
2920          * sections that make it up) is found in either the active, or the
2921          * output relocation list, depending on whether the relocation is to
2922          * be processed by this invocation of the linker, or inserted into the
2923          * output object.
2924          *
2925          * Build lists of relocations and symbols that will need modification,
2926          * and insert the strings they reference into the mstrtab string table.
2927          */
2928         if (strmerge_pass1(ofl, osp, mstrtab, rel_alpp, sym_alpp,
2929             &ofl->ofl_actrels) == 0)
2930                 goto return_s_error;
2931         if (strmerge_pass1(ofl, osp, mstrtab, rel_alpp, sym_alpp,
2932             &ofl->ofl_outrels) == 0)
2933                 goto return_s_error;
2934
2935         /*
2936          * Get the size of the new input section. Requesting the
2937          * string table size "cooks" the table, and finalizes its contents.
2938          */
2939         data_size = st_getstrtab_sz(mstrtab);
2940
2941         /* Create a new input section to hold the merged strings */
2942         if (new_section_from_template(ofl, isp, data_size,
2943             &mstrsec, &mstr_shdr, &mstr_data) == S_ERROR)
2944                 goto return_s_error;
2945         mstrsec->is_flags |= FLG_IS_GNSTRMRG;
2946
2947         /*
2948          * Allocate a data buffer for the new input section.
2949          * Then, associate the buffer with the string table descriptor.
2950          */
2951         if ((mstr_data->d_buf = libld_malloc(data_size)) == NULL)
2952                 goto return_s_error;
2953         if (st_setstrbuf(mstrtab, mstr_data->d_buf, data_size) == -1)
2954                 goto return_s_error;
2955
2956         /* Add the new section to the output image */
2957         if (ld_place_section(ofl, mstrsec, NULL, osp->os_identndx, NULL) ==
2958             (Os_desc *)S_ERROR)
2959                 goto return_s_error;
2960
2961         /*
2962          * Pass 2:
```



```
2963          *
2964          * Revisit the relocation descriptors with STT_SECTION symbols
2965          * that were saved by the first pass. Update each relocation
2966          * record so that the offset it contains is for the new section
2967          * instead of the original.
2968          */
2969         for (APLIST_TRAVERSE(*rel_alpp, idx, rsp)) {
2970                 const char      *name;
2971
2972                 /* Put the string into the merged string table */
2973                 name = strmerge_get_reloc_str(ofl, rsp);
2974                 st_setstring_status = st_setstring(mstrtab, name, &stoff);
2975                 if (st_setstring_status == -1) {
2976                         /*
2977                          * A failure to insert at this point means that
2978                          * something is corrupt. This isn't a resource issue.
2979                          */
2980                         assert(st_setstring_status != -1);
2981                         goto return_s_error;
2982                 }
2983
2984                 /*
2985                  * Alter the relocation to access the string at the
2986                  * new offset in our new string table.
2987                  *
2988                  * For SHT_RELA platforms, it suffices to simply
2989                  * update the rel_raddend field of the relocation.
2990                  *
2991                  * For SHT_REL platforms, the new "addend" value
2992                  * needs to be written at the address being relocated.
2993                  * However, we can't alter the input sections which
2994                  * are mapped readonly, and the output image has not
2995                  * been created yet. So, we defer this operation,
2996                  * using the rel_raddend field of the relocation
2997                  * which is normally 0 on a REL platform, to pass the
2998                  * new "addend" value to ld_perform_outreloc() or
2999                  * ld_do_activerelocs(). The FLG_REL_NADDEND flag
3000                  * tells them that this is the case.
3001                  */
3002                 if ((rsp->rel_flags & FLG_REL_RELA) == 0)   /* REL */
3003                         rsp->rel_flags |= FLG_REL_NADDEND;
3004                 rsp->rel_raddend = (Sxword)stoff;
3005
3006                 /*
3007                  * Generate a symbol name string for STT_SECTION symbols
3008                  * that might reference our merged section. This shows up
3009                  * in debug output and helps show how the relocation has
3010                  * changed from its original input section to our merged one.
3011                  */
3012                 if (ld_stt_section_sym_name(mstrsec) == NULL)
3013                         goto return_s_error;
3014         }
3015
3016         /*
3017          * Pass 3:
3018          *
3019          * Modify the symbols referenced by the relocation descriptors
3020          * so that they reference the new input section containing the
3021          * merged strings instead of the original input sections.
3022          */
3023         for (APLIST_TRAVERSE(*sym_alpp, idx, sdp)) {
3024                 /*
3025                  * If we've already processed this symbol, don't do it
3026                  * twice. strmerge_pass1() uses a heuristic (relocations to
3027                  * the same symbol clump together) to avoid inserting a
3028                  * given symbol more than once, but repeat symbols in
```

```
3029                 * the list can occur.
3030                 */
3031                if ((sdp->sd_isc->is_flags & FLG_IS_INSTRMRG) == 0)
3032                        continue;
3033
3034                if (ELF_ST_TYPE(sdp->sd_sym->st_info) != STT_SECTION) {
3035                        /*
3036                         * This is not an STT_SECTION symbol, so its
3037                         * value is the offset of the string within the
3038                         * input section. Update the address to reflect
3039                         * the address in our new merged section.
3040                         */
3041                        const char *name = sdp->sd_sym->st_value +
3042                            (char *)sdp->sd_isc->is_indata->d_buf;
3043
3044                        st_setstring_status =
3045                            st_setstring(mstrtab, name, &stoff);
3046                        if (st_setstring_status == -1) {
3047                                /*
3048                                 * A failure to insert at this point means
3049                                 * something is corrupt. This isn't a
3050                                 * resource issue.
3051                                 */
3052                                assert(st_setstring_status != -1);
3053                                goto return_s_error;
3054                        }
3055
3056                        if (ld_sym_copy(sdp) == S_ERROR)
3057                                goto return_s_error;
3058                        sdp->sd_sym->st_value = (Word)stoff;
3059                }
3060
3061                /* Redirect the symbol to our new merged section */
3062                sdp->sd_isc = mstrsec;
3063        }
3064
3065        /*
3066         * There are no references left to the original input string sections.
3067         * Mark them as discarded so they don't go into the output image.
3068         * At the same time, add up the sizes of the replaced sections.
3069         */
3070        data_size = 0;
3071        for (APLIST_TRAVERSE(osp->os_mstrisdescs, idx, isp)) {
3072                if (isp->is_flags & (FLG_IS_DISCARD | FLG_IS_GNSTRMRG))
3073                        continue;
3074
3075                data_size += isp->is_indata->d_size;
3076
3077                isp->is_flags |= FLG_IS_DISCARD;
3078                DBG_CALL(Dbg_sec_discarded(ofl->ofl_lml, isp, mstrsec));
3079        }
3080
3081        /* Report how much space we saved in the output section */
3082        DBG_CALL(Dbg_sec_genstr_compress(ofl->ofl_lml, osp->os_name, data_size,
3083            mstr_data->d_size));
3084
3085        st_destroy(mstrtab);
3086        return (1);
3087
3088 return_s_error:
3089        st_destroy(mstrtab);
3090        return (S_ERROR);
3091 }
3092
3093 /*
3094  * Update a data buffers size.  A number of sections have to be created, and
```



```
3095  * the sections header contributes to the size of the eventual section.  Thus,
3096  * a section may be created, and once all associated sections have been created,
3097  * we return to establish the required section size.
3098  */
3099 inline static void
3100 update_data_size(Os_desc *osp, ulong_t cnt)
3101 {
3102        Is_desc         *isec = ld_os_first_isdesc(osp);
3103        Elf_Data        *data = isec->is_indata;
3104        Shdr            *shdr = osp->os_shdr;
3105        size_t          size = cnt * shdr->sh_entsize;
3106
3107        shdr->sh_size = (Xword)size;
3108        data->d_size = size;
3109 }
3110
3111 /*
3112  * The following sections are built after all input file processing and symbol
3113  * validation has been carried out.  The order is important (because the
3114  * addition of a section adds a new symbol there is a chicken and egg problem
3115  * of maintaining the appropriate counts).  By maintaining a known order the
3116  * individual routines can compensate for later, known, additions.
3117  */
3118 uintptr_t
3119 ld_make_sections(Ofl_desc *ofl)
3120 {
3121        ofl_flag_t      flags = ofl->ofl_flags;
3122        Sg_desc         *sgp;
3123
3124        /*
3125         * Generate any special sections.
3126         */
3127        if (flags & FLG_OF_ADDVERS)
3128                if (make_comment(ofl) == S_ERROR)
3129                        return (S_ERROR);
3130
3131        if (make_interp(ofl) == S_ERROR)
3132                return (S_ERROR);
3133
3134        /*
3135         * Create a capabilities section if required.
3136         */
3137        if (make_cap(ofl, SHT_SUNW_cap, MSG_ORIG(MSG_SCN_SUNWCAP),
3138            ld_targ.t_id.id_cap) == S_ERROR)
3139                return (S_ERROR);
3140
3141        /*
3142         * Create any init/fini array sections.
3143         */
3144        if (make_array(ofl, SHT_INIT_ARRAY, MSG_ORIG(MSG_SCN_INITARRAY),
3145            ofl->ofl_initarray) == S_ERROR)
3146                return (S_ERROR);
3147
3148        if (make_array(ofl, SHT_FINI_ARRAY, MSG_ORIG(MSG_SCN_FINIARRAY),
3149            ofl->ofl_finiarray) == S_ERROR)
3150                return (S_ERROR);
3151
3152        if (make_array(ofl, SHT_PREINIT_ARRAY, MSG_ORIG(MSG_SCN_PREINITARRAY),
3153            ofl->ofl_preiarray) == S_ERROR)
3154                return (S_ERROR);
3155
3156        /*
3157         * Make the .plt section.  This occurs after any other relocation
3158         * sections are generated (see reloc_init()) to ensure that the
3159         * associated relocation section is after all the other relocation
3160         * sections.
```

```
3161          */
3162         if ((ofl->ofl_pltcnt) || (ofl->ofl_pltpad))
3163                 if (make_plt(ofl) == S_ERROR)
3164                         return (S_ERROR);
3165
3166         /*
3167          * Determine whether any sections or files are not referenced.  Under
3168          * -Dunused a diagnostic for any unused components is generated, under
3169          * -zignore the component is removed from the final output.
3170          */
3171         if (DBG_ENABLED || (ofl->ofl_flags1 & FLG_OF1_IGNPRC)) {
3172                 if (ignore_section_processing(ofl) == S_ERROR)
3173                         return (S_ERROR);
3174         }
3175
3176         /*
3177          * If we have detected a situation in which previously placed
3178          * output sections may have been discarded, perform the necessary
3179          * readjustment.
3180          */
3181         if (ofl->ofl_flags & FLG_OF_ADJOSCNT)
3182                 adjust_os_count(ofl);
3183
3184         /*
3185          * Do any of the output sections contain input sections that
3186          * are candidates for string table merging? For each such case,
3187          * we create a replacement section, insert it, and discard the
3188          * originals.
3189          *
3190          * rel_alpp and sym_alpp are used by ld_make_strmerge()
3191          * for its internal processing. We are responsible for the
3192          * initialization and cleanup, and ld_make_strmerge() handles the rest.
3193          * This allows us to reuse a single pair of memory buffers, allocated
3194          * for this processing, for all the output sections.
3195          */
3196         if ((ofl->ofl_flags1 & FLG_OF1_NCSTTAB) == 0) {
3197                 int     error_seen = 0;
3198                 APlist  *rel_alpp = NULL;
3199                 APlist  *sym_alpp = NULL;
3200                 Aliste  idx1;
3201
3202                 for (APLIST_TRAVERSE(ofl->ofl_segs, idx1, sgp)) {
3203                         Os_desc *osp;
3204                         Aliste  idx2;
3205
3206                         for (APLIST_TRAVERSE(sgp->sg_osdescs, idx2, osp))
3207                                 if ((osp->os_mstrisdescs != NULL) &&
3208                                     (ld_make_strmerge(ofl, osp,
3209                                     &rel_alpp, &sym_alpp) ==
3210                                     S_ERROR)) {
3211                                         error_seen = 1;
3212                                         break;
3213                                 }
3214                 }
3215                 if (rel_alpp != NULL)
3216                         libld_free(rel_alpp);
3217                 if (sym_alpp != NULL)
3218                         libld_free(sym_alpp);
3219                 if (error_seen != 0)
3220                         return (S_ERROR);
3221         }
3222
3223         /*
3224          * Add any necessary versioning information.
3225          */
3226         if (!(flags & FLG_OF_NOVERSEC)) {
```



```
3227                 if ((flags & FLG_OF_VERNEED) &&
3228                     (make_verneed(ofl) == S_ERROR))
3229                         return (S_ERROR);
3230                 if ((flags & FLG_OF_VERDEF) &&
3231                     (make_verdef(ofl) == S_ERROR))
3232                         return (S_ERROR);
3233                 if ((flags & (FLG_OF_VERNEED | FLG_OF_VERDEF)) &&
3234                     ((ofl->ofl_osversym = make_sym_sec(ofl,
3235                     MSG_ORIG(MSG_SCN_SUNWVERSYM), SHT_SUNW_versym,
3236                     ld_targ.t_id.id_version)) == (Os_desc*)S_ERROR))
3237                         return (S_ERROR);
3238         }
3239
3240         /*
3241          * Create a syminfo section if necessary.
3242          */
3243         if (flags & FLG_OF_SYMINFO) {
3244                 if ((ofl->ofl_ossyminfo = make_sym_sec(ofl,
3245                     MSG_ORIG(MSG_SCN_SUNWSYMINFO), SHT_SUNW_syminfo,
3246                     ld_targ.t_id.id_syminfo)) == (Os_desc *)S_ERROR)
3247                         return (S_ERROR);
3248         }
3249
3250         if (flags & FLG_OF_COMREL) {
3251                 /*
3252                  * If -zcombreloc is enabled then all relocations (except for
3253                  * the PLT's) are coalesced into a single relocation section.
3254                  */
3255                 if (ofl->ofl_reloccnt) {
3256                         if (make_reloc(ofl, NULL) == S_ERROR)
3257                                 return (S_ERROR);
3258                 }
3259         } else {
3260                 Aliste  idx1;
3261
3262                 /*
3263                  * Create the required output relocation sections.  Note, new
3264                  * sections may be added to the section list that is being
3265                  * traversed.  These insertions can move the elements of the
3266                  * Alist such that a section descriptor is re-read.  Recursion
3267                  * is prevented by maintaining a previous section pointer and
3268                  * insuring that this pointer isn't re-examined.
3269                  */
3270                 for (APLIST_TRAVERSE(ofl->ofl_segs, idx1, sgp)) {
3271                         Os_desc *osp, *posp = 0;
3272                         Aliste  idx2;
3273
3274                         for (APLIST_TRAVERSE(sgp->sg_osdescs, idx2, osp)) {
3275                                 if ((osp != posp) && osp->os_szoutrels &&
3276                                     (osp != ofl->ofl_osplt)) {
3277                                         if (make_reloc(ofl, osp) == S_ERROR)
3278                                                 return (S_ERROR);
3279                                 }
3280                                 posp = osp;
3281                         }
3282                 }
3283
3284                 /*
3285                  * If we're not building a combined relocation section, then
3286                  * build a .rel[a] section as required.
3287                  */
3288                 if (ofl->ofl_relocrelsz) {
3289                         if (make_reloc(ofl, NULL) == S_ERROR)
3290                                 return (S_ERROR);
3291                 }
3292         }
```

```
3294         /*
3295          * The PLT relocations are always in their own section, and we try to
3296          * keep them at the end of the PLT table.  We do this to keep the hot
3297          * "data" PLT's at the head of the table nearer the .dynsym & .hash.
3298          */
3299         if (ofl->ofl_osplt && ofl->ofl_relocpltsz) {
3300                 if (make_reloc(ofl, ofl->ofl_osplt) == S_ERROR)
3301                         return (S_ERROR);
3302         }
3303
3304         /*
3305          * Finally build the symbol and section header sections.
3306          */
3307         if (flags & FLG_OF_DYNAMIC) {
3308                 if (make_dynamic(ofl) == S_ERROR)
3309                         return (S_ERROR);
3310
3311                 /*
3312                  * A number of sections aren't necessary within a relocatable
3313                  * object, even if -dy has been used.
3314                  */
3315                 if (!(flags & FLG_OF_RELOBJ)) {
3316                         if (make_hash(ofl) == S_ERROR)
3317                                 return (S_ERROR);
3318                         if (make_dynstr(ofl) == S_ERROR)
3319                                 return (S_ERROR);
3320                         if (make_dynsym(ofl) == S_ERROR)
3321                                 return (S_ERROR);
3322                         if (ld_unwind_make_hdr(ofl) == S_ERROR)
3323                                 return (S_ERROR);
3324                         if (make_dynsort(ofl) == S_ERROR)
3325                                 return (S_ERROR);
3326                 }
3327         }
3328
3329         if (!(flags & FLG_OF_STRIP) || (flags & FLG_OF_RELOBJ) ||
3330             ((flags & FLG_OF_STATIC) && ofl->ofl_osversym)) {
3331                 /*
3332                  * Do we need to make a SHT_SYMTAB_SHNDX section
3333                  * for the dynsym.  If so - do it now.
3334                  */
3335                 if (ofl->ofl_osdynsym &&
3336                     ((ofl->ofl_shdrcnt + 3) >= SHN_LORESERVE)) {
3337                         if (make_dynsym_shndx(ofl) == S_ERROR)
3338                                 return (S_ERROR);
3339                 }
3340
3341                 if (make_strtab(ofl) == S_ERROR)
3342                         return (S_ERROR);
3343                 if (make_symtab(ofl) == S_ERROR)
3344                         return (S_ERROR);
3345         } else {
3346                 /*
3347                  * Do we need to make a SHT_SYMTAB_SHNDX section
3348                  * for the dynsym.  If so - do it now.
3349                  */
3350                 if (ofl->ofl_osdynsym &&
3351                     ((ofl->ofl_shdrcnt + 1) >= SHN_LORESERVE)) {
3352                         if (make_dynsym_shndx(ofl) == S_ERROR)
3353                                 return (S_ERROR);
3354                 }
3355         }
3356
3357         if (make_shstrtab(ofl) == S_ERROR)
3358                 return (S_ERROR);
```



```
3360         /*
3361          * Now that we've created all output sections, adjust the size of the
3362          * SHT_SUNW_versym and SHT_SUNW_syminfo section, which are dependent on
3363          * the associated symbol table sizes.
3364          */
3365         if (ofl->ofl_osversym || ofl->ofl_ossyminfo) {
3366                 ulong_t         cnt;
3367                 Is_desc         *isp;
3368                 Os_desc         *osp;
3369
3370                 if (OFL_IS_STATIC_OBJ(ofl))
3371                         osp = ofl->ofl_ossymtab;
3372                 else
3373                         osp = ofl->ofl_osdynsym;
3374
3375                 isp = ld_os_first_isdesc(osp);
3376                 cnt = (isp->is_shdr->sh_size / isp->is_shdr->sh_entsize);
3377
3378                 if (ofl->ofl_osversym)
3379                         update_data_size(ofl->ofl_osversym, cnt);
3380
3381                 if (ofl->ofl_ossyminfo)
3382                         update_data_size(ofl->ofl_ossyminfo, cnt);
3383         }
3384
3385         /*
3386          * Now that we've created all output sections, adjust the size of the
3387          * SHT_SUNW_capinfo, which is dependent on the associated symbol table
3388          * size.
3389          */
3390         if (ofl->ofl_oscapinfo) {
3391                 ulong_t cnt;
3392
3393                 /*
3394                  * Symbol capabilities symbols are placed directly after the
3395                  * STT_FILE symbol, section symbols, and any register symbols.
3396                  * Effectively these are the first of any series of demoted
3397                  * (scoped) symbols.
3398                  */
3399                 if (OFL_IS_STATIC_OBJ(ofl))
3400                         cnt = SYMTAB_ALL_CNT(ofl);
3401                 else
3402                         cnt = DYNSYM_ALL_CNT(ofl);
3403
3404                 update_data_size(ofl->ofl_oscapinfo, cnt);
3405         }
3406         return (1);
3407 }
3408
3409 /*
3410  * Build an additional data section - used to back OBJT symbol definitions
3411  * added with a mapfile.
3412  */
3413 Is_desc *
3414 ld_make_data(Ofl_desc *ofl, size_t size)
3415 {
3416         Shdr            *shdr;
3417         Elf_Data        *data;
3418         Is_desc         *isec;
3419
3420         if (new_section(ofl, SHT_PROGBITS, MSG_ORIG(MSG_SCN_DATA), 0,
3421             &isec, &shdr, &data) == S_ERROR)
3422                 return ((Is_desc *)S_ERROR);
3423
3424         data->d_size = size;
```

```
3425         shdr->sh_size = (Xword)size;
3426         shdr->sh_flags |= SHF_WRITE;

3428         if (aplist_append(&ofl->ofl_mapdata, isec, AL_CNT_OFL_MAPSECS) == NULL)
3429                 return ((Is_desc *)S_ERROR);

3431         return (isec);
3432 }

3434 /*
3435  * Build an additional text section - used to back FUNC symbol definitions
3436  * added with a mapfile.
3437  */
3438 Is_desc *
3439 ld_make_text(Ofl_desc *ofl, size_t size)
3440 {
3441         Shdr            *shdr;
3442         Elf_Data        *data;
3443         Is_desc         *isec;

3445         /*
3446          * Insure the size is sufficient to contain the minimum return
3447          * instruction.
3448          */
3449         if (size < ld_targ.t_nf.nf_size)
3450                 size = ld_targ.t_nf.nf_size;

3452         if (new_section(ofl, SHT_PROGBITS, MSG_ORIG(MSG_SCN_TEXT), 0,
3453             &isec, &shdr, &data) == S_ERROR)
3454                 return ((Is_desc *)S_ERROR);

3456         data->d_size = size;
3457         shdr->sh_size = (Xword)size;
3458         shdr->sh_flags |= SHF_EXECINSTR;

3460         /*
3461          * Fill the buffer with the appropriate return instruction.
3462          * Note that there is no need to swap bytes on a non-native,
3463          * link, as the data being copied is given in bytes.
3464          */
3465         if ((data->d_buf = libld_calloc(size, 1)) == NULL)
3466                 return ((Is_desc *)S_ERROR);
3467         (void) memcpy(data->d_buf, ld_targ.t_nf.nf_template,
3468             ld_targ.t_nf.nf_size);

3470         /*
3471          * If size was larger than required, and the target supplies
3472          * a fill function, use it to fill the balance. If there is no
3473          * fill function, we accept the 0-fill supplied by libld_calloc().
3474          */
3475         if ((ld_targ.t_ff.ff_execfill != NULL) && (size > ld_targ.t_nf.nf_size))
3476                 ld_targ.t_ff.ff_execfill(data->d_buf, ld_targ.t_nf.nf_size,
3477                     size - ld_targ.t_nf.nf_size);

3479         if (aplist_append(&ofl->ofl_maptext, isec, AL_CNT_OFL_MAPSECS) == NULL)
3480                 return ((Is_desc *)S_ERROR);

3482         return (isec);
3483 }

3485 void
3486 ld_comdat_validate(Ofl_desc *ofl, Ifl_desc *ifl)
3487 {
3488         int i;

3490         for (i = 0; i < ifl->ifl_shnum; i++) {
```



```
3491                 Is_desc *isp = ifl->ifl_isdesc[i];
3492                 int types = 0;
3493                 char buf[1024] = "";
3494                 Group_desc *gr = NULL;

3496                 if ((isp == NULL) || (isp->is_flags & FLG_IS_COMDAT) == 0)
3497                         continue;

3499                 if (isp->is_shdr->sh_type == SHT_SUNW_COMDAT) {
3500                         types++;
3501                         (void) strlcpy(buf, MSG_ORIG(MSG_STR_SUNW_COMDAT),
3502                             sizeof (buf));
3503                 }

3505                 if (strncmp(MSG_ORIG(MSG_SCN_GNU_LINKONCE), isp->is_name,
3506                     MSG_SCN_GNU_LINKONCE_SIZE) == 0) {
3507                         types++;
3508                         if (types > 1)
3509                                 (void) strlcat(buf, ", ", sizeof (buf));
3510                         (void) strlcat(buf, MSG_ORIG(MSG_SCN_GNU_LINKONCE),
3511                             sizeof (buf));
3512                 }

3514                 if ((isp->is_shdr->sh_flags & SHF_GROUP) &&
3515                     ((gr = ld_get_group(ofl, isp)) != NULL) &&
3516                     (gr->gd_data[0] & GRP_COMDAT)) {
3517                         types++;
3518                         if (types > 1)
3519                                 (void) strlcat(buf, ", ", sizeof (buf));
3520                         (void) strlcat(buf, MSG_ORIG(MSG_STR_GROUP),
3521                             sizeof (buf));
3522                 }

3524                 if (types > 1)
3525                         ld_eprintf(ofl, ERR_FATAL,
3526                             MSG_INTL(MSG_SCN_MULTICOMDAT), ifl->ifl_name,
3527                             EC_WORD(isp->is_scnndx), isp->is_name, buf);
3528         }
3529 }
```

```
**********************************************************
  118329 Wed May 27 19:49:05 2015
new/usr/src/cmd/sgs/libld/common/update.c
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
______unchanged_portion_omitted_

2058 /*
2059  * Build the dynamic section.
2060  *
2061  * This routine must be maintained in parallel with make_dynamic()
2062  * in sections.c
2063  */
2064 static int
2065 update_odynamic(Ofl_desc *ofl)
2066 {
2067         Aliste          idx;
2068         Ifl_desc        *ifl;
2069         Sym_desc        *sdp;
2070         Shdr            *shdr;
2071         Dyn             *_dyn = (Dyn *)ofl->ofl_osdynamic->os_outdata->d_buf;
2072         Dyn             *dyn;
2073         Os_desc         *symosp, *strosp;
2074         Str_tbl         *strtbl;
2075         size_t          stoff;
2076         ofl_flag_t      flags = ofl->ofl_flags;
2077         int             not_relobj = !(flags & FLG_OF_RELOBJ);
2078         Word            cnt;

2080         /*
2081          * Relocatable objects can be built with -r and -dy to trigger the
2082          * creation of a .dynamic section.  This model is used to create kernel
2083          * device drivers.  The .dynamic section provides a subset of userland
2084          * .dynamic entries, typically entries such as DT_NEEDED and DT_RUNPATH.
2085          *
2086          * Within a dynamic object, any .dynamic string references are to the
2087          * .dynstr table.  Within a relocatable object, these strings can reside
2088          * within the .strtab.
2089          */
2090         if (OFL_IS_STATIC_OBJ(ofl)) {
2091                 symosp = ofl->ofl_ossymtab;
2092                 strosp = ofl->ofl_osstrtab;
2093                 strtbl = ofl->ofl_strtab;
2094         } else {
2095                 symosp = ofl->ofl_osdynsym;
2096                 strosp = ofl->ofl_osdynstr;
2097                 strtbl = ofl->ofl_dynstrtab;
2098         }

2100         /* LINTED */
2101         ofl->ofl_osdynamic->os_shdr->sh_link = (Word)elf_ndxscn(strosp->os_scn);

2103         dyn = _dyn;

2105         for (APLIST_TRAVERSE(ofl->ofl_sos, idx, ifl)) {
2106                 if ((ifl->ifl_flags &
2107                     (FLG_IF_IGNORE | FLG_IF_DEPREQD)) == FLG_IF_IGNORE)
2108                         continue;

2110                 /*
2111                  * Create and set up the DT_POSFLAG_1 entry here if required.
2112                  */
2113                 if ((ifl->ifl_flags & MSK_IF_POSFLAG1) &&
```



```
2114                     (ifl->ifl_flags & FLG_IF_NEEDED) && not_relobj) {
2115                         dyn->d_tag = DT_POSFLAG_1;
2116                         if (ifl->ifl_flags & FLG_IF_LAZYLD)
2117                                 dyn->d_un.d_val = DF_P1_LAZYLOAD;
2118                         if (ifl->ifl_flags & FLG_IF_GRPPRM)
2119                                 dyn->d_un.d_val |= DF_P1_GROUPPERM;
2120                         if (ifl->ifl_flags & FLG_IF_DEFERRED)
2121                                 dyn->d_un.d_val |= DF_P1_DEFERRED;
2122                         dyn++;
2123                 }

2125                 if (ifl->ifl_flags & (FLG_IF_NEEDED | FLG_IF_NEEDSTR))
2126                         dyn->d_tag = DT_NEEDED;
2127                 else
2128                         continue;

2130                 (void) st_setstring(strtbl, ifl->ifl_soname, &stoff);
2131                 dyn->d_un.d_val = stoff;
2132                 /* LINTED */
2133                 ifl->ifl_neededndx = (Half)(((uintptr_t)dyn - (uintptr_t)_dyn) /
2134                     sizeof (Dyn));
2135                 dyn++;
2136         }

2138         if (not_relobj) {
2139                 if (ofl->ofl_dtsfltrs != NULL) {
2140                         Dfltr_desc      *dftp;

2142                         for (ALIST_TRAVERSE(ofl->ofl_dtsfltrs, idx, dftp)) {
2143                                 if (dftp->dft_flag == FLG_SY_AUXFLTR)
2144                                         dyn->d_tag = DT_SUNW_AUXILIARY;
2145                                 else
2146                                         dyn->d_tag = DT_SUNW_FILTER;

2148                                 (void) st_setstring(strtbl, dftp->dft_str,
2149                                     &stoff);
2150                                 dyn->d_un.d_val = stoff;
2151                                 dftp->dft_ndx = (Half)(((uintptr_t)dyn -
2152                                     (uintptr_t)_dyn) / sizeof (Dyn));
2153                                 dyn++;
2154                         }
2155                 }
2156                 if (((sdp = ld_sym_find(MSG_ORIG(MSG_SYM_INIT_U),
2157                     SYM_NOHASH, 0, ofl)) != NULL) &&
2158                     (sdp->sd_ref == REF_REL_NEED) &&
2159                     (sdp->sd_sym->st_shndx != SHN_UNDEF)) {
2160                         dyn->d_tag = DT_INIT;
2161                         dyn->d_un.d_ptr = sdp->sd_sym->st_value;
2162                         dyn++;
2163                 }
2164                 if (((sdp = ld_sym_find(MSG_ORIG(MSG_SYM_FINI_U),
2165                     SYM_NOHASH, 0, ofl)) != NULL) &&
2166                     (sdp->sd_ref == REF_REL_NEED) &&
2167                     (sdp->sd_sym->st_shndx != SHN_UNDEF)) {
2168                         dyn->d_tag = DT_FINI;
2169                         dyn->d_un.d_ptr = sdp->sd_sym->st_value;
2170                         dyn++;
2171                 }
2172                 if (ofl->ofl_soname) {
2173                         dyn->d_tag = DT_SONAME;
2174                         (void) st_setstring(strtbl, ofl->ofl_soname, &stoff);
2175                         dyn->d_un.d_val = stoff;
2176                         dyn++;
2177                 }
2178                 if (ofl->ofl_filtees) {
2179                         if (flags & FLG_OF_AUX) {
```

```
2180                                dyn->d_tag = DT_AUXILIARY;
2181                        } else {
2182                                dyn->d_tag = DT_FILTER;
2183                        }
2184                        (void) st_setstring(strtbl, ofl->ofl_filtees, &stoff);
2185                        dyn->d_un.d_val = stoff;
2186                        dyn++;
2187                }
2188        }

2190        if (ofl->ofl_rpath) {
2191                (void) st_setstring(strtbl, ofl->ofl_rpath, &stoff);
2192                dyn->d_tag = DT_RUNPATH;
2193                dyn->d_un.d_val = stoff;
2194                dyn++;
2195                dyn->d_tag = DT_RPATH;
2196                dyn->d_un.d_val = stoff;
2197                dyn++;
2198        }

2200        if (not_relobj) {
2201                Aliste  idx;
2202                Sg_desc *sgp;

2204                if (ofl->ofl_config) {
2205                        dyn->d_tag = DT_CONFIG;
2206                        (void) st_setstring(strtbl, ofl->ofl_config, &stoff);
2207                        dyn->d_un.d_val = stoff;
2208                        dyn++;
2209                }
2210                if (ofl->ofl_depaudit) {
2211                        dyn->d_tag = DT_DEPAUDIT;
2212                        (void) st_setstring(strtbl, ofl->ofl_depaudit, &stoff);
2213                        dyn->d_un.d_val = stoff;
2214                        dyn++;
2215                }
2216                if (ofl->ofl_audit) {
2217                        dyn->d_tag = DT_AUDIT;
2218                        (void) st_setstring(strtbl, ofl->ofl_audit, &stoff);
2219                        dyn->d_un.d_val = stoff;
2220                        dyn++;
2221                }

2223                dyn->d_tag = DT_HASH;
2224                dyn->d_un.d_ptr = ofl->ofl_oshash->os_shdr->sh_addr;
2225                dyn++;

2227                shdr = strosp->os_shdr;
2228                dyn->d_tag = DT_STRTAB;
2229                dyn->d_un.d_ptr = shdr->sh_addr;
2230                dyn++;

2232                dyn->d_tag = DT_STRSZ;
2233                dyn->d_un.d_ptr = shdr->sh_size;
2234                dyn++;

2236                /*
2237                 * Note, the shdr is set and used in the ofl->ofl_osldynsym case
2238                 * that follows.
2239                 */
2240                shdr = symosp->os_shdr;
2241                dyn->d_tag = DT_SYMTAB;
2242                dyn->d_un.d_ptr = shdr->sh_addr;
2243                dyn++;

2245                dyn->d_tag = DT_SYMENT;
```



```
2246                dyn->d_un.d_ptr = shdr->sh_entsize;
2247                dyn++;

2249                if (ofl->ofl_osldynsym) {
2250                        Shdr    *lshdr = ofl->ofl_osldynsym->os_shdr;

2252                        /*
2253                         * We have arranged for the .SUNW_ldynsym data to be
2254                         * immediately in front of the .dynsym data.
2255                         * This means that you could start at the top
2256                         * of .SUNW_ldynsym and see the data for both tables
2257                         * without a break. This is the view we want to
2258                         * provide for DT_SUNW_SYMTAB, which is why we
2259                         * add the lengths together.
2260                         */
2261                        dyn->d_tag = DT_SUNW_SYMTAB;
2262                        dyn->d_un.d_ptr = lshdr->sh_addr;
2263                        dyn++;

2265                        dyn->d_tag = DT_SUNW_SYMSZ;
2266                        dyn->d_un.d_val = lshdr->sh_size + shdr->sh_size;
2267                        dyn++;
2268                }

2270                if (ofl->ofl_osdynsymsort || ofl->ofl_osdyntlssort) {
2271                        dyn->d_tag = DT_SUNW_SORTENT;
2272                        dyn->d_un.d_val = sizeof (Word);
2273                        dyn++;
2274                }

2276                if (ofl->ofl_osdynsymsort) {
2277                        shdr = ofl->ofl_osdynsymsort->os_shdr;

2279                        dyn->d_tag = DT_SUNW_SYMSORT;
2280                        dyn->d_un.d_ptr = shdr->sh_addr;
2281                        dyn++;

2283                        dyn->d_tag = DT_SUNW_SYMSORTSZ;
2284                        dyn->d_un.d_val = shdr->sh_size;
2285                        dyn++;
2286                }

2288                if (ofl->ofl_osdyntlssort) {
2289                        shdr = ofl->ofl_osdyntlssort->os_shdr;

2291                        dyn->d_tag = DT_SUNW_TLSSORT;
2292                        dyn->d_un.d_ptr = shdr->sh_addr;
2293                        dyn++;

2295                        dyn->d_tag = DT_SUNW_TLSSORTSZ;
2296                        dyn->d_un.d_val = shdr->sh_size;
2297                        dyn++;
2298                }

2300                /*
2301                 * Reserve the DT_CHECKSUM entry.  Its value will be filled in
2302                 * after the complete image is built.
2303                 */
2304                dyn->d_tag = DT_CHECKSUM;
2305                ofl->ofl_checksum = &dyn->d_un.d_val;
2306                dyn++;

2308                /*
2309                 * Versioning sections: DT_VERDEF and DT_VERNEED.
2310                 *
2311                 * The Solaris ld does not produce DT_VERSYM, but the GNU ld
```

```
2312                 * does, in order to support their style of versioning, which
2313                 * differs from ours:
2314                 *
2315                 *      - The top bit of the 16-bit Versym index is
2316                 *              not part of the version, but is interpreted
2317                 *              as a "hidden bit".
2318                 *
2319                 *      - External (SHN_UNDEF) symbols can have non-zero
2320                 *              Versym values, which specify versions in
2321                 *              referenced objects, via the Verneed section.
2322                 *
2323                 *      - The vna_other field of the Vernaux structures
2324                 *              found in the Verneed section are not zero as
2325                 *              with Solaris, but instead contain the version
2326                 *              index to be used by Versym indices to reference
2327                 *              the given external version.
2328                 *
2329                 * The Solaris ld, rtld, and elfdump programs all interpret the
2330                 * presence of DT_VERSYM as meaning that GNU versioning rules
2331                 * apply to the given file. If DT_VERSYM is not present,
2332                 * then Solaris versioning rules apply. If we should ever need
2333                 * to change our ld so that it does issue DT_VERSYM, then
2334                 * this rule for detecting GNU versioning will no longer work.
2335                 * In that case, we will have to invent a way to explicitly
2336                 * specify the style of versioning in use, perhaps via a
2337                 * new dynamic entry named something like DT_SUNW_VERSIONSTYLE,
2338                 * where the d_un.d_val value specifies which style is to be
2339                 * used.
2340                 */
2341                if ((flags & (FLG_OF_VERDEF | FLG_OF_NOVERSEC)) ==
2342                    FLG_OF_VERDEF) {
2343                        shdr = ofl->ofl_osverdef->os_shdr;

2345                        dyn->d_tag = DT_VERDEF;
2346                        dyn->d_un.d_ptr = shdr->sh_addr;
2347                        dyn++;
2348                        dyn->d_tag = DT_VERDEFNUM;
2349                        dyn->d_un.d_ptr = shdr->sh_info;
2350                        dyn++;
2351                }
2352                if ((flags & (FLG_OF_VERNEED | FLG_OF_NOVERSEC)) ==
2353                    FLG_OF_VERNEED) {
2354                        shdr = ofl->ofl_osverneed->os_shdr;

2356                        dyn->d_tag = DT_VERNEED;
2357                        dyn->d_un.d_ptr = shdr->sh_addr;
2358                        dyn++;
2359                        dyn->d_tag = DT_VERNEEDNUM;
2360                        dyn->d_un.d_ptr = shdr->sh_info;
2361                        dyn++;
2362                }

2364                if ((flags & FLG_OF_COMREL) && ofl->ofl_relocrelcnt) {
2365                        dyn->d_tag = ld_targ.t_m.m_rel_dt_count;
2366                        dyn->d_un.d_val = ofl->ofl_relocrelcnt;
2367                        dyn++;
2368                }
2369                if (flags & FLG_OF_TEXTREL) {
2370                        /*
2371                         * Only the presence of this entry is used in this
2372                         * implementation, not the value stored.
2373                         */
2374                        dyn->d_tag = DT_TEXTREL;
2375                        dyn->d_un.d_val = 0;
2376                        dyn++;
2377                }
```



```
2379                if (ofl->ofl_osfiniarray) {
2380                        shdr = ofl->ofl_osfiniarray->os_shdr;

2382                        dyn->d_tag = DT_FINI_ARRAY;
2383                        dyn->d_un.d_ptr = shdr->sh_addr;
2384                        dyn++;

2386                        dyn->d_tag = DT_FINI_ARRAYSZ;
2387                        dyn->d_un.d_val = shdr->sh_size;
2388                        dyn++;
2389                }

2391                if (ofl->ofl_osinitarray) {
2392                        shdr = ofl->ofl_osinitarray->os_shdr;

2394                        dyn->d_tag = DT_INIT_ARRAY;
2395                        dyn->d_un.d_ptr = shdr->sh_addr;
2396                        dyn++;

2398                        dyn->d_tag = DT_INIT_ARRAYSZ;
2399                        dyn->d_un.d_val = shdr->sh_size;
2400                        dyn++;
2401                }

2403                if (ofl->ofl_ospreinitarray) {
2404                        shdr = ofl->ofl_ospreinitarray->os_shdr;

2406                        dyn->d_tag = DT_PREINIT_ARRAY;
2407                        dyn->d_un.d_ptr = shdr->sh_addr;
2408                        dyn++;

2410                        dyn->d_tag = DT_PREINIT_ARRAYSZ;
2411                        dyn->d_un.d_val = shdr->sh_size;
2412                        dyn++;
2413                }

2415                if (ofl->ofl_pltcnt) {
2416                        shdr = ofl->ofl_osplt->os_relosdesc->os_shdr;

2418                        dyn->d_tag = DT_PLTRELSZ;
2419                        dyn->d_un.d_ptr = shdr->sh_size;
2420                        dyn++;
2421                        dyn->d_tag = DT_PLTREL;
2422                        dyn->d_un.d_ptr = ld_targ.t_m.m_rel_dt_type;
2423                        dyn++;
2424                        dyn->d_tag = DT_JMPREL;
2425                        dyn->d_un.d_ptr = shdr->sh_addr;
2426                        dyn++;
2427                }
2428                if (ofl->ofl_pltpad) {
2429                        shdr = ofl->ofl_osplt->os_shdr;

2431                        dyn->d_tag = DT_PLTPAD;
2432                        if (ofl->ofl_pltcnt) {
2433                                dyn->d_un.d_ptr = shdr->sh_addr +
2434                                    ld_targ.t_m.m_plt_reservsz +
2435                                    ofl->ofl_pltcnt * ld_targ.t_m.m_plt_entsize;
2436                        } else
2437                                dyn->d_un.d_ptr = shdr->sh_addr;
2438                        dyn++;
2439                        dyn->d_tag = DT_PLTPADSZ;
2440                        dyn->d_un.d_val = ofl->ofl_pltpad *
2441                            ld_targ.t_m.m_plt_entsize;
2442                        dyn++;
2443                }
```

```
2444                 if (ofl->ofl_relocsz) {
2445                         shdr = ofl->ofl_osrelhead->os_shdr;

2447                         dyn->d_tag = ld_targ.t_m.m_rel_dt_type;
2448                         dyn->d_un.d_ptr = shdr->sh_addr;
2449                         dyn++;
2450                         dyn->d_tag = ld_targ.t_m.m_rel_dt_size;
2451                         dyn->d_un.d_ptr = ofl->ofl_relocsz;
2452                         dyn++;
2453                         dyn->d_tag = ld_targ.t_m.m_rel_dt_ent;
2454                         if (shdr->sh_type == SHT_REL)
2455                                 dyn->d_un.d_ptr = sizeof (Rel);
2456                         else
2457                                 dyn->d_un.d_ptr = sizeof (Rela);
2458                         dyn++;
2459                 }
2460                 if (ofl->ofl_ossyminfo) {
2461                         shdr = ofl->ofl_ossyminfo->os_shdr;

2463                         dyn->d_tag = DT_SYMINFO;
2464                         dyn->d_un.d_ptr = shdr->sh_addr;
2465                         dyn++;
2466                         dyn->d_tag = DT_SYMINSZ;
2467                         dyn->d_un.d_val = shdr->sh_size;
2468                         dyn++;
2469                         dyn->d_tag = DT_SYMINENT;
2470                         dyn->d_un.d_val = sizeof (Syminfo);
2471                         dyn++;
2472                 }
2473                 if (ofl->ofl_osmove) {
2474                         shdr = ofl->ofl_osmove->os_shdr;

2476                         dyn->d_tag = DT_MOVETAB;
2477                         dyn->d_un.d_val = shdr->sh_addr;
2478                         dyn++;
2479                         dyn->d_tag = DT_MOVESZ;
2480                         dyn->d_un.d_val = shdr->sh_size;
2481                         dyn++;
2482                         dyn->d_tag = DT_MOVEENT;
2483                         dyn->d_un.d_val = shdr->sh_entsize;
2484                         dyn++;
2485                 }
2486                 if (ofl->ofl_regsymcnt) {
2487                         int     ndx;

2489                         for (ndx = 0; ndx < ofl->ofl_regsymsno; ndx++) {
2490                                 if ((sdp = ofl->ofl_regsyms[ndx]) == NULL)
2491                                         continue;

2493                                 dyn->d_tag = ld_targ.t_m.m_dt_register;
2494                                 dyn->d_un.d_val = sdp->sd_symndx;
2495                                 dyn++;
2496                         }
2497                 }

2499                 for (APLIST_TRAVERSE(ofl->ofl_rtldinfo, idx, sdp)) {
2500                         dyn->d_tag = DT_SUNW_RTLDINF;
2501                         dyn->d_un.d_ptr = sdp->sd_sym->st_value;
2502                         dyn++;
2503                 }

2505                 if (((sgp = ofl->ofl_osdynamic->os_sgdesc) != NULL) &&
2506                     (sgp->sg_phdr.p_flags & PF_W) && ofl->ofl_osinterp) {
2507                         dyn->d_tag = DT_DEBUG;
2508                         dyn->d_un.d_ptr = 0;
2509                         dyn++;
```



```
2510                 }

2512                 if (ofl->ofl_oscap) {
2513                         dyn->d_tag = DT_SUNW_CAP;
2514                         dyn->d_un.d_val = ofl->ofl_oscap->os_shdr->sh_addr;
2515                         dyn++;
2516                 }
2517                 if (ofl->ofl_oscapinfo) {
2518                         dyn->d_tag = DT_SUNW_CAPINFO;
2519                         dyn->d_un.d_val = ofl->ofl_oscapinfo->os_shdr->sh_addr;
2520                         dyn++;
2521                 }
2522                 if (ofl->ofl_oscapchain) {
2523                         shdr = ofl->ofl_oscapchain->os_shdr;

2525                         dyn->d_tag = DT_SUNW_CAPCHAIN;
2526                         dyn->d_un.d_val = shdr->sh_addr;
2527                         dyn++;
2528                         dyn->d_tag = DT_SUNW_CAPCHAINSZ;
2529                         dyn->d_un.d_val = shdr->sh_size;
2530                         dyn++;
2531                         dyn->d_tag = DT_SUNW_CAPCHAINENT;
2532                         dyn->d_un.d_val = shdr->sh_entsize;
2533                         dyn++;
2534                 }

2536                 if (ofl->ofl_aslr != 0) {
2537                         dyn->d_tag = DT_SUNW_ASLR;
2538                         dyn->d_un.d_val = (ofl->ofl_aslr == 1);
2539                         dyn++;
2540                 }

2542 #endif /* ! codereview */
2543                 if (flags & FLG_OF_SYMBOLIC) {
2544                         dyn->d_tag = DT_SYMBOLIC;
2545                         dyn->d_un.d_val = 0;
2546                         dyn++;
2547                 }
2548         }

2550         dyn->d_tag = DT_FLAGS;
2551         dyn->d_un.d_val = ofl->ofl_dtflags;
2552         dyn++;

2554         /*
2555          * If -Bdirect was specified, but some NODIRECT symbols were specified
2556          * via a mapfile, or -znodirect was used on the command line, then
2557          * clear the DF_1_DIRECT flag.  The resultant object will use per-symbol
2558          * direct bindings rather than be enabled for global direct bindings.
2559          *
2560          * If any no-direct bindings exist within this object, set the
2561          * DF_1_NODIRECT flag.  ld(1) recognizes this flag when processing
2562          * dependencies, and performs extra work to ensure that no direct
2563          * bindings are established to the no-direct symbols that exist
2564          * within these dependencies.
2565          */
2566         if (ofl->ofl_flags1 & FLG_OF1_NGLBDIR)
2567                 ofl->ofl_dtflags_1 &= ~DF_1_DIRECT;
2568         if (ofl->ofl_flags1 & FLG_OF1_NDIRECT)
2569                 ofl->ofl_dtflags_1 |= DF_1_NODIRECT;

2571         dyn->d_tag = DT_FLAGS_1;
2572         dyn->d_un.d_val = ofl->ofl_dtflags_1;
2573         dyn++;

2575         dyn->d_tag = DT_SUNW_STRPAD;
```

```
2576         dyn->d_un.d_val = DYNSTR_EXTRA_PAD;
2577         dyn++;

2579         dyn->d_tag = DT_SUNW_LDMACH;
2580         dyn->d_un.d_val = ld_sunw_ldmach();
2581         dyn++;

2583         (*ld_targ.t_mr.mr_mach_update_odynamic)(ofl, &dyn);

2585         for (cnt = 1 + DYNAMIC_EXTRA_ELTS; cnt--; dyn++) {
2586                 dyn->d_tag = DT_NULL;
2587                 dyn->d_un.d_val = 0;
2588         }

2590         /*
2591          * Ensure that we wrote the right number of entries. If not, we either
2592          * miscounted in make_dynamic(), or we did something wrong in this
2593          * function.
2594          */
2595         assert((ofl->ofl_osdynamic->os_shdr->sh_size /
2596             ofl->ofl_osdynamic->os_shdr->sh_entsize) ==
2597             ((uintptr_t)dyn - (uintptr_t)_dyn) / sizeof (*dyn));

2599         return (1);
2600 }

2602 /*
2603  * Build the version definition section
2604  */
2605 static int
2606 update_overdef(Ofl_desc *ofl)
2607 {
2608         Aliste          idx1;
2609         Ver_desc        *vdp, *_vdp;
2610         Verdef          *vdf, *_vdf;
2611         int             num = 0;
2612         Os_desc         *strosp;
2613         Str_tbl         *strtbl;

2615         /*
2616          * Determine which string table to use.
2617          */
2618         if (OFL_IS_STATIC_OBJ(ofl)) {
2619                 strtbl = ofl->ofl_strtab;
2620                 strosp = ofl->ofl_osstrtab;
2621         } else {
2622                 strtbl = ofl->ofl_dynstrtab;
2623                 strosp = ofl->ofl_osdynstr;
2624         }

2626         /*
2627          * Traverse the version descriptors and update the version structures
2628          * to point to the dynstr name in preparation for building the version
2629          * section structure.
2630          */
2631         for (APLIST_TRAVERSE(ofl->ofl_verdesc, idx1, vdp)) {
2632                 Sym_desc        *sdp;

2634                 if (vdp->vd_flags & VER_FLG_BASE) {
2635                         const char      *name = vdp->vd_name;
2636                         size_t          stoff;

2638                         /*
2639                          * Create a new string table entry to represent the base
2640                          * version name (there is no corresponding symbol for
2641                          * this).
```



```
2642                          */
2643                         (void) st_setstring(strtbl, name, &stoff);
2644                         /* LINTED */
2645                         vdp->vd_name = (const char *)stoff;
2646                 } else {
2647                         sdp = ld_sym_find(vdp->vd_name, vdp->vd_hash, 0, ofl);
2648                         /* LINTED */
2649                         vdp->vd_name = (const char *)
2650                             (uintptr_t)sdp->sd_sym->st_name;
2651                 }
2652         }

2654         _vdf = vdf = (Verdef *)ofl->ofl_osverdef->os_outdata->d_buf;

2656         /*
2657          * Traverse the version descriptors and update the version section to
2658          * reflect each version and its associated dependencies.
2659          */
2660         for (APLIST_TRAVERSE(ofl->ofl_verdesc, idx1, vdp)) {
2661                 Aliste          idx2;
2662                 Half            cnt = 1;
2663                 Verdaux         *vdap, *_vdap;

2665                 _vdap = vdap = (Verdaux *)(vdf + 1);

2667                 vdf->vd_version = VER_DEF_CURRENT;
2668                 vdf->vd_flags   = vdp->vd_flags & MSK_VER_USER;
2669                 vdf->vd_ndx     = vdp->vd_ndx;
2670                 vdf->vd_hash    = vdp->vd_hash;

2672                 /* LINTED */
2673                 vdap->vda_name = (uintptr_t)vdp->vd_name;
2674                 vdap++;
2675                 /* LINTED */
2676                 _vdap->vda_next = (Word)((uintptr_t)vdap - (uintptr_t)_vdap);

2678                 /*
2679                  * Traverse this versions dependency list generating the
2680                  * appropriate version dependency entries.
2681                  */
2682                 for (APLIST_TRAVERSE(vdp->vd_deps, idx2, _vdp)) {
2683                         /* LINTED */
2684                         vdap->vda_name = (uintptr_t)_vdp->vd_name;
2685                         _vdap = vdap;
2686                         vdap++, cnt++;
2687                         /* LINTED */
2688                         _vdap->vda_next = (Word)((uintptr_t)vdap -
2689                             (uintptr_t)_vdap);
2690                 }
2691                 _vdap->vda_next = 0;

2693                 /*
2694                  * Record the versions auxiliary array offset and the associated
2695                  * dependency count.
2696                  */
2697                 /* LINTED */
2698                 vdf->vd_aux = (Word)((uintptr_t)(vdf + 1) - (uintptr_t)vdf);
2699                 vdf->vd_cnt = cnt;

2701                 /*
2702                  * Record the next versions offset and update the version
2703                  * pointer.  Remember the previous version offset as the very
2704                  * last structures next pointer should be null.
2705                  */
2706                 _vdf = vdf;
2707                 vdf = (Verdef *)vdap, num++;
```

```
2708                 /* LINTED */
2709                 _vdf->vd_next = (Word)((uintptr_t)vdf - (uintptr_t)_vdf);
2710         }
2711         _vdf->vd_next = 0;

2713         /*
2714          * Record the string table association with the version definition
2715          * section, and the symbol table associated with the version symbol
2716          * table (the actual contents of the version symbol table are filled
2717          * in during symbol update).
2718          */
2719         /* LINTED */
2720         ofl->ofl_osverdef->os_shdr->sh_link = (Word)elf_ndxscn(strosp->os_scn);

2722         /*
2723          * The version definition sections 'info' field is used to indicate the
2724          * number of entries in this section.
2725          */
2726         ofl->ofl_osverdef->os_shdr->sh_info = num;

2728         return (1);
2729 }

2731 /*
2732  * Finish the version symbol index section
2733  */
2734 static void
2735 update_oversym(Ofl_desc *ofl)
2736 {
2737         Os_desc         *osp;

2739         /*
2740          * Record the symbol table associated with the version symbol table.
2741          * The contents of the version symbol table are filled in during
2742          * symbol update.
2743          */
2744         if (OFL_IS_STATIC_OBJ(ofl))
2745                 osp = ofl->ofl_ossymtab;
2746         else
2747                 osp = ofl->ofl_osdynsym;

2749         /* LINTED */
2750         ofl->ofl_osversym->os_shdr->sh_link = (Word)elf_ndxscn(osp->os_scn);
2751 }

2753 /*
2754  * Build the version needed section
2755  */
2756 static int
2757 update_overneed(Ofl_desc *ofl)
2758 {
2759         Aliste          idx1;
2760         Ifl_desc        *ifl;
2761         Verneed         *vnd, *_vnd;
2762         Os_desc         *strosp;
2763         Str_tbl         *strtbl;
2764         Word            num = 0;

2766         _vnd = vnd = (Verneed *)ofl->ofl_osverneed->os_outdata->d_buf;

2768         /*
2769          * Determine which string table is appropriate.
2770          */
2771         if (OFL_IS_STATIC_OBJ(ofl)) {
2772                 strosp = ofl->ofl_osstrtab;
2773                 strtbl = ofl->ofl_strtab;
```



```
2774         } else {
2775                 strosp = ofl->ofl_osdynstr;
2776                 strtbl = ofl->ofl_dynstrtab;
2777         }

2779         /*
2780          * Traverse the shared object list looking for dependencies that have
2781          * versions defined within them.
2782          */
2783         for (APLIST_TRAVERSE(ofl->ofl_sos, idx1, ifl)) {
2784                 Half            _cnt;
2785                 Word            cnt = 0;
2786                 Vernaux         *_vnap, *vnap;
2787                 size_t          stoff;

2789                 if (!(ifl->ifl_flags & FLG_IF_VERNEED))
2790                         continue;

2792                 vnd->vn_version = VER_NEED_CURRENT;

2794                 (void) st_setstring(strtbl, ifl->ifl_soname, &stoff);
2795                 vnd->vn_file = stoff;

2797                 _vnap = vnap = (Vernaux *)(vnd + 1);

2799                 /*
2800                  * Traverse the version index list recording
2801                  * each version as a needed dependency.
2802                  */
2803                 for (_cnt = 0; _cnt <= ifl->ifl_vercnt; _cnt++) {
2804                         Ver_index       *vip = &ifl->ifl_verndx[_cnt];

2806                         if (vip->vi_flags & FLG_VER_REFER) {
2807                                 (void) st_setstring(strtbl, vip->vi_name,
2808                                     &stoff);
2809                                 vnap->vna_name = stoff;

2811                                 if (vip->vi_desc) {
2812                                         vnap->vna_hash = vip->vi_desc->vd_hash;
2813                                         vnap->vna_flags =
2814                                             vip->vi_desc->vd_flags;
2815                                 } else {
2816                                         vnap->vna_hash = 0;
2817                                         vnap->vna_flags = 0;
2818                                 }
2819                                 vnap->vna_other = vip->vi_overndx;

2821                                 /*
2822                                  * If version A inherits version B, then
2823                                  * B is implicit in A. It suffices for ld.so.1
2824                                  * to verify A at runtime and skip B. The
2825                                  * version normalization process sets the INFO
2826                                  * flag for the versions we want ld.so.1 to
2827                                  * skip.
2828                                  */
2829                                 if (vip->vi_flags & VER_FLG_INFO)
2830                                         vnap->vna_flags |= VER_FLG_INFO;

2832                                 _vnap = vnap;
2833                                 vnap++, cnt++;
2834                                 _vnap->vna_next =
2835                                     /* LINTED */
2836                                     (Word)((uintptr_t)vnap - (uintptr_t)_vnap);
2837                         }
2838                 }
```

```
2840                 _vnap->vna_next = 0;
2841
2842                 /*
2843                  * Record the versions auxiliary array offset and
2844                  * the associated dependency count.
2845                  */
2846                 /* LINTED */
2847                 vnd->vn_aux = (Word)((uintptr_t)(vnd + 1) - (uintptr_t)vnd);
2848                 /* LINTED */
2849                 vnd->vn_cnt = (Half)cnt;
2850
2851                 /*
2852                  * Record the next versions offset and update the version
2853                  * pointer.  Remember the previous version offset as the very
2854                  * last structures next pointer should be null.
2855                  */
2856                 _vnd = vnd;
2857                 vnd = (Verneed *)vnap, num++;
2858                 /* LINTED */
2859                 _vnd->vn_next = (Word)((uintptr_t)vnd - (uintptr_t)_vnd);
2860         }
2861         _vnd->vn_next = 0;
2862
2863         /*
2864          * Use sh_link to record the associated string table section, and
2865          * sh_info to indicate the number of entries contained in the section.
2866          */
2867         /* LINTED */
2868         ofl->ofl_osverneed->os_shdr->sh_link = (Word)elf_ndxscn(strosp->os_scn);
2869         ofl->ofl_osverneed->os_shdr->sh_info = num;
2870
2871         return (1);
2872 }
2873
2874 /*
2875  * Update syminfo section.
2876  */
2877 static uintptr_t
2878 update_osyminfo(Ofl_desc *ofl)
2879 {
2880         Os_desc         *symosp, *infosp = ofl->ofl_ossyminfo;
2881         Syminfo         *sip = infosp->os_outdata->d_buf;
2882         Shdr            *shdr = infosp->os_shdr;
2883         char            *strtab;
2884         Aliste          idx;
2885         Sym_desc        *sdp;
2886         Sfltr_desc      *sftp;
2887
2888         if (ofl->ofl_flags & FLG_OF_RELOBJ) {
2889                 symosp = ofl->ofl_ossymtab;
2890                 strtab = ofl->ofl_osstrtab->os_outdata->d_buf;
2891         } else {
2892                 symosp = ofl->ofl_osdynsym;
2893                 strtab = ofl->ofl_osdynstr->os_outdata->d_buf;
2894         }
2895
2896         /* LINTED */
2897         infosp->os_shdr->sh_link = (Word)elf_ndxscn(symosp->os_scn);
2898         if (ofl->ofl_osdynamic)
2899                 infosp->os_shdr->sh_info =
2900                     /* LINTED */
2901                     (Word)elf_ndxscn(ofl->ofl_osdynamic->os_scn);
2902
2903         /*
2904          * Update any references with the index into the dynamic table.
2905          */
```



```
2906         for (APLIST_TRAVERSE(ofl->ofl_symdtent, idx, sdp))
2907                 sip[sdp->sd_symndx].si_boundto = sdp->sd_file->ifl_neededndx;
2908
2909         /*
2910          * Update any filtee references with the index into the dynamic table.
2911          */
2912         for (ALIST_TRAVERSE(ofl->ofl_symfltrs, idx, sftp)) {
2913                 Dfltr_desc      *dftp;
2914
2915                 dftp = alist_item(ofl->ofl_dtsfltrs, sftp->sft_idx);
2916                 sip[sftp->sft_sdp->sd_symndx].si_boundto = dftp->dft_ndx;
2917         }
2918
2919         /*
2920          * Display debugging information about section.
2921          */
2922         DBG_CALL(Dbg_syminfo_title(ofl->ofl_lml));
2923         if (DBG_ENABLED) {
2924                 Word    _cnt, cnt = shdr->sh_size / shdr->sh_entsize;
2925                 Sym     *symtab = symosp->os_outdata->d_buf;
2926                 Dyn     *dyn;
2927
2928                 if (ofl->ofl_osdynamic)
2929                         dyn = ofl->ofl_osdynamic->os_outdata->d_buf;
2930                 else
2931                         dyn = NULL;
2932
2933                 for (_cnt = 1; _cnt < cnt; _cnt++) {
2934                         if (sip[_cnt].si_flags || sip[_cnt].si_boundto)
2935                                 /* LINTED */
2936                                 DBG_CALL(Dbg_syminfo_entry(ofl->ofl_lml, _cnt,
2937                                     &sip[_cnt], &symtab[_cnt], strtab, dyn));
2938                 }
2939         }
2940         return (1);
2941 }
2942
2943 /*
2944  * Build the output elf header.
2945  */
2946 static uintptr_t
2947 update_oehdr(Ofl_desc * ofl)
2948 {
2949         Ehdr    *ehdr = ofl->ofl_nehdr;
2950
2951         /*
2952          * If an entry point symbol has already been established (refer
2953          * sym_validate()) simply update the elf header entry point with the
2954          * symbols value.  If no entry point is defined it will have been filled
2955          * with the start address of the first section within the text segment
2956          * (refer update_outfile()).
2957          */
2958         if (ofl->ofl_entry)
2959                 ehdr->e_entry =
2960                     ((Sym_desc *)(ofl->ofl_entry))->sd_sym->st_value;
2961
2962         ehdr->e_ident[EI_DATA] = ld_targ.t_m.m_data;
2963         ehdr->e_version = ofl->ofl_dehdr->e_version;
2964
2965         /*
2966          * When generating a relocatable object under -z symbolcap, set the
2967          * e_machine to be generic, and remove any e_flags.  Input relocatable
2968          * objects may identify alternative e_machine (m.machplus) and e_flags
2969          * values.  However, the functions within the created output object
2970          * are selected at runtime using the capabilities mechanism, which
2971          * supersedes the e-machine and e_flags information.  Therefore,
```

```
2972          * e_machine and e_flag values are not propagated to the output object,
2973          * as these values might prevent the kernel from loading the object
2974          * before the runtime linker gets control.
2975          */
2976         if (ofl->ofl_flags & FLG_OF_OTOSCAP) {
2977                 ehdr->e_machine = ld_targ.t_m.m_mach;
2978                 ehdr->e_flags = 0;
2979         } else {
2980                 /*
2981                  * Note. it may be necessary to update the e_flags field in the
2982                  * machine dependent section.
2983                  */
2984                 ehdr->e_machine = ofl->ofl_dehdr->e_machine;
2985                 ehdr->e_flags = ofl->ofl_dehdr->e_flags;

2987                 if (ehdr->e_machine != ld_targ.t_m.m_mach) {
2988                         if (ehdr->e_machine != ld_targ.t_m.m_machplus)
2989                                 return (S_ERROR);
2990                         if ((ehdr->e_flags & ld_targ.t_m.m_flagsplus) == 0)
2991                                 return (S_ERROR);
2992                 }
2993         }

2995         if (ofl->ofl_flags & FLG_OF_SHAROBJ)
2996                 ehdr->e_type = ET_DYN;
2997         else if (ofl->ofl_flags & FLG_OF_RELOBJ)
2998                 ehdr->e_type = ET_REL;
2999         else
3000                 ehdr->e_type = ET_EXEC;

3002         return (1);
3003 }

3005 /*
3006  * Perform move table expansion.
3007  */
3008 static void
3009 expand_move(Ofl_desc *ofl, Sym_desc *sdp, Move *mvp)
3010 {
3011         Os_desc         *osp;
3012         uchar_t         *taddr, *taddr0;
3013         Sxword          offset;
3014         Half            cnt;
3015         uint_t          stride;

3017         osp = ofl->ofl_isparexpn->is_osdesc;
3018         offset = sdp->sd_sym->st_value - osp->os_shdr->sh_addr;

3020         taddr0 = taddr = osp->os_outdata->d_buf;
3021         taddr += offset;
3022         taddr = taddr + mvp->m_poffset;

3024         for (cnt = 0; cnt < mvp->m_repeat; cnt++) {
3025                 /* LINTED */
3026                 DBG_CALL(Dbg_move_expand(ofl->ofl_lml, mvp,
3027                     (Addr)(taddr - taddr0)));
3028                 stride = (uint_t)mvp->m_stride + 1;

3030                 /*
3031                  * Update the target address based upon the move entry size.
3032                  * This size was validated in ld_process_move().
3033                  */
3034                 /* LINTED */
3035                 switch (ELF_M_SIZE(mvp->m_info)) {
3036                 case 1:
3037                         /* LINTED */
```



```
3038                         *taddr = (uchar_t)mvp->m_value;
3039                         taddr += stride;
3040                         break;
3041                 case 2:
3042                         /* LINTED */
3043                         *((Half *)taddr) = (Half)mvp->m_value;
3044                         taddr += 2 * stride;
3045                         break;
3046                 case 4:
3047                         /* LINTED */
3048                         *((Word *)taddr) = (Word)mvp->m_value;
3049                         taddr += 4 * stride;
3050                         break;
3051                 case 8:
3052                         /* LINTED */
3053                         *((u_longlong_t *)taddr) = mvp->m_value;
3054                         taddr += 8 * stride;
3055                         break;
3056                 }
3057         }
3058 }

3060 /*
3061  * Update Move sections.
3062  */
3063 static void
3064 update_move(Ofl_desc *ofl)
3065 {
3066         Word            ndx = 0;
3067         ofl_flag_t      flags = ofl->ofl_flags;
3068         Move            *omvp;
3069         Aliste          idx1;
3070         Sym_desc        *sdp;

3072         /*
3073          * Determine the index of the symbol table that will be referenced by
3074          * the Move section.
3075          */
3076         if (OFL_ALLOW_DYNSYM(ofl))
3077                 /* LINTED */
3078                 ndx = (Word) elf_ndxscn(ofl->ofl_osdynsym->os_scn);
3079         else if (!(flags & FLG_OF_STRIP) || (flags & FLG_OF_RELOBJ))
3080                 /* LINTED */
3081                 ndx = (Word) elf_ndxscn(ofl->ofl_ossymtab->os_scn);

3083         /*
3084          * Update sh_link of the Move section, and point to the new Move data.
3085          */
3086         if (ofl->ofl_osmove) {
3087                 ofl->ofl_osmove->os_shdr->sh_link = ndx;
3088                 omvp = (Move *)ofl->ofl_osmove->os_outdata->d_buf;
3089         }

3091         /*
3092          * Update symbol entry index
3093          */
3094         for (APLIST_TRAVERSE(ofl->ofl_parsyms, idx1, sdp)) {
3095                 Aliste          idx2;
3096                 Mv_desc         *mdp;

3098                 /*
3099                  * Expand move table
3100                  */
3101                 if (sdp->sd_flags & FLG_SY_PAREXPN) {
3102                         const char      *str;
```

```
3104                        if (flags & FLG_OF_STATIC)
3105                                str = MSG_INTL(MSG_PSYM_EXPREASON1);
3106                        else if (ofl->ofl_flags1 & FLG_OF1_NOPARTI)
3107                                str = MSG_INTL(MSG_PSYM_EXPREASON2);
3108                        else
3109                                str = MSG_INTL(MSG_PSYM_EXPREASON3);

3111                        DBG_CALL(Dbg_move_parexpn(ofl->ofl_lml,
3112                            sdp->sd_name, str));

3114                        for (ALIST_TRAVERSE(sdp->sd_move, idx2, mdp)) {
3115                                DBG_CALL(Dbg_move_entry1(ofl->ofl_lml, 0,
3116                                    mdp->md_move, sdp));
3117                                expand_move(ofl, sdp, mdp->md_move);
3118                        }
3119                        continue;
3120                }

3122                /*
3123                 * Process move table
3124                 */
3125                DBG_CALL(Dbg_move_outmove(ofl->ofl_lml, sdp->sd_name));

3127                for (ALIST_TRAVERSE(sdp->sd_move, idx2, mdp)) {
3128                        Move    *imvp;
3129                        int     idx = 1;
3130                        Sym     *sym;

3132                        imvp = mdp->md_move;
3133                        sym = sdp->sd_sym;

3135                        DBG_CALL(Dbg_move_entry1(ofl->ofl_lml, 1, imvp, sdp));

3137                        *omvp = *imvp;
3138                        if ((flags & FLG_OF_RELOBJ) == 0) {
3139                                if (ELF_ST_BIND(sym->st_info) == STB_LOCAL) {
3140                                        Os_desc *osp = sdp->sd_isc->is_osdesc;
3141                                        Word    ndx = osp->os_identndx;

3143                                        omvp->m_info =
3144                                            /* LINTED */
3145                                            ELF_M_INFO(ndx, imvp->m_info);

3147                                        if (ELF_ST_TYPE(sym->st_info) !=
3148                                            STT_SECTION) {
3149                                                omvp->m_poffset =
3150                                                    sym->st_value -
3151                                                    osp->os_shdr->sh_addr +
3152                                                    imvp->m_poffset;
3153                                        }
3154                                } else {
3155                                        omvp->m_info =
3156                                            /* LINTED */
3157                                            ELF_M_INFO(sdp->sd_symndx,
3158                                            imvp->m_info);
3159                                }
3160                        } else {
3161                                Boolean         isredloc = FALSE;

3163                                if ((ELF_ST_BIND(sym->st_info) == STB_LOCAL) &&
3164                                    (ofl->ofl_flags & FLG_OF_REDLSYM))
3165                                        isredloc = TRUE;

3167                                if (isredloc && !(sdp->sd_move)) {
3168                                        Os_desc *osp = sdp->sd_isc->is_osdesc;
3169                                        Word    ndx = osp->os_identndx;
```



```
3171                                        omvp->m_info =
3172                                            /* LINTED */
3173                                            ELF_M_INFO(ndx, imvp->m_info);

3175                                        omvp->m_poffset += sym->st_value;
3176                                } else {
3177                                        if (isredloc)
3178                                                DBG_CALL(Dbg_syms_reduce(ofl,
3179                                                    DBG_SYM_REDUCE_RETAIN,
3180                                                    sdp, idx,
3181                                                    ofl->ofl_osmove->os_name));

3183                                        omvp->m_info =
3184                                            /* LINTED */
3185                                            ELF_M_INFO(sdp->sd_symndx,
3186                                            imvp->m_info);
3187                                }
3188                        }

3190                        DBG_CALL(Dbg_move_entry1(ofl->ofl_lml, 0, omvp, sdp));
3191                        omvp++;
3192                        idx++;
3193                }
3194        }
3195 }

3197 /*
3198  * Scan through the SHT_GROUP output sections.  Update their sh_link/sh_info
3199  * fields as well as the section contents.
3200  */
3201 static uintptr_t
3202 update_ogroup(Ofl_desc *ofl)
3203 {
3204         Aliste          idx;
3205         Os_desc         *osp;
3206         uintptr_t       error = 0;

3208         for (APLIST_TRAVERSE(ofl->ofl_osgroups, idx, osp)) {
3209                 Is_desc         *isp;
3210                 Ifl_desc        *ifl;
3211                 Shdr            *shdr = osp->os_shdr;
3212                 Sym_desc        *sdp;
3213                 Xword           i, grpcnt;
3214                 Word            *gdata;

3216                 /*
3217                  * Since input GROUP sections always create unique
3218                  * output GROUP sections - we know there is only one
3219                  * item on the list.
3220                  */
3221                 isp = ld_os_first_isdesc(osp);

3223                 ifl = isp->is_file;
3224                 sdp = ifl->ifl_oldndx[isp->is_shdr->sh_info];
3225                 shdr->sh_link = (Word)elf_ndxscn(ofl->ofl_ossymtab->os_scn);
3226                 shdr->sh_info = sdp->sd_symndx;

3228                 /*
3229                  * Scan through the group data section and update
3230                  * all of the links to new values.
3231                  */
3232                 grpcnt = shdr->sh_size / shdr->sh_entsize;
3233                 gdata = (Word *)osp->os_outdata->d_buf;

3235                 for (i = 1; i < grpcnt; i++) {
```

```
3236                        Os_desc *_osp;
3237                        Is_desc *_isp = ifl->ifl_isdesc[gdata[i]];

3239                        /*
3240                         * If the referenced section didn't make it to the
3241                         * output file - just zero out the entry.
3242                         */
3243                        if ((_osp = _isp->is_osdesc) == NULL)
3244                                gdata[i] = 0;
3245                        else
3246                                gdata[i] = (Word)elf_ndxscn(_osp->os_scn);
3247                }
3248        }
3249        return (error);
3250 }

3252 static void
3253 update_ostrtab(Os_desc *osp, Str_tbl *stp, uint_t extra)
3254 {
3255        Elf_Data        *data;

3257        if (osp == NULL)
3258                return;

3260        data = osp->os_outdata;
3261        assert(data->d_size == (st_getstrtab_sz(stp) + extra));
3262        (void) st_setstrbuf(stp, data->d_buf, data->d_size - extra);
3263        /* If leaving an extra hole at the end, zero it */
3264        if (extra > 0)
3265                (void) memset((char *)data->d_buf + data->d_size - extra,
3266                    0x0, extra);
3267 }

3269 /*
3270  * Update capabilities information.
3271  *
3272  * If string table capabilities exist, then the associated string must be
3273  * translated into an offset into the string table.
3274  */
3275 static void
3276 update_oscap(Ofl_desc *ofl)
3277 {
3278        Os_desc         *strosp, *cosp;
3279        Cap             *cap;
3280        Str_tbl         *strtbl;
3281        Capstr          *capstr;
3282        size_t          stoff;
3283        Aliste          idx1;

3285        /*
3286         * Determine which symbol table or string table is appropriate.
3287         */
3288        if (OFL_IS_STATIC_OBJ(ofl)) {
3289                strosp = ofl->ofl_osstrtab;
3290                strtbl = ofl->ofl_strtab;
3291        } else {
3292                strosp = ofl->ofl_osdynstr;
3293                strtbl = ofl->ofl_dynstrtab;
3294        }

3296        /*
3297         * If symbol capabilities exist, set the sh_link field of the .SUNW_cap
3298         * section to the .SUNW_capinfo section.
3299         */
3300        if (ofl->ofl_oscapinfo) {
3301                cosp = ofl->ofl_oscap;
```



```
3302                cosp->os_shdr->sh_link =
3303                    (Word)elf_ndxscn(ofl->ofl_oscapinfo->os_scn);
3304        }

3306        /*
3307         * If there are capability strings to process, set the sh_info
3308         * field of the .SUNW_cap section to the associated string table, and
3309         * proceed to process any CA_SUNW_PLAT entries.
3310         */
3311        if ((ofl->ofl_flags & FLG_OF_CAPSTRS) == 0)
3312                return;

3314        cosp = ofl->ofl_oscap;
3315        cosp->os_shdr->sh_info = (Word)elf_ndxscn(strosp->os_scn);

3317        cap = ofl->ofl_oscap->os_outdata->d_buf;

3319        /*
3320         * Determine whether an object capability identifier, or object
3321         * machine/platform capabilities exists.
3322         */
3323        capstr = &ofl->ofl_ocapset.oc_id;
3324        if (capstr->cs_str) {
3325                (void) st_setstring(strtbl, capstr->cs_str, &stoff);
3326                cap[capstr->cs_ndx].c_un.c_ptr = stoff;
3327        }
3328        for (ALIST_TRAVERSE(ofl->ofl_ocapset.oc_plat.cl_val, idx1, capstr)) {
3329                (void) st_setstring(strtbl, capstr->cs_str, &stoff);
3330                cap[capstr->cs_ndx].c_un.c_ptr = stoff;
3331        }
3332        for (ALIST_TRAVERSE(ofl->ofl_ocapset.oc_mach.cl_val, idx1, capstr)) {
3333                (void) st_setstring(strtbl, capstr->cs_str, &stoff);
3334                cap[capstr->cs_ndx].c_un.c_ptr = stoff;
3335        }

3337        /*
3338         * Determine any symbol capability identifiers, or machine/platform
3339         * capabilities.
3340         */
3341        if (ofl->ofl_capgroups) {
3342                Cap_group       *cgp;

3344                for (APLIST_TRAVERSE(ofl->ofl_capgroups, idx1, cgp)) {
3345                        Objcapset       *ocapset = &cgp->cg_set;
3346                        Aliste          idx2;

3348                        capstr = &ocapset->oc_id;
3349                        if (capstr->cs_str) {
3350                                (void) st_setstring(strtbl, capstr->cs_str,
3351                                    &stoff);
3352                                cap[capstr->cs_ndx].c_un.c_ptr = stoff;
3353                        }
3354                        for (ALIST_TRAVERSE(ocapset->oc_plat.cl_val, idx2,
3355                            capstr)) {
3356                                (void) st_setstring(strtbl, capstr->cs_str,
3357                                    &stoff);
3358                                cap[capstr->cs_ndx].c_un.c_ptr = stoff;
3359                        }
3360                        for (ALIST_TRAVERSE(ocapset->oc_mach.cl_val, idx2,
3361                            capstr)) {
3362                                (void) st_setstring(strtbl, capstr->cs_str,
3363                                    &stoff);
3364                                cap[capstr->cs_ndx].c_un.c_ptr = stoff;
3365                        }
3366                }
3367        }
```

```
3368 }
3369
3370 /*
3371  * Update the .SUNW_capinfo, and possibly the .SUNW_capchain sections.
3372  */
3373 static void
3374 update_oscapinfo(Ofl_desc *ofl)
3375 {
3376         Os_desc         *symosp, *ciosp, *ccosp = NULL;
3377         Capinfo         *ocapinfo;
3378         Capchain        *ocapchain;
3379         Cap_avlnode     *cav;
3380         Word            chainndx = 0;
3381
3382         /*
3383          * Determine which symbol table is appropriate.
3384          */
3385         if (OFL_IS_STATIC_OBJ(ofl))
3386                 symosp = ofl->ofl_ossymtab;
3387         else
3388                 symosp = ofl->ofl_osdynsym;
3389
3390         /*
3391          * Update the .SUNW_capinfo sh_link to point to the appropriate symbol
3392          * table section.  If we're creating a dynamic object, the
3393          * .SUNW_capinfo sh_info is updated to point to the .SUNW_capchain
3394          * section.
3395          */
3396         ciosp = ofl->ofl_oscapinfo;
3397         ciosp->os_shdr->sh_link = (Word)elf_ndxscn(symosp->os_scn);
3398
3399         if (OFL_IS_STATIC_OBJ(ofl) == 0) {
3400                 ccosp = ofl->ofl_oscapchain;
3401                 ciosp->os_shdr->sh_info = (Word)elf_ndxscn(ccosp->os_scn);
3402         }
3403
3404         /*
3405          * Establish the data for each section.  The first element of each
3406          * section defines the section's version number.
3407          */
3408         ocapinfo = ciosp->os_outdata->d_buf;
3409         ocapinfo[0] = CAPINFO_CURRENT;
3410         if (ccosp) {
3411                 ocapchain = ccosp->os_outdata->d_buf;
3412                 ocapchain[chainndx++] = CAPCHAIN_CURRENT;
3413         }
3414
3415         /*
3416          * Traverse all capabilities families.  Each member has a .SUNW_capinfo
3417          * assignment.  The .SUNW_capinfo entry differs for relocatable objects
3418          * and dynamic objects.
3419          *
3420          * Relocatable objects:
3421          *                       ELF_C_GROUP             ELF_C_SYM
3422          *
3423          * Family lead:          CAPINFO_SUNW_GLOB       lead symbol index
3424          * Family lead alias:    CAPINFO_SUNW_GLOB       lead symbol index
3425          * Family member:        .SUNW_cap index         lead symbol index
3426          *
3427          * Dynamic objects:
3428          *                       ELF_C_GROUP             ELF_C_SYM
3429          *
3430          * Family lead:          CAPINFO_SUNW_GLOB       .SUNW_capchain index
3431          * Family lead alias:    CAPINFO_SUNW_GLOB       .SUNW_capchain index
3432          * Family member:        .SUNW_cap index         lead symbol index
3433          *
```



```
3434          * The ELF_C_GROUP field identifies a capabilities symbol.  Lead
3435          * capability symbols, and lead capability aliases are identified by
3436          * a CAPINFO_SUNW_GLOB group identifier.  For family members, the
3437          * ELF_C_GROUP provides an index to the associate capabilities group
3438          * (i.e, an index into the SUNW_cap section that defines a group).
3439          *
3440          * For relocatable objects, the ELF_C_SYM field identifies the lead
3441          * capability symbol.  For the lead symbol itself, the .SUNW_capinfo
3442          * index is the same as the ELF_C_SYM value.  For lead alias symbols,
3443          * the .SUNW_capinfo index differs from the ELF_C_SYM value.  This
3444          * differentiation of CAPINFO_SUNW_GLOB symbols allows ld(1) to
3445          * identify, and propagate lead alias symbols.  For example, the lead
3446          * capability symbol memcpy() would have the ELF_C_SYM for memcpy(),
3447          * and the lead alias _memcpy() would also have the ELF_C_SYM for
3448          * memcpy().
3449          *
3450          * For dynamic objects, both a lead capability symbol, and alias symbol
3451          * would have a ELF_C_SYM value that represents the same capability
3452          * chain index.  The capability chain allows ld.so.1 to traverse a
3453          * family chain for a given lead symbol, and select the most appropriate
3454          * family member.  The .SUNW_capchain array contains a series of symbol
3455          * indexes for each family member:
3456          *
3457          *    chaincap[n]  chaincap[n + 1]  chaincap[n + 2]  chaincap[n + x]
3458          *      foo() ndx    foo%x() ndx        foo%y() ndx     0
3459          *
3460          * For family members, the ELF_C_SYM value associates the capability
3461          * members with their family lead symbol.  This association, although
3462          * unused within a dynamic object, allows ld(1) to identify, and
3463          * propagate family members when processing relocatable objects.
3464          */
3465         for (cav = avl_first(ofl->ofl_capfamilies); cav;
3466             cav = AVL_NEXT(ofl->ofl_capfamilies, cav)) {
3467                 Cap_sym         *csp;
3468                 Aliste          idx;
3469                 Sym_desc        *asdp, *lsdp = cav->cn_symavlnode.sav_sdp;
3470
3471                 if (ccosp) {
3472                         /*
3473                          * For a dynamic object, identify this lead symbol, and
3474                          * point it to the head of a capability chain.  Set the
3475                          * head of the capability chain to the same lead symbol.
3476                          */
3477                         ocapinfo[lsdp->sd_symndx] =
3478                             ELF_C_INFO(chainndx, CAPINFO_SUNW_GLOB);
3479                         ocapchain[chainndx] = lsdp->sd_symndx;
3480                 } else {
3481                         /*
3482                          * For a relocatable object, identify this lead symbol,
3483                          * and set the lead symbol index to itself.
3484                          */
3485                         ocapinfo[lsdp->sd_symndx] =
3486                             ELF_C_INFO(lsdp->sd_symndx, CAPINFO_SUNW_GLOB);
3487                 }
3488
3489                 /*
3490                  * Gather any lead symbol aliases.
3491                  */
3492                 for (APLIST_TRAVERSE(cav->cn_aliases, idx, asdp)) {
3493                         if (ccosp) {
3494                                 /*
3495                                  * For a dynamic object, identify this lead
3496                                  * alias symbol, and point it to the same
3497                                  * capability chain index as the lead symbol.
3498                                  */
3499                                 ocapinfo[asdp->sd_symndx] =
```

```
3500                                    ELF_C_INFO(chainndx, CAPINFO_SUNW_GLOB);
3501                            } else {
3502                                    /*
3503                                     * For a relocatable object, identify this lead
3504                                     * alias symbol, and set the lead symbol index
3505                                     * to the lead symbol.
3506                                     */
3507                                    ocapinfo[asdp->sd_symndx] =
3508                                        ELF_C_INFO(lsdp->sd_symndx,
3509                                        CAPINFO_SUNW_GLOB);
3510                            }
3511                    }

3513                    chainndx++;

3515                    /*
3516                     * Gather the family members.
3517                     */
3518                    for (APLIST_TRAVERSE(cav->cn_members, idx, csp)) {
3519                            Sym_desc        *msdp = csp->cs_sdp;

3521                            /*
3522                             * Identify the members capability group, and the lead
3523                             * symbol of the family this symbol is a member of.
3524                             */
3525                            ocapinfo[msdp->sd_symndx] =
3526                                ELF_C_INFO(lsdp->sd_symndx, csp->cs_group->cg_ndx);
3527                            if (ccosp) {
3528                                    /*
3529                                     * For a dynamic object, set the next capability
3530                                     * chain to point to this family member.
3531                                     */
3532                                    ocapchain[chainndx++] = msdp->sd_symndx;
3533                            }
3534                    }

3536                    /*
3537                     * Any chain of family members is terminated with a 0 element.
3538                     */
3539                    if (ccosp)
3540                            ocapchain[chainndx++] = 0;
3541            }
3542    }

3544    /*
3545     * Translate the shdr->sh_{link, info} from its input section value to that
3546     * of the corresponding shdr->sh_{link, info} output section value.
3547     */
3548    static Word
3549    translate_link(Ofl_desc *ofl, Os_desc *osp, Word link, const char *msg)
3550    {
3551            Is_desc         *isp;
3552            Ifl_desc        *ifl;

3554            /*
3555             * Don't translate the special section numbers.
3556             */
3557            if (link >= SHN_LORESERVE)
3558                    return (link);

3560            /*
3561             * Does this output section translate back to an input file.  If not
3562             * then there is no translation to do.  In this case we will assume that
3563             * if sh_link has a value, it's the right value.
3564             */
3565            isp = ld_os_first_isdesc(osp);
```



```
3566            if ((ifl = isp->is_file) == NULL)
3567                    return (link);

3569            /*
3570             * Sanity check to make sure that the sh_{link, info} value
3571             * is within range for the input file.
3572             */
3573            if (link >= ifl->ifl_shnum) {
3574                    ld_eprintf(ofl, ERR_WARNING, msg, ifl->ifl_name,
3575                        EC_WORD(isp->is_scnndx), isp->is_name, EC_XWORD(link));
3576                    return (link);
3577            }

3579            /*
3580             * Follow the link to the input section.
3581             */
3582            if ((isp = ifl->ifl_isdesc[link]) == NULL)
3583                    return (0);
3584            if ((osp = isp->is_osdesc) == NULL)
3585                    return (0);

3587            /* LINTED */
3588            return ((Word)elf_ndxscn(osp->os_scn));
3589    }

3591    /*
3592     * Having created all of the necessary sections, segments, and associated
3593     * headers, fill in the program headers and update any other data in the
3594     * output image.  Some general rules:
3595     *
3596     *  -   If an interpreter is required always generate a PT_PHDR entry as
3597     *      well.  It is this entry that triggers the kernel into passing the
3598     *      interpreter an aux vector instead of just a file descriptor.
3599     *
3600     *  -   When generating an image that will be interpreted (ie. a dynamic
3601     *      executable, a shared object, or a static executable that has been
3602     *      provided with an interpreter - weird, but possible), make the initial
3603     *      loadable segment include both the ehdr and phdr[].  Both of these
3604     *      tables are used by the interpreter therefore it seems more intuitive
3605     *      to explicitly defined them as part of the mapped image rather than
3606     *      relying on page rounding by the interpreter to allow their access.
3607     *
3608     *  -   When generating a static image that does not require an interpreter
3609     *      have the first loadable segment indicate the address of the first
3610     *      .section as the start address (things like /kernel/unix and ufsboot
3611     *      expect this behavior).
3612     */
3613    uintptr_t
3614    ld_update_outfile(Ofl_desc *ofl)
3615    {
3616            Addr            size, etext, vaddr;
3617            Sg_desc         *sgp;
3618            Sg_desc         *dtracesgp = NULL, *capsgp = NULL, *intpsgp = NULL;
3619            Os_desc         *osp;
3620            int             phdrndx = 0, segndx = -1, secndx, intppndx, intpsndx;
3621            int             dtracepndx, dtracesndx, cappndx, capsndx;
3622            Ehdr            *ehdr = ofl->ofl_nehdr;
3623            Shdr            *hshdr;
3624            Phdr            *_phdr = NULL;
3625            Word            phdrsz = (ehdr->e_phnum * ehdr->e_phentsize), shscnndx;
3626            ofl_flag_t      flags = ofl->ofl_flags;
3627            Word            ehdrsz = ehdr->e_ehsize;
3628            Boolean         nobits;
3629            Off             offset;
3630            Aliste          idx1;
```

```
3632         /*
3633          * Initialize the starting address for the first segment.  Executables
3634          * have different starting addresses depending upon the target ABI,
3635          * where as shared objects have a starting address of 0.  If this is
3636          * a 64-bit executable that is being constructed to run in a restricted
3637          * address space, use an alternative origin that will provide more free
3638          * address space for the the eventual process.
3639          */
3640         if (ofl->ofl_flags & FLG_OF_EXEC) {
3641 #if     defined(_ELF64)
3642                 if (ofl->ofl_ocapset.oc_sf_1.cm_val & SF1_SUNW_ADDR32)
3643                         vaddr = ld_targ.t_m.m_segm_aorigin;
3644                 else
3645 #endif
3646                         vaddr = ld_targ.t_m.m_segm_origin;
3647         } else
3648                 vaddr = 0;

3650         /*
3651          * Loop through the segment descriptors and pick out what we need.
3652          */
3653         DBG_CALL(Dbg_seg_title(ofl->ofl_lml));
3654         for (APLIST_TRAVERSE(ofl->ofl_segs, idx1, sgp)) {
3655                 Phdr            *phdr = &(sgp->sg_phdr);
3656                 Xword           p_align;
3657                 Aliste          idx2;
3658                 Sym_desc        *sdp;

3660                 segndx++;

3662                 /*
3663                  * If an interpreter is required generate a PT_INTERP and
3664                  * PT_PHDR program header entry.  The PT_PHDR entry describes
3665                  * the program header table itself.  This information will be
3666                  * passed via the aux vector to the interpreter (ld.so.1).
3667                  * The program header array is actually part of the first
3668                  * loadable segment (and the PT_PHDR entry is the first entry),
3669                  * therefore its virtual address isn't known until the first
3670                  * loadable segment is processed.
3671                  */
3672                 if (phdr->p_type == PT_PHDR) {
3673                         if (ofl->ofl_osinterp) {
3674                                 phdr->p_offset = ehdr->e_phoff;
3675                                 phdr->p_filesz = phdr->p_memsz = phdrsz;

3677                                 DBG_CALL(Dbg_seg_entry(ofl, segndx, sgp));
3678                                 ofl->ofl_phdr[phdrndx++] = *phdr;
3679                         }
3680                         continue;
3681                 }
3682                 if (phdr->p_type == PT_INTERP) {
3683                         if (ofl->ofl_osinterp) {
3684                                 intpsgp = sgp;
3685                                 intpsndx = segndx;
3686                                 intppndx = phdrndx++;
3687                         }
3688                         continue;
3689                 }

3691                 /*
3692                  * If we are creating a PT_SUNWDTRACE segment, remember where
3693                  * the program header is.  The header values are assigned after
3694                  * update_osym() has completed and the symbol table addresses
3695                  * have been updated.
3696                  */
3697                 if (phdr->p_type == PT_SUNWDTRACE) {
```



```
3698                         if (ofl->ofl_dtracesym &&
3699                             ((flags & FLG_OF_RELOBJ) == 0)) {
3700                                 dtracesgp = sgp;
3701                                 dtracesndx = segndx;
3702                                 dtracepndx = phdrndx++;
3703                         }
3704                         continue;
3705                 }

3707                 /*
3708                  * If a hardware/software capabilities section is required,
3709                  * generate the PT_SUNWCAP header.  Note, as this comes before
3710                  * the first loadable segment, we don't yet know its real
3711                  * virtual address.  This is updated later.
3712                  */
3713                 if (phdr->p_type == PT_SUNWCAP) {
3714                         if (ofl->ofl_oscap && (ofl->ofl_flags & FLG_OF_PTCAP) &&
3715                             ((flags & FLG_OF_RELOBJ) == 0)) {
3716                                 capsgp = sgp;
3717                                 capsndx = segndx;
3718                                 cappndx = phdrndx++;
3719                         }
3720                         continue;
3721                 }

3723                 /*
3724                  * As the dynamic program header occurs after the loadable
3725                  * headers in the segment descriptor table, all the address
3726                  * information for the .dynamic output section will have been
3727                  * figured out by now.
3728                  */
3729                 if (phdr->p_type == PT_DYNAMIC) {
3730                         if (OFL_ALLOW_DYNSYM(ofl)) {
3731                                 Shdr    *shdr = ofl->ofl_osdynamic->os_shdr;

3733                                 phdr->p_vaddr = shdr->sh_addr;
3734                                 phdr->p_offset = shdr->sh_offset;
3735                                 phdr->p_filesz = shdr->sh_size;
3736                                 phdr->p_flags = ld_targ.t_m.m_dataseg_perm;

3738                                 DBG_CALL(Dbg_seg_entry(ofl, segndx, sgp));
3739                                 ofl->ofl_phdr[phdrndx++] = *phdr;
3740                         }
3741                         continue;
3742                 }

3744                 /*
3745                  * As the unwind (.eh_frame_hdr) program header occurs after
3746                  * the loadable headers in the segment descriptor table, all
3747                  * the address information for the .eh_frame output section
3748                  * will have been figured out by now.
3749                  */
3750                 if (phdr->p_type == PT_SUNW_UNWIND) {
3751                         Shdr        *shdr;

3753                         if (ofl->ofl_unwindhdr == NULL)
3754                                 continue;

3756                         shdr = ofl->ofl_unwindhdr->os_shdr;

3758                         phdr->p_flags = PF_R;
3759                         phdr->p_vaddr = shdr->sh_addr;
3760                         phdr->p_memsz = shdr->sh_size;
3761                         phdr->p_filesz = shdr->sh_size;
3762                         phdr->p_offset = shdr->sh_offset;
3763                         phdr->p_align = shdr->sh_addralign;
```

```
3764                    phdr->p_paddr = 0;
3765                    ofl->ofl_phdr[phdrndx++] = *phdr;
3766                    continue;
3767            }

3769            /*
3770             * The sunwstack program is used to convey non-default
3771             * flags for the process stack. Only emit it if it would
3772             * change the default.
3773             */
3774            if (phdr->p_type == PT_SUNWSTACK) {
3775                    if (((flags & FLG_OF_RELOBJ) == 0) &&
3776                        ((sgp->sg_flags & FLG_SG_DISABLED) == 0))
3777                            ofl->ofl_phdr[phdrndx++] = *phdr;
3778                    continue;
3779            }

3781            /*
3782             * As the TLS program header occurs after the loadable
3783             * headers in the segment descriptor table, all the address
3784             * information for the .tls output section will have been
3785             * figured out by now.
3786             */
3787            if (phdr->p_type == PT_TLS) {
3788                    Os_desc         *tlsosp;
3789                    Shdr            *lastfileshdr = NULL;
3790                    Shdr            *firstshdr = NULL, *lastshdr;
3791                    Aliste          idx;

3793                    if (ofl->ofl_ostlsseg == NULL)
3794                            continue;

3796                    /*
3797                     * Scan the output sections that have contributed TLS.
3798                     * Remember the first and last so as to determine the
3799                     * TLS memory size requirement.  Remember the last
3800                     * progbits section to determine the TLS data
3801                     * contribution, which determines the TLS program
3802                     * header filesz.
3803                     */
3804                    for (APLIST_TRAVERSE(ofl->ofl_ostlsseg, idx, tlsosp)) {
3805                            Shdr    *tlsshdr = tlsosp->os_shdr;

3807                            if (firstshdr == NULL)
3808                                    firstshdr = tlsshdr;
3809                            if (tlsshdr->sh_type != SHT_NOBITS)
3810                                    lastfileshdr = tlsshdr;
3811                            lastshdr = tlsshdr;
3812                    }

3814                    phdr->p_flags = PF_R | PF_W;
3815                    phdr->p_vaddr = firstshdr->sh_addr;
3816                    phdr->p_offset = firstshdr->sh_offset;
3817                    phdr->p_align = firstshdr->sh_addralign;

3819                    /*
3820                     * Determine the initialized TLS data size.  This
3821                     * address range is from the start of the TLS segment
3822                     * to the end of the last piece of initialized data.
3823                     */
3824                    if (lastfileshdr)
3825                            phdr->p_filesz = lastfileshdr->sh_offset +
3826                                lastfileshdr->sh_size - phdr->p_offset;
3827                    else
3828                            phdr->p_filesz = 0;
```



```
3830                    /*
3831                     * Determine the total TLS memory size.  This includes
3832                     * all TLS data and TLS uninitialized data.  This
3833                     * address range is from the start of the TLS segment
3834                     * to the memory address of the last piece of
3835                     * uninitialized data.
3836                     */
3837                    phdr->p_memsz = lastshdr->sh_addr +
3838                        lastshdr->sh_size - phdr->p_vaddr;

3840                    DBG_CALL(Dbg_seg_entry(ofl, segndx, sgp));
3841                    ofl->ofl_phdr[phdrndx] = *phdr;
3842                    ofl->ofl_tlsphdr = &ofl->ofl_phdr[phdrndx++];
3843                    continue;
3844            }

3846            /*
3847             * If this is an empty segment declaration, it will occur after
3848             * all other loadable segments.  As empty segments can be
3849             * defined with fixed addresses, make sure that no loadable
3850             * segments overlap.  This might occur as the object evolves
3851             * and the loadable segments grow, thus encroaching upon an
3852             * existing segment reservation.
3853             *
3854             * Segments are only created for dynamic objects, thus this
3855             * checking can be skipped when building a relocatable object.
3856             */
3857            if (!(flags & FLG_OF_RELOBJ) &&
3858                (sgp->sg_flags & FLG_SG_EMPTY)) {
3859                    int     i;
3860                    Addr    v_e;

3862                    vaddr = phdr->p_vaddr;
3863                    phdr->p_memsz = sgp->sg_length;
3864                    DBG_CALL(Dbg_seg_entry(ofl, segndx, sgp));
3865                    ofl->ofl_phdr[phdrndx++] = *phdr;

3867                    if (phdr->p_type != PT_LOAD)
3868                            continue;

3870                    v_e = vaddr + phdr->p_memsz;

3872                    /*
3873                     * Check overlaps
3874                     */
3875                    for (i = 0; i < phdrndx - 1; i++) {
3876                            Addr    p_s = (ofl->ofl_phdr[i]).p_vaddr;
3877                            Addr    p_e;

3879                            if ((ofl->ofl_phdr[i]).p_type != PT_LOAD)
3880                                    continue;

3882                            p_e = p_s + (ofl->ofl_phdr[i]).p_memsz;
3883                            if (((p_s <= vaddr) && (p_e > vaddr)) ||
3884                                ((vaddr <= p_s) && (v_e > p_s)))
3885                                    ld_eprintf(ofl, ERR_WARNING,
3886                                        MSG_INTL(MSG_UPD_SEGOVERLAP),
3887                                        ofl->ofl_name, EC_ADDR(p_e),
3888                                        sgp->sg_name, EC_ADDR(vaddr));
3889                    }
3890                    continue;
3891            }

3893            /*
3894             * Having processed any of the special program headers any
3895             * remaining headers will be built to express individual
```

```
3896                 * segments.  Segments are only built if they have output
3897                 * section descriptors associated with them (ie. some form of
3898                 * input section has been matched to this segment).
3899                 */
3900                if (sgp->sg_osdescs == NULL)
3901                        continue;
3902
3903                /*
3904                 * Determine the segments offset and size from the section
3905                 * information provided from elf_update().
3906                 * Allow for multiple NOBITS sections.
3907                 */
3908                osp = sgp->sg_osdescs->apl_data[0];
3909                hshdr = osp->os_shdr;
3910
3911                phdr->p_filesz = 0;
3912                phdr->p_memsz = 0;
3913                phdr->p_offset = offset = hshdr->sh_offset;
3914
3915                nobits = ((hshdr->sh_type == SHT_NOBITS) &&
3916                    ((sgp->sg_flags & FLG_SG_PHREQ) == 0));
3917
3918                for (APLIST_TRAVERSE(sgp->sg_osdescs, idx2, osp)) {
3919                        Shdr    *shdr = osp->os_shdr;
3920
3921                        p_align = 0;
3922                        if (shdr->sh_addralign > p_align)
3923                                p_align = shdr->sh_addralign;
3924
3925                        offset = (Off)S_ROUND(offset, shdr->sh_addralign);
3926                        offset += shdr->sh_size;
3927
3928                        if (shdr->sh_type != SHT_NOBITS) {
3929                                if (nobits) {
3930                                        ld_eprintf(ofl, ERR_FATAL,
3931                                            MSG_INTL(MSG_UPD_NOBITS));
3932                                        return (S_ERROR);
3933                                }
3934                                phdr->p_filesz = offset - phdr->p_offset;
3935                        } else if ((sgp->sg_flags & FLG_SG_PHREQ) == 0)
3936                                nobits = TRUE;
3937                }
3938                phdr->p_memsz = offset - hshdr->sh_offset;
3939
3940                /*
3941                 * If this is the first loadable segment of a dynamic object,
3942                 * or an interpreter has been specified (a static object built
3943                 * with an interpreter will still be given a PT_HDR entry), then
3944                 * compensate for the elf header and program header array.  Both
3945                 * of these are actually part of the loadable segment as they
3946                 * may be inspected by the interpreter.  Adjust the segments
3947                 * size and offset accordingly.
3948                 */
3949                if ((_phdr == NULL) && (phdr->p_type == PT_LOAD) &&
3950                    ((ofl->ofl_osinterp) || (flags & FLG_OF_DYNAMIC)) &&
3951                    (!(ofl->ofl_dtflags_1 & DF_1_NOHDR))) {
3952                        size = (Addr)S_ROUND((phdrsz + ehdrsz),
3953                            hshdr->sh_addralign);
3954                        phdr->p_offset -= size;
3955                        phdr->p_filesz += size;
3956                        phdr->p_memsz += size;
3957                }
3958
3959                /*
3960                 * If segment size symbols are required (specified via a
3961                 * mapfile) update their value.
```



```
3962                 */
3963                for (APLIST_TRAVERSE(sgp->sg_sizesym, idx2, sdp))
3964                        sdp->sd_sym->st_value = phdr->p_memsz;
3965
3966                /*
3967                 * If no file content has been assigned to this segment (it
3968                 * only contains no-bits sections), then reset the offset for
3969                 * consistency.
3970                 */
3971                if (phdr->p_filesz == 0)
3972                        phdr->p_offset = 0;
3973
3974                /*
3975                 * If a virtual address has been specified for this segment
3976                 * from a mapfile use it and make sure the previous segment
3977                 * does not run into this segment.
3978                 */
3979                if (phdr->p_type == PT_LOAD) {
3980                        if ((sgp->sg_flags & FLG_SG_P_VADDR)) {
3981                                if (_phdr && (vaddr > phdr->p_vaddr) &&
3982                                    (phdr->p_type == PT_LOAD))
3983                                        ld_eprintf(ofl, ERR_WARNING,
3984                                            MSG_INTL(MSG_UPD_SEGOVERLAP),
3985                                            ofl->ofl_name, EC_ADDR(vaddr),
3986                                            sgp->sg_name,
3987                                            EC_ADDR(phdr->p_vaddr));
3988                                vaddr = phdr->p_vaddr;
3989                                phdr->p_align = 0;
3990                        } else {
3991                                vaddr = phdr->p_vaddr =
3992                                    (Addr)S_ROUND(vaddr, phdr->p_align);
3993                        }
3994                }
3995
3996                /*
3997                 * Adjust the address offset and p_align if needed.
3998                 */
3999                if (((sgp->sg_flags & FLG_SG_P_VADDR) == 0) &&
4000                    ((ofl->ofl_dtflags_1 & DF_1_NOHDR) == 0)) {
4001                        if (phdr->p_align != 0)
4002                                vaddr += phdr->p_offset % phdr->p_align;
4003                        else
4004                                vaddr += phdr->p_offset;
4005                        phdr->p_vaddr = vaddr;
4006                }
4007
4008                /*
4009                 * If an interpreter is required set the virtual address of the
4010                 * PT_PHDR program header now that we know the virtual address
4011                 * of the loadable segment that contains it.  Update the
4012                 * PT_SUNWCAP header similarly.
4013                 */
4014                if ((_phdr == NULL) && (phdr->p_type == PT_LOAD)) {
4015                        _phdr = phdr;
4016
4017                        if ((ofl->ofl_dtflags_1 & DF_1_NOHDR) == 0) {
4018                                if (ofl->ofl_osinterp)
4019                                        ofl->ofl_phdr[0].p_vaddr =
4020                                            vaddr + ehdrsz;
4021
4022                                /*
4023                                 * Finally, if we're creating a dynamic object
4024                                 * (or a static object in which an interpreter
4025                                 * is specified) update the vaddr to reflect
4026                                 * the address of the first section within this
4027                                 * segment.
```

```
4028                                 */
4029                                if ((ofl->ofl_osinterp) ||
4030                                    (flags & FLG_OF_DYNAMIC))
4031                                        vaddr += size;
4032                        } else {
4033                                /*
4034                                 * If the DF_1_NOHDR flag was set, and an
4035                                 * interpreter is being generated, the PT_PHDR
4036                                 * will not be part of any loadable segment.
4037                                 */
4038                                if (ofl->ofl_osinterp) {
4039                                        ofl->ofl_phdr[0].p_vaddr = 0;
4040                                        ofl->ofl_phdr[0].p_memsz = 0;
4041                                        ofl->ofl_phdr[0].p_flags = 0;
4042                                }
4043                        }
4044                }

4046                /*
4047                 * Ensure the ELF entry point defaults to zero.  Typically, this
4048                 * value is overridden in update_oehdr() to one of the standard
4049                 * entry points.  Historically, this default was set to the
4050                 * address of first executable section, but this has since been
4051                 * found to be more confusing than it is helpful.
4052                 */
4053                ehdr->e_entry = 0;

4055                DBG_CALL(Dbg_seg_entry(ofl, segndx, sgp));

4057                /*
4058                 * Traverse the output section descriptors for this segment so
4059                 * that we can update the section headers addresses.  We've
4060                 * calculated the virtual address of the initial section within
4061                 * this segment, so each successive section can be calculated
4062                 * based on their offsets from each other.
4063                 */
4064                secndx = 0;
4065                hshdr = 0;
4066                for (APLIST_TRAVERSE(sgp->sg_osdescs, idx2, osp)) {
4067                        Shdr    *shdr = osp->os_shdr;

4069                        if (shdr->sh_link)
4070                                shdr->sh_link = translate_link(ofl, osp,
4071                                    shdr->sh_link, MSG_INTL(MSG_FIL_INVSHLINK));

4073                        if (shdr->sh_info && (shdr->sh_flags & SHF_INFO_LINK))
4074                                shdr->sh_info = translate_link(ofl, osp,
4075                                    shdr->sh_info, MSG_INTL(MSG_FIL_INVSHINFO));

4077                        if (!(flags & FLG_OF_RELOBJ) &&
4078                            (phdr->p_type == PT_LOAD)) {
4079                                if (hshdr)
4080                                        vaddr += (shdr->sh_offset -
4081                                            hshdr->sh_offset);

4083                                shdr->sh_addr = vaddr;
4084                                hshdr = shdr;
4085                        }

4087                        DBG_CALL(Dbg_seg_os(ofl, osp, secndx));
4088                        secndx++;
4089                }

4091                /*
4092                 * Establish the virtual address of the end of the last section
4093                 * in this segment so that the next segments offset can be
```



```
4094                 * calculated from this.
4095                 */
4096                if (hshdr)
4097                        vaddr += hshdr->sh_size;

4099                /*
4100                 * Output sections for this segment complete.  Adjust the
4101                 * virtual offset for the last sections size, and make sure we
4102                 * haven't exceeded any maximum segment length specification.
4103                 */
4104                if ((sgp->sg_length != 0) && (sgp->sg_length < phdr->p_memsz)) {
4105                        ld_eprintf(ofl, ERR_FATAL, MSG_INTL(MSG_UPD_LARGSIZE),
4106                            ofl->ofl_name, sgp->sg_name,
4107                            EC_XWORD(phdr->p_memsz), EC_XWORD(sgp->sg_length));
4108                        return (S_ERROR);
4109                }

4111                if (phdr->p_type == PT_NOTE) {
4112                        phdr->p_vaddr = 0;
4113                        phdr->p_paddr = 0;
4114                        phdr->p_align = 0;
4115                        phdr->p_memsz = 0;
4116                }

4118                if ((phdr->p_type != PT_NULL) && !(flags & FLG_OF_RELOBJ))
4119                        ofl->ofl_phdr[phdrndx++] = *phdr;
4120        }

4122        /*
4123         * Update any new output sections.  When building the initial output
4124         * image, a number of sections were created but left uninitialized (eg.
4125         * .dynsym, .dynstr, .symtab, .symtab, etc.).  Here we update these
4126         * sections with the appropriate data.  Other sections may still be
4127         * modified via reloc_process().
4128         *
4129         * Copy the interpreter name into the .interp section.
4130         */
4131        if (ofl->ofl_interp)
4132                (void) strcpy((char *)ofl->ofl_osinterp->os_outdata->d_buf,
4133                    ofl->ofl_interp);

4135        /*
4136         * Update the .shstrtab, .strtab and .dynstr sections.
4137         */
4138        update_ostrtab(ofl->ofl_osshstrtab, ofl->ofl_shdrsttab, 0);
4139        update_ostrtab(ofl->ofl_osstrtab, ofl->ofl_strtab, 0);
4140        update_ostrtab(ofl->ofl_osdynstr, ofl->ofl_dynstrtab, DYNSTR_EXTRA_PAD);

4142        /*
4143         * Build any output symbol tables, the symbols information is copied
4144         * and updated into the new output image.
4145         */
4146        if ((etext = update_osym(ofl)) == (Addr)S_ERROR)
4147                return (S_ERROR);

4149        /*
4150         * If we have an PT_INTERP phdr, update it now from the associated
4151         * section information.
4152         */
4153        if (intpsgp) {
4154                Phdr    *phdr = &(intpsgp->sg_phdr);
4155                Shdr    *shdr = ofl->ofl_osinterp->os_shdr;

4157                phdr->p_vaddr = shdr->sh_addr;
4158                phdr->p_offset = shdr->sh_offset;
4159                phdr->p_memsz = phdr->p_filesz = shdr->sh_size;
```

```
4160                 phdr->p_flags = PF_R;
4161
4162                 DBG_CALL(Dbg_seg_entry(ofl, intpsndx, intpsgp));
4163                 ofl->ofl_phdr[intppndx] = *phdr;
4164         }
4165
4166         /*
4167          * If we have a PT_SUNWDTRACE phdr, update it now with the address of
4168          * the symbol.  It's only now been updated via update_sym().
4169          */
4170         if (dtracesgp) {
4171                 Phdr            *aphdr, *phdr = &(dtracesgp->sg_phdr);
4172                 Sym_desc        *sdp = ofl->ofl_dtracesym;
4173
4174                 phdr->p_vaddr = sdp->sd_sym->st_value;
4175                 phdr->p_memsz = sdp->sd_sym->st_size;
4176
4177                 /*
4178                  * Take permissions from the segment that the symbol is
4179                  * associated with.
4180                  */
4181                 aphdr = &sdp->sd_isc->is_osdesc->os_sgdesc->sg_phdr;
4182                 assert(aphdr);
4183                 phdr->p_flags = aphdr->p_flags;
4184
4185                 DBG_CALL(Dbg_seg_entry(ofl, dtracesndx, dtracesgp));
4186                 ofl->ofl_phdr[dtracepndx] = *phdr;
4187         }
4188
4189         /*
4190          * If we have a PT_SUNWCAP phdr, update it now from the associated
4191          * section information.
4192          */
4193         if (capsgp) {
4194                 Phdr    *phdr = &(capsgp->sg_phdr);
4195                 Shdr    *shdr = ofl->ofl_oscap->os_shdr;
4196
4197                 phdr->p_vaddr = shdr->sh_addr;
4198                 phdr->p_offset = shdr->sh_offset;
4199                 phdr->p_memsz = phdr->p_filesz = shdr->sh_size;
4200                 phdr->p_flags = PF_R;
4201
4202                 DBG_CALL(Dbg_seg_entry(ofl, capsndx, capsgp));
4203                 ofl->ofl_phdr[cappndx] = *phdr;
4204         }
4205
4206         /*
4207          * Update the GROUP sections.
4208          */
4209         if (update_ogroup(ofl) == S_ERROR)
4210                 return (S_ERROR);
4211
4212         /*
4213          * Update Move Table.
4214          */
4215         if (ofl->ofl_osmove || ofl->ofl_isparexpn)
4216                 update_move(ofl);
4217
4218         /*
4219          * Build any output headers, version information, dynamic structure and
4220          * syminfo structure.
4221          */
4222         if (update_oehdr(ofl) == S_ERROR)
4223                 return (S_ERROR);
4224         if (!(flags & FLG_OF_NOVERSEC)) {
4225                 if ((flags & FLG_OF_VERDEF) &&
```



```
4226                     (update_overdef(ofl) == S_ERROR))
4227                         return (S_ERROR);
4228                 if ((flags & FLG_OF_VERNEED) &&
4229                     (update_overneed(ofl) == S_ERROR))
4230                         return (S_ERROR);
4231                 if (flags & (FLG_OF_VERNEED | FLG_OF_VERDEF))
4232                         update_oversym(ofl);
4233         }
4234         if (flags & FLG_OF_DYNAMIC) {
4235                 if (update_odynamic(ofl) == S_ERROR)
4236                         return (S_ERROR);
4237         }
4238         if (ofl->ofl_ossyminfo) {
4239                 if (update_osyminfo(ofl) == S_ERROR)
4240                         return (S_ERROR);
4241         }
4242
4243         /*
4244          * Update capabilities information if required.
4245          */
4246         if (ofl->ofl_oscap)
4247                 update_oscap(ofl);
4248         if (ofl->ofl_oscapinfo)
4249                 update_oscapinfo(ofl);
4250
4251         /*
4252          * Sanity test: the first and last data byte of a string table
4253          * must be NULL.
4254          */
4255         assert((ofl->ofl_osshstrtab == NULL) ||
4256             (*((char *)ofl->ofl_osshstrtab->os_outdata->d_buf) == '\0'));
4257         assert((ofl->ofl_osshstrtab == NULL) ||
4258             (*(((char *)ofl->ofl_osshstrtab->os_outdata->d_buf) +
4259             ofl->ofl_osshstrtab->os_outdata->d_size - 1) == '\0'));
4260
4261         assert((ofl->ofl_osstrtab == NULL) ||
4262             (*((char *)ofl->ofl_osstrtab->os_outdata->d_buf) == '\0'));
4263         assert((ofl->ofl_osstrtab == NULL) ||
4264             (*(((char *)ofl->ofl_osstrtab->os_outdata->d_buf) +
4265             ofl->ofl_osstrtab->os_outdata->d_size - 1) == '\0'));
4266
4267         assert((ofl->ofl_osdynstr == NULL) ||
4268             (*((char *)ofl->ofl_osdynstr->os_outdata->d_buf) == '\0'));
4269         assert((ofl->ofl_osdynstr == NULL) ||
4270             (*(((char *)ofl->ofl_osdynstr->os_outdata->d_buf) +
4271             ofl->ofl_osdynstr->os_outdata->d_size - DYNSTR_EXTRA_PAD - 1) ==
4272             '\0'));
4273
4274         /*
4275          * Emit Strtab diagnostics.
4276          */
4277         DBG_CALL(Dbg_sec_strtab(ofl->ofl_lml, ofl->ofl_osshstrtab,
4278             ofl->ofl_shdrsttab));
4279         DBG_CALL(Dbg_sec_strtab(ofl->ofl_lml, ofl->ofl_osstrtab,
4280             ofl->ofl_strtab));
4281         DBG_CALL(Dbg_sec_strtab(ofl->ofl_lml, ofl->ofl_osdynstr,
4282             ofl->ofl_dynstrtab));
4283
4284         /*
4285          * Initialize the section headers string table index within the elf
4286          * header.
4287          */
4288         /* LINTED */
4289         if ((shscnndx = elf_ndxscn(ofl->ofl_osshstrtab->os_scn)) <
4290             SHN_LORESERVE) {
4291                 ofl->ofl_nehdr->e_shstrndx =
```

```
4292                    /* LINTED */
4293                    (Half)shscnndx;
4294            } else {
4295                    /*
4296                     * If the STRTAB section index doesn't fit into
4297                     * e_shstrndx, then we store it in 'shdr[0].st_link'.
4298                     */
4299                    Elf_Scn *scn;
4300                    Shdr    *shdr0;

4302                    if ((scn = elf_getscn(ofl->ofl_elf, 0)) == NULL) {
4303                            ld_eprintf(ofl, ERR_ELF, MSG_INTL(MSG_ELF_GETSCN),
4304                                ofl->ofl_name);
4305                            return (S_ERROR);
4306                    }
4307                    if ((shdr0 = elf_getshdr(scn)) == NULL) {
4308                            ld_eprintf(ofl, ERR_ELF, MSG_INTL(MSG_ELF_GETSHDR),
4309                                ofl->ofl_name);
4310                            return (S_ERROR);
4311                    }
4312                    ofl->ofl_nehdr->e_shstrndx = SHN_XINDEX;
4313                    shdr0->sh_link = shscnndx;
4314            }

4316            return ((uintptr_t)etext);
4317    }
```

```
**********************************************************
    3349 Wed May 27 19:49:06 2015
new/usr/src/cmd/svc/milestone/Makefile
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
```

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
#
# Copyright (c) 2004, 2010, Oracle and/or its affiliates. All rights reserved.
#

include ../../Makefile.cmd

FILEMODE = 0444

BUILTXML= \
        console-login.xml

FSSVCS= \
        local-fs.xml \
        minimal-fs.xml \
        root-fs.xml \
        usr-fs.xml

FSMANIFESTS= $(FSSVCS:%=$(ROOTSVCSYSTEMFILESYSTEM)/%)

NETSVCS= \
        network-initial.xml \
        network-install.xml \
        network-iptun.xml \
        network-ipqos.xml \
        network-location.xml \
        network-loopback.xml \
        network-netcfg.xml \
        network-netmask.xml \
        network-netcfg.xml \
        network-physical.xml \
        network-routing-setup.xml \
        network-service.xml

NETMANIFESTS= $(NETSVCS:%=$(ROOTSVCNETWORK)/%)

MAINMILESTONES= \
        multi-user-server.xml \
        multi-user.xml \
```



```
        name-services.xml \
        network.xml \
        single-user.xml \
        sysconfig.xml

MAINMANIFESTS= $(MAINMILESTONES:%=$(ROOTSVCMILESTONE)/%)

SYSDEVSVCS= \
        devices-local.xml \
        devices-audio.xml

SYSDEVMANIFESTS= $(SYSDEVSVCS:%=$(ROOTSVCSYSTEMDEVICE)/%)

SYSTEMSVCS= \
        boot-archive.xml \
        console-login.xml \
        early-manifest-import.xml \
        identity.xml \
        manifest-import.xml \
        process-security.xml \
#endif /* ! codereview */
        rmtmpfiles.xml  \
        vtdaemon.xml

SYSTEMMANIFESTS = $(SYSTEMSVCS:%=$(ROOTSVCSYSTEM)/%)

SYSTEMSVCSVCS= \
        restarter.xml \
        global.xml

SYSTEMSVCMANIFESTS= $(SYSTEMSVCSVCS:%=$(ROOTSVCSYSTEM)/svc/%)

MISCFILES= \
        README.share

SYSTEMMISCFILES = $(MISCFILES:%.share=$(ROOT)/lib/svc/share/%)

#
# MANIFEST is used solely in the construction of the check target.
#
MANIFEST= $(FSSVCS) $(NETSVCS) $(MAINMILESTONES) $(SYSTEMSVCS) \
        $(SYSDEVSVCS) $(SYSTEMSVCSVCS)

SVCMETHOD=\
        boot-archive \
        console-login \
        devices-audio \
        devices-local \
        fs-local \
        fs-minimal \
        fs-root \
        fs-usr \
        identity-domain \
        identity-node \
        manifest-import \
        net-loc \
        net-loopback \
        net-init \
        net-install \
        net-iptun \
        net-ipqos \
        net-netmask \
        net-nwam \
        net-physical \
        net-routing-setup \
        net-svc \
```

```
125         rmtmpfiles \
126         vtdaemon

128 $(ROOTSVCMETHOD) := FILEMODE = 0555

130 all: $(BUILTXML)

132 install: $(FSMANIFESTS) $(MAINMANIFESTS) $(NETMANIFESTS) $(SYSTEMMANIFESTS) \
133         $(ROOTSVCMETHOD) $(SYSDEVMANIFESTS) $(SYSTEMSVCMANIFESTS) \
134         $(SYSTEMMISCFILES)

136 check:  $(CHKMANIFEST)

138 console-login.xml: make-console-login-xml
139         $(SH) ./make-console-login-xml

141 clobber: clean
142         -$(RM) $(BUILTXML)

144 $(ROOTSVCMILESTONE)/%: %
145         $(INS.file)

147 $(ROOTSVCNETWORK)/%: %
148         $(INS.file)

150 $(ROOTSVCSYSTEM)/%: %
151         $(INS.file)

153 $(ROOTSVCSYSTEMDEVICE)/%: %
154         $(INS.file)

156 $(ROOTSVCSYSTEMFILESYSTEM)/%: %
157         $(INS.file)

159 $(ROOTSVCSYSTEM)/svc/%: %
160         $(INS.file)

162 $(ROOT)/lib/svc/share/%: %.share
163         $(INS.rename)

165 clean lint _msg:
```

```
**********************************************************
    2208 Wed May 27 19:49:07 2015
new/usr/src/cmd/svc/milestone/process-security.xml
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
```

```
<?xml version='1.0'?>
<!DOCTYPE service_bundle SYSTEM '/usr/share/lib/xml/dtd/service_bundle.dtd.1'>

<!--
 CDDL HEADER START

 This file and its contents are supplied under the terms of the
 Common Development and Distribution License ("CDDL"), version 1.0.
 You may only use this file in accordance with the terms of version
 1.0 of the CDDL.

 A full copy of the text of the CDDL should have accompanied this
 source.  A copy of the CDDL is also available via the Internet at
 http://www.illumos.org/license/CDDL.

 CDDL HEADER END

        NOTE:  This service manifest is not editable; its contents will
        be overwritten by package or patch operations, including
        operating system upgrade.  Make customizations in a different
        file.
-->

<service_bundle type="manifest" name="process-security">
        <service name="system/process-security" type="service" version="1">
                <!-- Initial state of the service is disabled -->
                <create_default_instance enabled="false" />

                <single_instance />

                <!-- We don't actually have any methods, but we create a
                     default instance so that we show up in svcs -a -->

                <exec_method type="method" name="start" exec=":true" timeout_sec
                <exec_method type="method" name="stop" exec=":true" timeout_seco

                <property_group name='startd' type='framework'>
                  <propval name='duration' type='astring' value='transient' />
                </property_group>

                <property_group name='secflags' type='application'>
                  <propval name='aslr' type='boolean' value='false' />

                  <propval name='value_authorization' type='astring'
                           value='solaris.smf.value.process-security' />
                </property_group>

                <stability value="Unstable" />

                <template>
                        <common_name>
                                <loctext xml:lang='C'>Security Flag Configuratio
                        </common_name>
                        <documentation>
                                <manpage title='security-flags' section='5'
                                         manpath='/usr/share/man' />
                                <manpage title='psecflags' section='1'
                                         manpath='/usr/share/man' />
```



```
                        </documentation>
                </template>
        </service>
</service_bundle>
#endif /* ! codereview */
```

```
**********************************************************
   33938 Wed May 27 19:49:07 2015
new/usr/src/cmd/svc/startd/startd.c
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright (c) 2004, 2010, Oracle and/or its affiliates. All rights reserved.
 * Copyright 2015, Joyent, Inc.
 */

/*
 * startd.c - the master restarter
 *
 * svc.startd comprises two halves.  The graph engine is based in graph.c and
 * maintains the service dependency graph based on the information in the
 * repository.  For each service it also tracks the current state and the
 * restarter responsible for the service.  Based on the graph, events from the
 * repository (mostly administrative requests from svcadm), and messages from
 * the restarters, the graph engine makes decisions about how the services
 * should be manipulated and sends commands to the appropriate restarters.
 * Communication between the graph engine and the restarters is embodied in
 * protocol.c.
 *
 * The second half of svc.startd is the restarter for services managed by
 * svc.startd and is primarily contained in restarter.c.  It responds to graph
 * engine commands by executing methods, updating the repository, and sending
 * feedback (mostly state updates) to the graph engine.
 *
 * Overview of the SMF Architecture
 *
 * There are a few different components that make up SMF and are responsible
 * for different pieces of functionality that are used:
 *
 * svc.startd(1M): A daemon that is in charge of starting, stopping, and
 *     restarting services and instances.
 * svc.configd(1M): A daemon that manages the repository that stores
 *     information, property groups, and state of the different services and
 *     instances.
 * libscf(3LIB): A C library that provides the glue for communicating,
 *     accessing, and updating information about services and instances.
 * svccfg(1M): A utility to add and remove services as well as change the
 *     properties associated with different services and instances.
```



```
 * svcadm(1M): A utility to control the different instance of a service. You
 *     can use this to enable and disable them among some other useful things.
 * svcs(1): A utility that reports on the status of various services on the
 *     system.
 *
 * The following block diagram explains how these components communicate:
 *
 * The SMF Block Diagram
 *                                                          Repository
 *   This attempts to show       +---------+              +--------+
 *   the relations between       |         |     SQL      |        |
 *   the different pieces        | configd |<------------>| SQLite |
 *   that make SMF work and      |         | Transaction  |        |
 *   users/administrators        +---------+              +--------+
 *   call into.                   ^      ^
 *                                |      |
 *                   door_call(3C)|      | door_call(3C)
 *                                |      |
 *                                v      v
 *      +----------+     +--------+      +--------+      +----------+
 *      |          |     |        |      |        |      |  svccfg  |
 *      |  startd  |<--->| libscf |      | libscf |<---->|  svcadm  |
 *      |          |     | (3LIB) |      | (3LIB) |      |   svcs   |
 *      +----------+     +--------+      +--------+      +----------+
 *        ^      ^
 *        |      | fork(2)/exec(2)
 *        |      | libcontract(3LIB)
 *        v      v                            Various System/User services
 *       +-------------------------------------------------------------------+
 *       | system/filesystem/local:default      system/coreadm:default       |
 *       | network/loopback:default             system/zones:default         |
 *       | milestone/multi-user:default         system/cron:default          |
 *       | system/console-login:default         network/ssh:default          |
 *       | system/pfexec:default                system/svc/restarter:default |
 *       +-------------------------------------------------------------------+
 *
 * Chatting with Configd and Sharing Repository Information
 *
 * As you run commands with svcs, svccfg, and svcadm, they are all creating a
 * libscf handle to communicate with configd. As calls are made via libscf they
 * ultimately go and talk to configd to get information. However, how we
 * actually are talking to configd is not as straightforward as it appears.
 *
 * When configd starts up it creates a door located at
 * /etc/svc/volatile/repository_door. This door runs the routine called
 * main_switcher() from usr/src/cmd/svc/configd/maindoor.c. When you first
 * invoke svc(cfg|s|adm), one of the first things that occurs is creating a
 * scf_handle_t and binding it to configd by calling scf_handle_bind(). This
 * function makes a door call to configd and gets returned a new file
 * descriptor. This file descriptor is itself another door which calls into
 * configd's client_switcher(). This is the door that is actually used when
 * getting and fetching properties, and many other useful things.
 *
 * svc.startd needs a way to notice the changes that occur to the repository.
 * For example, if you enabled a service that was not previously running, it's
 * up to startd to notice that this has happened, check dependencies, and
 * eventually start up the service. The way it gets these notifications is via
 * a thread who's sole purpose in life is to call _scf_notify_wait(). This
 * function acts like poll(2) but for changes that occur in the repository.
 * Once this thread gets the event, it dispatches the event appropriately.
 *
 * The Events of svc.startd
 *
 * svc.startd has to handle a lot of complexity. Understanding how you go from
 * getting the notification that a service was enabled to actually enabling it
 * is not obvious from a cursory glance. The first thing to keep in mind is
```

```
125  * that startd maintains a graph of all the related services and instances so
126  * it can keep track of what is enabled, what dependencies exist, etc. all so
127  * that it can answer the question of what is affected by a change. Internally
128  * there are a lot of different queues for events, threads to process these
129  * queues, and different paths to have events enter these queues. What follows
130  * is a diagram that attempts to explain some of those paths, though it's
131  * important to note that for some of these pieces, such as the graph and
132  * vertex events, there are many additional ways and code paths these threads
133  * and functions can take. And yes, restarter_event_enqueue() is not the same
134  * thing as restarter_queue_event().
135  *
136  *   Threads/Functions                 Queues                  Threads/Functions
137  *
138  * called by various
139  *     +----------------+            +-------+                  +-------------+
140  * --->| graph_protocol | graph_event | graph |   graph_event_   | graph_event |
141  * --->| _send_event()  |------------>| event |----------------->| _thread     |
142  *     +----------------+ _enqueue()  | queue |   dequeue()      +-------------+
143  *                                    +-------+                        |
144  *  _scf_notify_wait()                                vertex_send_event()|
145  *  |                                                                    v
146  *  |  +------------------+                                  +--------------------+
147  *  +->| repository_event | vertex_send_event()              | restarter_protocol |
148  *     | _thread          |----------------------------------->| _send_event()      |
149  *     +------------------+                                  +--------------------+
150  *                                                             |    | out to other
151  *                restarter_                        restarter_ |    | restarters
152  *                event_dequeue() +-----------+  event_     |    | not startd
153  *               +----------------| restarter |<------------+    +------------->
154  *               v                |   event   |  enqueue()
155  *      +-----------------+       |   queue   |             +------------------>
156  *      | restarter_event |       +-----------+             |+----------------->
157  *      | _thread         |                                 ||+---------------->
158  *      +-----------------+                                 ||| start/stop inst
159  *               |               +--------------+       +--------------------+
160  *               |               |   instance   |       | restarter_process_ |
161  *               +-------------->|    event     |------>| events             |
162  *                restarter_     |    queue     |       | per-instance lwp   |
163  *                queue_event()  +--------------+       +--------------------+
164  *                                                          ||| various funcs
165  *                                                          ||| controlling
166  *                                                          ||| instance state
167  *                                                          ||+---------------->
168  *                                                          |+----------------->
169  *                                                          +------------------>
170  *
171  * What's important to take away is that there is a queue for each instance on
172  * the system that handles events related to dealing directly with that
173  * instance and that events can be added to it because of changes to properties
174  * that are made to configd and acted upon asynchronously by startd.
175  *
176  * Error handling
177  *
178  * In general, when svc.startd runs out of memory it reattempts a few times,
179  * sleeping inbetween, before giving up and exiting (see startd_alloc_retry()).
180  * When a repository connection is broken (libscf calls fail with
181  * SCF_ERROR_CONNECTION_BROKEN, librestart and internal functions return
182  * ECONNABORTED), svc.startd calls libscf_rebind_handle(), which coordinates
183  * with the svc.configd-restarting thread, fork_configd_thread(), via
184  * st->st_configd_live_cv, and rebinds the repository handle.  Doing so resets
185  * all libscf state associated with that handle, so functions which do this
186  * should communicate the event to their callers (usually by returning
187  * ECONNRESET) so they may reset their state appropriately.
188  *
189  * External references
190  *
```



```
191  * svc.configd generates special security audit events for changes to some
192  * restarter related properties.  See the special_props_list array in
193  * usr/src/cmd/svc/configd/rc_node.c for the properties that cause these audit
194  * events.  If you change the semantics of these propereties within startd, you
195  * will probably need to update rc_node.c
196  */

198 #include <stdio.h>
199 #include <stdio_ext.h>
200 #include <sys/mnttab.h>         /* uses FILE * without including stdio.h */
201 #include <alloca.h>
202 #include <sys/mount.h>
203 #include <sys/stat.h>
204 #include <sys/types.h>
205 #include <sys/wait.h>
206 #include <sys/proc.h>
207 #endif /* ! codereview */
208 #include <assert.h>
209 #include <errno.h>
210 #include <fcntl.h>
211 #include <ftw.h>
212 #include <libintl.h>
213 #include <libscf.h>
214 #include <libscf_priv.h>
215 #include <libuutil.h>
216 #include <locale.h>
217 #include <poll.h>
218 #include <pthread.h>
219 #include <signal.h>
220 #include <stdarg.h>
221 #include <stdlib.h>
222 #include <string.h>
223 #include <strings.h>
224 #include <unistd.h>

226 #include "startd.h"
227 #include "protocol.h"

229 extern int psecflags(idtype_t, id_t, psecflags_cmd_t, uint_t);

231 #endif /* ! codereview */
232 ssize_t max_scf_name_size;
233 ssize_t max_scf_fmri_size;
234 ssize_t max_scf_value_size;

236 mode_t fmask;
237 mode_t dmask;

239 graph_update_t *gu;
240 restarter_update_t *ru;

242 startd_state_t *st;

244 boolean_t booting_to_single_user = B_FALSE;

246 const char * const admin_actions[] = {
247     SCF_PROPERTY_DEGRADED,
248     SCF_PROPERTY_MAINT_OFF,
249     SCF_PROPERTY_MAINT_ON,
250     SCF_PROPERTY_MAINT_ON_IMMEDIATE,
251     SCF_PROPERTY_REFRESH,
252     SCF_PROPERTY_RESTART
253 };

255 const int admin_events[NACTIONS] = {
256     RESTARTER_EVENT_TYPE_ADMIN_DEGRADED,
```

```
257     RESTARTER_EVENT_TYPE_ADMIN_MAINT_OFF,
258     RESTARTER_EVENT_TYPE_ADMIN_MAINT_ON,
259     RESTARTER_EVENT_TYPE_ADMIN_MAINT_ON_IMMEDIATE,
260     RESTARTER_EVENT_TYPE_ADMIN_REFRESH,
261     RESTARTER_EVENT_TYPE_ADMIN_RESTART
262 };

264 const char * const instance_state_str[] = {
265         "none",
266         "uninitialized",
267         "maintenance",
268         "offline",
269         "disabled",
270         "online",
271         "degraded"
272 };

274 static int finished = 0;
275 static int opt_reconfig = 0;
276 static uint8_t prop_reconfig = 0;

278 #define INITIAL_REBIND_ATTEMPTS 5
279 #define INITIAL_REBIND_DELAY    3

281 pthread_mutexattr_t mutex_attrs;

283 #ifdef DEBUG
284 const char *
285 _umem_debug_init(void)
286 {
287         return ("default,verbose");     /* UMEM_DEBUG setting */
288 }

290 const char *
291 _umem_logging_init(void)
292 {
293         return ("fail,contents");       /* UMEM_LOGGING setting */
294 }
295 #endif

297 const char *
298 _umem_options_init(void)
299 {
300         /*
301          * To reduce our memory footprint, we set our UMEM_OPTIONS to indicate
302          * that we do not wish to have per-CPU magazines -- if svc.startd is so
303          * hot on CPU such that this becomes a scalability problem, there are
304          * likely deeper things amiss...
305          */
306         return ("nomagazines");         /* UMEM_OPTIONS setting */
307 }

309 /*
310  * startd_alloc_retry()
311  *   Wrapper for allocation functions.  Retries with a decaying time
312  *   value on failure to allocate, and aborts startd if failure is
313  *   persistent.
314  */
315 void *
316 startd_alloc_retry(void *f(size_t, int), size_t sz)
317 {
318         void *p;
319         uint_t try, msecs;

321         p = f(sz, UMEM_DEFAULT);
322         if (p != NULL || sz == 0)
```



```
323                 return (p);

325         msecs = ALLOC_DELAY;

327         for (try = 0; p == NULL && try < ALLOC_RETRY; ++try) {
328                 (void) poll(NULL, 0, msecs);
329                 msecs *= ALLOC_DELAY_MULT;
330                 p = f(sz, UMEM_DEFAULT);
331                 if (p != NULL)
332                         return (p);
333         }

335         uu_die("Insufficient memory.\n");
336         /* NOTREACHED */
337 }

339 void *
340 safe_realloc(void *p, size_t sz)
341 {
342         uint_t try, msecs;

344         p = realloc(p, sz);
345         if (p != NULL || sz == 0)
346                 return (p);

348         msecs = ALLOC_DELAY;

350         for (try = 0; errno == EAGAIN && try < ALLOC_RETRY; ++try) {
351                 (void) poll(NULL, 0, msecs);
352                 p = realloc(p, sz);
353                 if (p != NULL)
354                         return (p);
355                 msecs *= ALLOC_DELAY_MULT;
356         }

358         uu_die("Insufficient memory.\n");
359         /* NOTREACHED */
360 }

362 char *
363 safe_strdup(const char *s)
364 {
365         uint_t try, msecs;
366         char *d;

368         d = strdup(s);
369         if (d != NULL)
370                 return (d);

372         msecs = ALLOC_DELAY;

374         for (try = 0;
375             (errno == EAGAIN || errno == ENOMEM) && try < ALLOC_RETRY;
376             ++try) {
377                 (void) poll(NULL, 0, msecs);
378                 d = strdup(s);
379                 if (d != NULL)
380                         return (d);
381                 msecs *= ALLOC_DELAY_MULT;
382         }

384         uu_die("Insufficient memory.\n");
385         /* NOTREACHED */
386 }
```

```
389 void
390 startd_free(void *p, size_t sz)
391 {
392         umem_free(p, sz);
393 }

395 /*
396  * Creates a uu_list_pool_t with the same retry policy as startd_alloc().
397  * Only returns NULL for UU_ERROR_UNKNOWN_FLAG and UU_ERROR_NOT_SUPPORTED.
398  */
399 uu_list_pool_t *
400 startd_list_pool_create(const char *name, size_t e, size_t o,
401     uu_compare_fn_t *f, uint32_t flags)
402 {
403         uu_list_pool_t *pool;
404         uint_t try, msecs;

406         pool = uu_list_pool_create(name, e, o, f, flags);
407         if (pool != NULL)
408                 return (pool);

410         msecs = ALLOC_DELAY;

412         for (try = 0; uu_error() == UU_ERROR_NO_MEMORY && try < ALLOC_RETRY;
413             ++try) {
414                 (void) poll(NULL, 0, msecs);
415                 pool = uu_list_pool_create(name, e, o, f, flags);
416                 if (pool != NULL)
417                         return (pool);
418                 msecs *= ALLOC_DELAY_MULT;
419         }

421         if (try < ALLOC_RETRY)
422                 return (NULL);

424         uu_die("Insufficient memory.\n");
425         /* NOTREACHED */
426 }

428 /*
429  * Creates a uu_list_t with the same retry policy as startd_alloc().  Only
430  * returns NULL for UU_ERROR_UNKNOWN_FLAG and UU_ERROR_NOT_SUPPORTED.
431  */
432 uu_list_t *
433 startd_list_create(uu_list_pool_t *pool, void *parent, uint32_t flags)
434 {
435         uu_list_t *list;
436         uint_t try, msecs;

438         list = uu_list_create(pool, parent, flags);
439         if (list != NULL)
440                 return (list);

442         msecs = ALLOC_DELAY;

444         for (try = 0; uu_error() == UU_ERROR_NO_MEMORY && try < ALLOC_RETRY;
445             ++try) {
446                 (void) poll(NULL, 0, msecs);
447                 list = uu_list_create(pool, parent, flags);
448                 if (list != NULL)
449                         return (list);
450                 msecs *= ALLOC_DELAY_MULT;
451         }

453         if (try < ALLOC_RETRY)
454                 return (NULL);
```



```
456         uu_die("Insufficient memory.\n");
457         /* NOTREACHED */
458 }

460 pthread_t
461 startd_thread_create(void *(*func)(void *), void *ptr)
462 {
463         int err;
464         pthread_t tid;

466         err = pthread_create(&tid, NULL, func, ptr);
467         if (err != 0) {
468                 assert(err == EAGAIN);
469                 uu_die("Could not create thread.\n");
470         }

472         err = pthread_detach(tid);
473         assert(err == 0);

475         return (tid);
476 }

478 extern int info_events_all;

480 struct psf_desc {
481         char *name;
482         uint_t flag;
483 } procsec_flag_tbl[] = {
484         { "aslr",       PROC_SEC_ASLR },
485         { NULL, NULL }
486 };

488 static void
489 init_secflags(scf_handle_t *hndl)
490 {
491         scf_property_t *prop;
492         scf_value_t *val;
493         struct psf_desc *psfd = NULL;
494         char *proc_sec_fmri = "svc:/system/process-security/"
495             ":properties/secflags";

497         for (psfd = procsec_flag_tbl; psfd->name != NULL; psfd++) {
498                 char *pfmri;
499                 uint8_t flag;

501                 prop = safe_scf_property_create(hndl);
502                 val = safe_scf_value_create(hndl);

504                 if ((pfmri = uu_msprintf("%s/%s", proc_sec_fmri, psfd->name)) ==
505                         uu_die("Allocation failure\n");

507                 if (scf_handle_decode_fmri(hndl, pfmri,
508                     NULL, NULL, NULL, NULL, prop, NULL) != 0)
509                         goto next;

511                 if (scf_property_get_value(prop, val) != 0)
512                         goto next;

514                 (void) scf_value_get_boolean(val, &flag);

516                 /*
517                  * XXX: This will fail if the zone had PRIV_PROC_SECFLAGS
518                  * removed.
519                  *
520                  * I'm not sure what we should do in that case -- I'd still
```

```
521                  * like this to be settable based on a zonecfg setting, too.
522                  *
523                  * We only set things explicitly _on_ here, rather than
524                  * explicitly _off_ such that a zone's settings do not
525                  * permanently override those from the GZ.
526                  *
527                  * XXX: This might be a bit crap, we sorta want a tri-state
528                  */
529                 if (flag != 0) {
530                         if (psecflags(P_PID, P_MYID,
531                             PSECFLAGS_ENABLE, psfd->flag) != 0) {
532                                 uu_warn("couldn't set security flags: %s\n",
533                                     strerror(errno));
534                         }
535                 }
536 next:
537                 uu_free(pfmri);
538                 scf_value_destroy(val);
539                 scf_property_destroy(prop);
540         }
541 }

543 #endif /* ! codereview */
544 static int
545 read_startd_config(void)
546 {
547         scf_handle_t *hndl;
548         scf_instance_t *inst;
549         scf_propertygroup_t *pg;
550         scf_property_t *prop;
551         scf_value_t *val;
552         scf_iter_t *iter, *piter;
553         instance_data_t idata;
554         char *buf, *vbuf;
555         char *startd_options_fmri = uu_msprintf("%s/:properties/options",
556             SCF_SERVICE_STARTD);
557         char *startd_reconfigure_fmri = uu_msprintf(
558             "%s/:properties/system/reconfigure", SCF_SERVICE_STARTD);
559         char *env_opts, *lasts, *cp;
560         int bind_fails = 0;
561         int ret = 0, r;
562         uint_t count = 0, msecs = ALLOC_DELAY;
563         size_t sz;
564         ctid_t ctid;
565         uint64_t uint64;

567         buf = startd_alloc(max_scf_fmri_size);

569         if (startd_options_fmri == NULL || startd_reconfigure_fmri == NULL)
570                 uu_die("Allocation failure\n");

572         st->st_log_prefix = LOG_PREFIX_EARLY;

574         if ((st->st_log_file = getenv("STARTD_DEFAULT_LOG")) == NULL) {
575                 st->st_log_file = startd_alloc(strlen(STARTD_DEFAULT_LOG) + 1);

577                 (void) strcpy(st->st_log_file, STARTD_DEFAULT_LOG);
578         }

580         st->st_door_path = getenv("STARTD_ALT_DOOR");

582         /*
583          * Read "options" property group.
584          */
585         for (hndl = libscf_handle_create_bound(SCF_VERSION); hndl == NULL;
586             hndl = libscf_handle_create_bound(SCF_VERSION), bind_fails++) {
```



```
587                 (void) sleep(INITIAL_REBIND_DELAY);

589                 if (bind_fails > INITIAL_REBIND_ATTEMPTS) {
590                         /*
591                          * In the case that we can't bind to the repository
592                          * (which should have been started), we need to allow
593                          * the user into maintenance mode to determine what's
594                          * failed.
595                          */
596                         log_framework(LOG_INFO, "Couldn't fetch "
597                             "default settings: %s\n",
598                             scf_strerror(scf_error()));

600                         ret = -1;

602                         goto noscfout;
603                 }
604         }

606         idata.i_fmri = SCF_SERVICE_STARTD;
607         idata.i_state = RESTARTER_STATE_NONE;
608         idata.i_next_state = RESTARTER_STATE_NONE;
609 timestamp:
610         switch (r = _restarter_commit_states(hndl, &idata,
611             RESTARTER_STATE_ONLINE, RESTARTER_STATE_NONE,
612             restarter_get_str_short(restarter_str_insert_in_graph))) {
613         case 0:
614                 break;

616         case ENOMEM:
617                 ++count;
618                 if (count < ALLOC_RETRY) {
619                         (void) poll(NULL, 0, msecs);
620                         msecs *= ALLOC_DELAY_MULT;
621                         goto timestamp;
622                 }

624                 uu_die("Insufficient memory.\n");
625                 /* NOTREACHED */

627         case ECONNABORTED:
628                 libscf_handle_rebind(hndl);
629                 goto timestamp;

631         case ENOENT:
632         case EPERM:
633         case EACCES:
634         case EROFS:
635                 log_error(LOG_INFO, "Could set state of %s: %s.\n",
636                     idata.i_fmri, strerror(r));
637                 break;

639         case EINVAL:
640         default:
641                 bad_error("_restarter_commit_states", r);
642         }

644         pg = safe_scf_pg_create(hndl);
645         prop = safe_scf_property_create(hndl);
646         val = safe_scf_value_create(hndl);
647         inst = safe_scf_instance_create(hndl);

649         /* set startd's restarter properties */
650         if (scf_handle_decode_fmri(hndl, SCF_SERVICE_STARTD, NULL, NULL, inst,
651             NULL, NULL, SCF_DECODE_FMRI_EXACT) == 0) {
652                 (void) libscf_write_start_pid(inst, getpid());
```

```
653                 ctid = proc_get_ctid();
654                 if (ctid != -1) {
655                         uint64 = (uint64_t)ctid;
656                         (void) libscf_inst_set_count_prop(inst,
657                             SCF_PG_RESTARTER, SCF_PG_RESTARTER_TYPE,
658                             SCF_PG_RESTARTER_FLAGS, SCF_PROPERTY_CONTRACT,
659                             uint64);
660                 }
661                 (void) libscf_note_method_log(inst, LOG_PREFIX_EARLY,
662                     STARTD_DEFAULT_LOG);
663                 (void) libscf_note_method_log(inst, LOG_PREFIX_NORMAL,
664                     STARTD_DEFAULT_LOG);
665         }

667         /* Read reconfigure property for recovery. */
668         if (scf_handle_decode_fmri(hndl, startd_reconfigure_fmri, NULL, NULL,
669             NULL, NULL, prop, NULL) != -1 &&
670             scf_property_get_value(prop, val) == 0)
671                 (void) scf_value_get_boolean(val, &prop_reconfig);

673         /*
674          * Set up the initial process secflags.  We do this super early, and
675          * in svc.startd, so that it's inherited by as much stuff as possible
676          * upon boot.
677          */
678         init_secflags(hndl);

680 #endif /* ! codereview */
681         if (scf_handle_decode_fmri(hndl, startd_options_fmri, NULL, NULL, NULL,
682             pg, NULL, SCF_DECODE_FMRI_TRUNCATE) == -1) {
683                 /*
684                  * No configuration options defined.
685                  */
686                 if (scf_error() != SCF_ERROR_NOT_FOUND)
687                         uu_warn("Couldn't read configuration from 'options' "
688                             "group: %s\n", scf_strerror(scf_error()));
689                 goto scfout;
690         }

692         /*
693          * If there is no "options" group defined, then our defaults are fine.
694          */
695         if (scf_pg_get_name(pg, NULL, 0) < 0)
696                 goto scfout;

698         /* get info_events_all */
699         info_events_all = libscf_get_info_events_all(pg);

701         /* Iterate through. */
702         iter = safe_scf_iter_create(hndl);

704         (void) scf_iter_pg_properties(iter, pg);

706         piter = safe_scf_iter_create(hndl);
707         vbuf = startd_alloc(max_scf_value_size);

709         while ((scf_iter_next_property(iter, prop) == 1)) {
710                 scf_type_t ty;

712                 if (scf_property_get_name(prop, buf, max_scf_fmri_size) < 0)
713                         continue;

715                 if (strcmp(buf, "logging") != 0 &&
716                     strcmp(buf, "boot_messages") != 0)
717                         continue;
```



```
719                 if (scf_property_type(prop, &ty) != 0) {
720                         switch (scf_error()) {
721                         case SCF_ERROR_CONNECTION_BROKEN:
722                         default:
723                                 libscf_handle_rebind(hndl);
724                                 continue;

726                         case SCF_ERROR_DELETED:
727                                 continue;

729                         case SCF_ERROR_NOT_BOUND:
730                         case SCF_ERROR_NOT_SET:
731                                 bad_error("scf_property_type", scf_error());
732                         }
733                 }

735                 if (ty != SCF_TYPE_ASTRING) {
736                         uu_warn("property \"options/%s\" is not of type "
737                             "astring; ignored.\n", buf);
738                         continue;
739                 }

741                 if (scf_property_get_value(prop, val) != 0) {
742                         switch (scf_error()) {
743                         case SCF_ERROR_CONNECTION_BROKEN:
744                         default:
745                                 return (ECONNABORTED);

747                         case SCF_ERROR_DELETED:
748                         case SCF_ERROR_NOT_FOUND:
749                                 return (0);

751                         case SCF_ERROR_CONSTRAINT_VIOLATED:
752                                 uu_warn("property \"options/%s\" has multiple "
753                                     "values; ignored.\n", buf);
754                                 continue;

756                         case SCF_ERROR_PERMISSION_DENIED:
757                                 uu_warn("property \"options/%s\" cannot be "
758                                     "read because startd has insufficient "
759                                     "permission; ignored.\n", buf);
760                                 continue;

762                         case SCF_ERROR_HANDLE_MISMATCH:
763                         case SCF_ERROR_NOT_BOUND:
764                         case SCF_ERROR_NOT_SET:
765                                 bad_error("scf_property_get_value",
766                                     scf_error());
767                         }
768                 }

770                 if (scf_value_get_astring(val, vbuf, max_scf_value_size) < 0)
771                         bad_error("scf_value_get_astring", scf_error());

773                 if (strcmp("logging", buf) == 0) {
774                         if (strcmp("verbose", vbuf) == 0) {
775                                 st->st_boot_flags = STARTD_BOOT_VERBOSE;
776                                 st->st_log_level_min = LOG_INFO;
777                         } else if (strcmp("debug", vbuf) == 0) {
778                                 st->st_boot_flags = STARTD_BOOT_VERBOSE;
779                                 st->st_log_level_min = LOG_DEBUG;
780                         } else if (strcmp("quiet", vbuf) == 0) {
781                                 st->st_log_level_min = LOG_NOTICE;
782                         } else {
783                                 uu_warn("unknown options/logging "
784                                     "value '%s' ignored\n", vbuf);
```

```
785                         }
786
787                 } else if (strcmp("boot_messages", buf) == 0) {
788                         if (strcmp("quiet", vbuf) == 0) {
789                                 st->st_boot_flags = STARTD_BOOT_QUIET;
790                         } else if (strcmp("verbose", vbuf) == 0) {
791                                 st->st_boot_flags = STARTD_BOOT_VERBOSE;
792                         } else {
793                                 log_framework(LOG_NOTICE, "unknown "
794                                     "options/boot_messages value '%s' "
795                                     "ignored\n", vbuf);
796                         }
797
798                 }
799         }
800
801         startd_free(vbuf, max_scf_value_size);
802         scf_iter_destroy(piter);
803
804         scf_iter_destroy(iter);
805
806 scfout:
807         scf_value_destroy(val);
808         scf_pg_destroy(pg);
809         scf_property_destroy(prop);
810         scf_instance_destroy(inst);
811         (void) scf_handle_unbind(hndl);
812         scf_handle_destroy(hndl);
813
814 noscfout:
815         startd_free(buf, max_scf_fmri_size);
816         uu_free(startd_options_fmri);
817         uu_free(startd_reconfigure_fmri);
818
819         if (booting_to_single_user) {
820                 st->st_subgraph = startd_alloc(max_scf_fmri_size);
821                 sz = strlcpy(st->st_subgraph, "milestone/single-user:default",
822                     max_scf_fmri_size);
823                 assert(sz < max_scf_fmri_size);
824         }
825
826         /*
827          * Options passed in as boot arguments override repository defaults.
828          */
829         env_opts = getenv("SMF_OPTIONS");
830         if (env_opts == NULL)
831                 return (ret);
832
833         for (cp = strtok_r(env_opts, ",", &lasts); cp != NULL;
834             cp = strtok_r(NULL, ",", &lasts)) {
835                 if (strcmp(cp, "debug") == 0) {
836                         st->st_boot_flags = STARTD_BOOT_VERBOSE;
837                         st->st_log_level_min = LOG_DEBUG;
838
839                         /* -m debug should send messages to console */
840                         st->st_log_flags =
841                             st->st_log_flags | STARTD_LOG_TERMINAL;
842                 } else if (strcmp(cp, "verbose") == 0) {
843                         st->st_boot_flags = STARTD_BOOT_VERBOSE;
844                         st->st_log_level_min = LOG_INFO;
845                 } else if (strcmp(cp, "seed") == 0) {
846                         uu_warn("SMF option \"%s\" unimplemented.\n", cp);
847                 } else if (strcmp(cp, "quiet") == 0) {
848                         st->st_log_level_min = LOG_NOTICE;
849                 } else if (strncmp(cp, "milestone=",
850                     sizeof ("milestone=") - 1) == 0) {
```



```
851                         char *mp = cp + sizeof ("milestone=") - 1;
852
853                         if (booting_to_single_user)
854                                 continue;
855
856                         if (st->st_subgraph == NULL) {
857                                 st->st_subgraph =
858                                     startd_alloc(max_scf_fmri_size);
859                                 st->st_subgraph[0] = '\0';
860                         }
861
862                         if (mp[0] == '\0' || strcmp(mp, "all") == 0) {
863                                 (void) strcpy(st->st_subgraph, "all");
864                         } else if (strcmp(mp, "su") == 0 ||
865                             strcmp(mp, "single-user") == 0) {
866                                 (void) strcpy(st->st_subgraph,
867                                     "milestone/single-user:default");
868                         } else if (strcmp(mp, "mu") == 0 ||
869                             strcmp(mp, "multi-user") == 0) {
870                                 (void) strcpy(st->st_subgraph,
871                                     "milestone/multi-user:default");
872                         } else if (strcmp(mp, "mus") == 0 ||
873                             strcmp(mp, "multi-user-server") == 0) {
874                                 (void) strcpy(st->st_subgraph,
875                                     "milestone/multi-user-server:default");
876                         } else if (strcmp(mp, "none") == 0) {
877                                 (void) strcpy(st->st_subgraph, "none");
878                         } else {
879                                 log_framework(LOG_NOTICE,
880                                     "invalid milestone option value "
881                                     "'%s' ignored\n", mp);
882                         }
883                 } else {
884                         uu_warn("Unknown SMF option \"%s\".\n", cp);
885                 }
886         }
887
888         return (ret);
889 }
890
891 /*
892  * void set_boot_env()
893  *
894  * If -r was passed or /reconfigure exists, this is a reconfig
895  * reboot.  We need to make sure that this information is given
896  * to the appropriate services the first time they're started
897  * by setting the system/reconfigure repository property,
898  * as well as pass the _INIT_RECONFIG variable on to the rcS
899  * start method so that legacy services can continue to use it.
900  *
901  * This function must never be called before contract_init(), as
902  * it sets st_initial.  get_startd_config() sets prop_reconfig from
903  * pre-existing repository state.
904  */
905 static void
906 set_boot_env()
907 {
908         struct stat sb;
909         int r;
910
911         /*
912          * Check if property still is set -- indicates we didn't get
913          * far enough previously to unset it.  Otherwise, if this isn't
914          * the first startup, don't re-process /reconfigure or the
915          * boot flag.
916          */
```

```
917         if (prop_reconfig != 1 && st->st_initial != 1)
918                 return;

920         /* If /reconfigure exists, also set opt_reconfig. */
921         if (stat("/reconfigure", &sb) != -1)
922                 opt_reconfig = 1;

924         /* Nothing to do.  Just return. */
925         if (opt_reconfig == 0 && prop_reconfig == 0)
926                 return;

928         /*
929          * Set startd's reconfigure property.  This property is
930          * then cleared by successful completion of the single-user
931          * milestone.
932          */
933         if (prop_reconfig != 1) {
934                 r = libscf_set_reconfig(1);
935                 switch (r) {
936                 case 0:
937                         break;

939                 case ENOENT:
940                 case EPERM:
941                 case EACCES:
942                 case EROFS:
943                         log_error(LOG_WARNING, "Could not set reconfiguration "
944                             "property: %s\n", strerror(r));
945                         break;

947                 default:
948                         bad_error("libscf_set_reconfig", r);
949                 }
950         }
951 }

953 static void
954 startup(void)
955 {
956         ctid_t configd_ctid;
957         int err;

959         /*
960          * Initialize data structures.
961          */
962         gu = startd_zalloc(sizeof (graph_update_t));
963         ru = startd_zalloc(sizeof (restarter_update_t));

965         (void) pthread_cond_init(&st->st_load_cv, NULL);
966         (void) pthread_cond_init(&st->st_configd_live_cv, NULL);
967         (void) pthread_cond_init(&gu->gu_cv, NULL);
968         (void) pthread_cond_init(&gu->gu_freeze_cv, NULL);
969         (void) pthread_cond_init(&ru->restarter_update_cv, NULL);
970         (void) pthread_mutex_init(&st->st_load_lock, &mutex_attrs);
971         (void) pthread_mutex_init(&st->st_configd_live_lock, &mutex_attrs);
972         (void) pthread_mutex_init(&gu->gu_lock, &mutex_attrs);
973         (void) pthread_mutex_init(&gu->gu_freeze_lock, &mutex_attrs);
974         (void) pthread_mutex_init(&ru->restarter_update_lock, &mutex_attrs);

976         configd_ctid = contract_init();

978         if (configd_ctid != -1)
979                 log_framework(LOG_DEBUG, "Existing configd contract %ld; not "
980                     "starting svc.configd\n", configd_ctid);

982         /*
```



```
983          * Call utmpx_init() before creating the fork_configd() thread.
984          */
985         utmpx_init();

987         (void) startd_thread_create(fork_configd_thread, (void *)configd_ctid);

989         /*
990          * Await, if necessary, configd's initial arrival.
991          */
992         MUTEX_LOCK(&st->st_configd_live_lock);
993         while (!st->st_configd_lives) {
994                 log_framework(LOG_DEBUG, "Awaiting cv signal on "
995                     "configd_live_cv\n");
996                 err = pthread_cond_wait(&st->st_configd_live_cv,
997                     &st->st_configd_live_lock);
998                 assert(err == 0);
999         }
1000        MUTEX_UNLOCK(&st->st_configd_live_lock);

1002        wait_init();

1004        if (read_startd_config())
1005                log_framework(LOG_INFO, "svc.configd unable to provide startd "
1006                    "optional settings\n");

1008        log_init();
1009        dict_init();
1010        timeout_init();
1011        restarter_protocol_init();
1012        restarter_init();

1014        /*
1015         * svc.configd is started by fork_configd_thread so repository access is
1016         * available, run early manifest import before continuing with starting
1017         * graph engine and the rest of startd.
1018         */
1019        log_framework(LOG_DEBUG, "Calling fork_emi...\n");
1020        fork_emi();

1022        graph_protocol_init();
1023        graph_init();

1025        init_env();

1027        set_boot_env();
1028        restarter_start();
1029        graph_engine_start();
1030 }

1032 static void
1033 usage(const char *name)
1034 {
1035         uu_warn(gettext("usage: %s [-n]\n"), name);
1036         exit(UU_EXIT_USAGE);
1037 }

1039 static int
1040 daemonize_start(void)
1041 {
1042         pid_t pid;
1043         int fd;

1045         if ((pid = fork1()) < 0)
1046                 return (-1);

1048         if (pid != 0)
```

```
1049                 exit(0);

1051         (void) close(STDIN_FILENO);

1053         if ((fd = open("/dev/null", O_RDONLY)) == -1) {
1054                 uu_warn(gettext("can't connect stdin to /dev/null"));
1055         } else if (fd != STDIN_FILENO) {
1056                 (void) dup2(fd, STDIN_FILENO);
1057                 startd_close(fd);
1058         }

1060         closefrom(3);
1061         (void) dup2(STDERR_FILENO, STDOUT_FILENO);

1063         (void) setsid();
1064         (void) chdir("/");

1066         /* Use default umask that init handed us, but 022 to create files. */
1067         dmask = umask(022);
1068         fmask = umask(dmask);

1070         return (0);
1071 }

1073 /*ARGSUSED*/
1074 static void
1075 die_handler(int sig, siginfo_t *info, void *data)
1076 {
1077         finished = 1;
1078 }

1080 int
1081 main(int argc, char *argv[])
1082 {
1083         int opt;
1084         int daemonize = 1;
1085         struct sigaction act;
1086         sigset_t nullset;
1087         struct stat sb;

1089         (void) uu_setpname(argv[0]);

1091         st = startd_zalloc(sizeof (startd_state_t));

1093         (void) pthread_mutexattr_init(&mutex_attrs);
1094 #ifndef NDEBUG
1095         (void) pthread_mutexattr_settype(&mutex_attrs,
1096             PTHREAD_MUTEX_ERRORCHECK);
1097 #endif

1099         max_scf_name_size = scf_limit(SCF_LIMIT_MAX_NAME_LENGTH);
1100         max_scf_value_size = scf_limit(SCF_LIMIT_MAX_VALUE_LENGTH);
1101         max_scf_fmri_size = scf_limit(SCF_LIMIT_MAX_FMRI_LENGTH);

1103         if (max_scf_name_size == -1 || max_scf_value_size == -1 ||
1104             max_scf_value_size == -1)
1105                 uu_die("Can't determine repository maximum lengths.\n");

1107         max_scf_name_size++;
1108         max_scf_value_size++;
1109         max_scf_fmri_size++;

1111         st->st_log_flags = STARTD_LOG_FILE | STARTD_LOG_SYSLOG;
1112         st->st_log_level_min = LOG_NOTICE;

1114         while ((opt = getopt(argc, argv, "nrs")) != EOF) {
```



```
1115                 switch (opt) {
1116                 case 'n':
1117                         daemonize = 0;
1118                         break;
1119                 case 'r':                       /* reconfiguration boot */
1120                         opt_reconfig = 1;
1121                         break;
1122                 case 's':                       /* single-user mode */
1123                         booting_to_single_user = B_TRUE;
1124                         break;
1125                 default:
1126                         usage(argv[0]);         /* exits */
1127                 }
1128         }

1130         if (optind != argc)
1131                 usage(argv[0]);

1133         (void) enable_extended_FILE_stdio(-1, -1);

1135         if (daemonize)
1136                 if (daemonize_start() < 0)
1137                         uu_die("Can't daemonize\n");

1139         log_init();

1141         if (stat("/etc/svc/volatile/resetting", &sb) != -1) {
1142                 log_framework(LOG_NOTICE, "Restarter quiesced.\n");

1144                 for (;;)
1145                         (void) pause();
1146         }

1148         act.sa_sigaction = &die_handler;
1149         (void) sigfillset(&act.sa_mask);
1150         act.sa_flags = SA_SIGINFO;
1151         (void) sigaction(SIGINT, &act, NULL);
1152         (void) sigaction(SIGTERM, &act, NULL);

1154         startup();

1156         (void) sigemptyset(&nullset);
1157         while (!finished) {
1158                 log_framework(LOG_DEBUG, "Main thread paused\n");
1159                 (void) sigsuspend(&nullset);
1160         }

1162         (void) log_framework(LOG_DEBUG, "Restarter exiting.\n");
1163         return (0);
1164 }
```

```
**********************************************************
   68588 Wed May 27 19:49:07 2015
new/usr/src/cmd/truss/print.c
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright (c) 1989, 2010, Oracle and/or its affiliates. All rights reserved.
 * Copyright (c) 2015, Joyent, Inc. All rights reserved.
 */

/*      Copyright (c) 1984, 1986, 1987, 1988, 1989 AT&T */
/*        All Rights Reserved   */

/* Copyright (c) 2013, OmniTI Computer Consulting, Inc. All rights reserved. */

#define _SYSCALL32      /* make 32-bit compat headers visible */

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <signal.h>
#include <termio.h>
#include <stddef.h>
#include <limits.h>
#include <fcntl.h>
#include <ctype.h>
#include <sys/types.h>
#include <sys/mman.h>
#include <sys/resource.h>
#include <sys/ulimit.h>
#include <sys/utsname.h>
#include <sys/kstat.h>
#include <sys/modctl.h>
#include <sys/acl.h>
#include <stropts.h>
#include <sys/isa_defs.h>
#include <sys/systeminfo.h>
#include <sys/cladm.h>
#include <sys/lwp.h>
#include <bsm/audit.h>
#include <libproc.h>
```



```
#include <priv.h>
#include <sys/aio.h>
#include <sys/aiocb.h>
#include <sys/corectl.h>
#include <sys/cpc_impl.h>
#include <sys/priocntl.h>
#include <sys/tspriocntl.h>
#include <sys/iapriocntl.h>
#include <sys/rtpriocntl.h>
#include <sys/fsspriocntl.h>
#include <sys/fxpriocntl.h>
#include <sys/proc.h>
#endif /* ! codereview */
#include <netdb.h>
#include <nss_dbdefs.h>
#include <sys/socketvar.h>
#include <netinet/in.h>
#include <netinet/tcp.h>
#include <netinet/udp.h>
#include <netinet/sctp.h>
#include <net/route.h>
#include <sys/utrap.h>
#include <sys/lgrp_user.h>
#include <sys/door.h>
#include <sys/tsol/tndb.h>
#include <sys/rctl.h>
#include <sys/rctl_impl.h>
#include <sys/fork.h>
#include <sys/task.h>
#include <sys/random.h>
#include "ramdata.h"
#include "print.h"
#include "proto.h"
#include "systable.h"

void grow(private_t *, int nbyte);

#define GROW(nb) if (pri->sys_leng + (nb) >= pri->sys_ssize) grow(pri, (nb))


/*ARGSUSED*/
void
prt_nov(private_t *pri, int raw, long val)      /* print nothing */
{
}

/*ARGSUSED*/
void
prt_dec(private_t *pri, int raw, long val)      /* print as decimal */
{
        GROW(24);
        if (data_model == PR_MODEL_ILP32)
                pri->sys_leng += sprintf(pri->sys_string + pri->sys_leng,
                    "%d", (int)val);
        else
                pri->sys_leng += sprintf(pri->sys_string + pri->sys_leng,
                    "%ld", val);
}

/*ARGSUSED*/
void
prt_uns(private_t *pri, int raw, long val)      /* print as unsigned decimal */
{
        GROW(24);
        if (data_model == PR_MODEL_ILP32)
                pri->sys_leng += sprintf(pri->sys_string + pri->sys_leng,
```

```
125                     "%u", (int)val);
126         else
127                 pri->sys_leng += sprintf(pri->sys_string + pri->sys_leng,
128                     "%lu", val);
129 }

131 /* print as unsigned decimal, except for -1 */
132 void
133 prt_un1(private_t *pri, int raw, long val)
134 {
135         if ((int)val == -1)
136                 prt_dec(pri, raw, val);
137         else
138                 prt_uns(pri, raw, val);
139 }

141 /*ARGSUSED*/
142 void
143 prt_oct(private_t *pri, int raw, long val)      /* print as octal */
144 {
145         GROW(24);
146         if (data_model == PR_MODEL_ILP32)
147                 pri->sys_leng += sprintf(pri->sys_string + pri->sys_leng,
148                     "%#o", (int)val);
149         else
150                 pri->sys_leng += sprintf(pri->sys_string + pri->sys_leng,
151                     "%#lo", val);
152 }

154 /*ARGSUSED*/
155 void
156 prt_hex(private_t *pri, int raw, long val)      /* print as hexadecimal */
157 {
158         GROW(20);
159         if (data_model == PR_MODEL_ILP32)
160                 pri->sys_leng += sprintf(pri->sys_string + pri->sys_leng,
161                     "0x%.8X", (int)val);
162         else
163                 pri->sys_leng += sprintf(pri->sys_string + pri->sys_leng,
164                     "0x%.8lX", val);
165 }

167 /* print as hexadecimal (half size) */
168 /*ARGSUSED*/
169 void
170 prt_hhx(private_t *pri, int raw, long val)
171 {
172         GROW(20);
173         if (data_model == PR_MODEL_ILP32)
174                 pri->sys_leng += sprintf(pri->sys_string + pri->sys_leng,
175                     "0x%.4X", (int)val);
176         else
177                 pri->sys_leng += sprintf(pri->sys_string + pri->sys_leng,
178                     "0x%.4lX", val);
179 }

181 /* print as decimal if small, else hexadecimal */
182 /*ARGSUSED*/
183 void
184 prt_dex(private_t *pri, int raw, long val)
185 {
186         if (val & 0xff000000)
187                 prt_hex(pri, 0, val);
188         else
189                 prt_dec(pri, 0, val);
190 }
```



```
192 /* print long long offset */
193 /*ARGSUSED*/
194 void
195 prt_llo(private_t *pri, int raw, long val1, long val2)
196 {
197         int hival;
198         int loval;

200 #ifdef  _LONG_LONG_LTOH
201         hival = (int)val2;
202         loval = (int)val1;
203 #else
204         hival = (int)val1;
205         loval = (int)val2;
206 #endif

208         if (hival == 0) {
209                 prt_dex(pri, 0, loval);
210         } else {
211                 GROW(18);
212                 pri->sys_leng +=
213                     sprintf(pri->sys_string + pri->sys_leng, "0x%.8X%.8X",
214                     hival, loval);
215         }
216 }

218 void
219 escape_string(private_t *pri, const char *s)
220 {
221         /*
222          * We want to avoid outputting unprintable characters that may
223          * destroy the user's terminal.  So we do one pass to find any
224          * unprintable characters, size the array appropriately, and
225          * then walk each character by hand.  Those that are unprintable
226          * are replaced by a hex escape (\xNN).  We also escape quotes for
227          * completeness.
228          */
229         int i, unprintable, quotes;
230         size_t len = strlen(s);
231         for (i = 0, unprintable = 0, quotes = 0; i < len; i++) {
232                 if (!isprint(s[i]))
233                         unprintable++;
234                 if (s[i] == '"')
235                         quotes++;
236         }

238         GROW(len + 3 * unprintable + quotes + 2);

240         pri->sys_string[pri->sys_leng++] = '"';
241         for (i = 0; i < len; i++) {
242                 if (s[i] == '"')
243                         pri->sys_string[pri->sys_leng++] = '\\';

245                 if (isprint(s[i])) {
246                         pri->sys_string[pri->sys_leng++] = s[i];
247                 } else {
248                         pri->sys_leng += sprintf(pri->sys_string +
249                             pri->sys_leng, "\\x%02x", (uint8_t)s[i]);
250                 }
251         }
252         pri->sys_string[pri->sys_leng++] = '"';
253 }

255 void
256 prt_stg(private_t *pri, int raw, long val)      /* print as string */
```

```
{
        char *s = raw? NULL : fetchstring(pri, (long)val, PATH_MAX);

        if (s == NULL)
                prt_hex(pri, 0, val);
        else
                escape_string(pri, s);
}

/* print as string returned from syscall */
void
prt_rst(private_t *pri, int raw, long val)
{
        char *s = (raw || pri->Errno)? NULL :
            fetchstring(pri, (long)val, PATH_MAX);

        if (s == NULL)
                prt_hex(pri, 0, val);
        else {
                GROW((int)strlen(s) + 2);
                pri->sys_leng += snprintf(pri->sys_string + pri->sys_leng,
                    pri->sys_ssize - pri->sys_leng, "\"%s\"", s);
        }
}

/* print contents of readlink() buffer */
void
prt_rlk(private_t *pri, int raw, long val)
{
        char *s = (raw || pri->Errno || pri->Rval1 <= 0)? NULL :
            fetchstring(pri, (long)val,
            (pri->Rval1 > PATH_MAX)? PATH_MAX : (int)pri->Rval1);

        if (s == NULL)
                prt_hex(pri, 0, val);
        else {
                GROW((int)strlen(s) + 2);
                pri->sys_leng += snprintf(pri->sys_string + pri->sys_leng,
                    pri->sys_ssize - pri->sys_leng, "\"%s\"", s);
        }
}

void
prt_ioc(private_t *pri, int raw, long val)      /* print ioctl code */
{
        const char *s = raw? NULL : ioctlname(pri, (int)val);

        if (s == NULL)
                prt_hex(pri, 0, val);
        else
                outstring(pri, s);
}

void
prt_ioa(private_t *pri, int raw, long val)      /* print ioctl argument */
{
        const char *s;

        /* cheating -- look at the ioctl() code */
        switch (pri->sys_args[1]) {

        /* kstat ioctl()s */
        case KSTAT_IOC_READ:
        case KSTAT_IOC_WRITE:
#ifdef _LP64
                if (data_model == PR_MODEL_ILP32)
```



```
                        prt_stg(pri, raw,
                            val + offsetof(kstat32_t, ks_name[0]));
                else
#endif
                        prt_stg(pri, raw,
                            val + offsetof(kstat_t, ks_name[0]));
                break;

        /* streams ioctl()s */
        case I_LOOK:
                prt_rst(pri, raw, val);
                break;
        case I_PUSH:
        case I_FIND:
                prt_stg(pri, raw, val);
                break;
        case I_LINK:
        case I_UNLINK:
        case I_SENDFD:
                prt_dec(pri, 0, val);
                break;
        case I_SRDOPT:
                if (raw || (s = strrdopt(val)) == NULL)
                        prt_dec(pri, 0, val);
                else
                        outstring(pri, s);
                break;
        case I_SETSIG:
                if (raw || (s = strevents(pri, val)) == NULL)
                        prt_hex(pri, 0, val);
                else
                        outstring(pri, s);
                break;
        case I_FLUSH:
                if (raw || (s = strflush(val)) == NULL)
                        prt_dec(pri, 0, val);
                else
                        outstring(pri, s);
                break;

        /* tty ioctl()s */
        case TCSBRK:
        case TCXONC:
        case TCFLSH:
        case TCDSET:
                prt_dec(pri, 0, val);
                break;

        default:
                prt_hex(pri, 0, val);
                break;
        }
}

void
prt_pip(private_t *pri, int raw, long val)      /* print pipe code */
{
        const char *s = NULL;

        if (!raw) {
                switch (val) {
                case O_CLOEXEC:
                        s = "O_CLOEXEC";
                        break;
                case O_NONBLOCK:
                        s = "O_NONBLOCK";
```

```
389                         break;
390                 case O_CLOEXEC|O_NONBLOCK:
391                         s = "O_CLOEXEC|O_NONBLOCK";
392                         break;
393                 }
394         }

396         if (s == NULL)
397                 prt_dex(pri, 0, val);
398         else
399                 outstring(pri, s);
400 }

402 void
403 prt_pfd(private_t *pri, int raw, long val)      /* print pipe code */
404 {
405         int fds[2];
406         char str[32];

408         /* the fds only have meaning if the return value is 0 */
409         if (!raw &&
410             pri->Rval1 >= 0 &&
411             Pread(Proc, fds, sizeof (fds), (long)val) == sizeof (fds)) {
412                 (void) snprintf(str, sizeof (str), "[%d,%d]", fds[0], fds[1]);
413                 outstring(pri, str);
414         } else {
415                 prt_hex(pri, 0, val);
416         }
417 }

419 void
420 prt_fcn(private_t *pri, int raw, long val)      /* print fcntl code */
421 {
422         const char *s = raw? NULL : fcntlname(val);

424         if (s == NULL)
425                 prt_dec(pri, 0, val);
426         else
427                 outstring(pri, s);
428 }

430 void
431 prt_s86(private_t *pri, int raw, long val)      /* print sysi86 code */
432 {

434         const char *s = raw? NULL : si86name(val);

436         if (s == NULL)
437                 prt_dec(pri, 0, val);
438         else
439                 outstring(pri, s);
440 }

442 void
443 prt_uts(private_t *pri, int raw, long val)      /* print utssys code */
444 {
445         const char *s = raw? NULL : utscode(val);

447         if (s == NULL)
448                 prt_dec(pri, 0, val);
449         else
450                 outstring(pri, s);
451 }

453 void
454 prt_msc(private_t *pri, int raw, long val)      /* print msgsys command */
```



```
455 {
456         const char *s = raw? NULL : msgcmd(val);

458         if (s == NULL)
459                 prt_dec(pri, 0, val);
460         else
461                 outstring(pri, s);
462 }

464 void
465 prt_msf(private_t *pri, int raw, long val)      /* print msgsys flags */
466 {
467         const char *s = raw? NULL : msgflags(pri, (int)val);

469         if (s == NULL)
470                 prt_oct(pri, 0, val);
471         else
472                 outstring(pri, s);
473 }

475 void
476 prt_smc(private_t *pri, int raw, long val)      /* print semsys command */
477 {
478         const char *s = raw? NULL : semcmd(val);

480         if (s == NULL)
481                 prt_dec(pri, 0, val);
482         else
483                 outstring(pri, s);
484 }

486 void
487 prt_sef(private_t *pri, int raw, long val)      /* print semsys flags */
488 {
489         const char *s = raw? NULL : semflags(pri, (int)val);

491         if (s == NULL)
492                 prt_oct(pri, 0, val);
493         else
494                 outstring(pri, s);
495 }

497 void
498 prt_shc(private_t *pri, int raw, long val)      /* print shmsys command */
499 {
500         const char *s = raw? NULL : shmcmd(val);

502         if (s == NULL)
503                 prt_dec(pri, 0, val);
504         else
505                 outstring(pri, s);
506 }

508 void
509 prt_shf(private_t *pri, int raw, long val)      /* print shmsys flags */
510 {
511         const char *s = raw? NULL : shmflags(pri, (int)val);

513         if (s == NULL)
514                 prt_oct(pri, 0, val);
515         else
516                 outstring(pri, s);
517 }

519 void
520 prt_sfs(private_t *pri, int raw, long val)      /* print sysfs code */
```

```
{
        const char *s = raw? NULL : sfsname(val);

        if (s == NULL)
                prt_dec(pri, 0, val);
        else
                outstring(pri, s);
}

void
prt_opn(private_t *pri, int raw, long val)      /* print open code */
{
        const char *s = raw? NULL : openarg(pri, val);

        if (s == NULL)
                prt_oct(pri, 0, val);
        else
                outstring(pri, s);
}

void
prt_sig(private_t *pri, int raw, long val)      /* print signal name */
{
        const char *s = raw? NULL : signame(pri, (int)val);

        if (s == NULL)
                prt_hex(pri, 0, val);
        else
                outstring(pri, s);
}

void
prt_smf(private_t *pri, int raw, long val) /* print streams message flags */
{
        switch (val) {
        case 0:
                prt_dec(pri, 0, val);
                break;
        case RS_HIPRI:
                if (raw)
                        prt_hhx(pri, 0, val);
                else
                        outstring(pri, "RS_HIPRI");
                break;
        default:
                prt_hhx(pri, 0, val);
                break;
        }
}

void
prt_mtf(private_t *pri, int raw, long val)      /* print mount flags */
{
        const char *s = raw? NULL : mountflags(pri, val);

        if (s == NULL)
                prt_hex(pri, 0, val);
        else
                outstring(pri, s);
}

void
prt_mft(private_t *pri, int raw, long val) /* print mount file system type */
{
        if (val >= 0 && val < 256)
                prt_dec(pri, 0, val);
```



```
        else if (raw)
                prt_hex(pri, 0, val);
        else
                prt_stg(pri, raw, val);
}

#define ISREAD(code) \
        ((code) == SYS_read || (code) == SYS_pread || (code) == SYS_pread64 || \
        (code) == SYS_recv || (code) == SYS_recvfrom)
#define ISWRITE(code) \
        ((code) == SYS_write || (code) == SYS_pwrite || \
        (code) == SYS_pwrite64 || (code) == SYS_send || (code) == SYS_sendto)

/* print contents of read() or write() I/O buffer */
void
prt_iob(private_t *pri, int raw, long val)
{
        const lwpstatus_t *Lsp = pri->lwpstat;
        int syscall = Lsp->pr_what;
        int fdp1 = pri->sys_args[0] + 1;
        ssize_t nbyte = ISWRITE(syscall)? pri->sys_args[2] :
            (pri->Errno? 0 : pri->Rval1);
        int elsewhere = FALSE;          /* TRUE iff dumped elsewhere */
        char buffer[IOBSIZE];

        pri->iob_buf[0] = '\0';

        if (Lsp->pr_why == PR_SYSEXIT && nbyte > IOBSIZE) {
                if (ISREAD(syscall))
                        elsewhere = prismember(&readfd, fdp1);
                else
                        elsewhere = prismember(&writefd, fdp1);
        }

        if (nbyte <= 0 || elsewhere)
                prt_hex(pri, 0, val);
        else {
                int nb = nbyte > IOBSIZE? IOBSIZE : (int)nbyte;

                if (Pread(Proc, buffer, (size_t)nb, (long)val) != nb)
                        prt_hex(pri, 0, val);
                else {
                        pri->iob_buf[0] = '"';
                        showbytes(buffer, nb, pri->iob_buf + 1);
                        (void) strlcat(pri->iob_buf,
                            (nb == nbyte)?
                            (const char *)"\"" : (const char *)"\"..",
                            sizeof (pri->iob_buf));
                        if (raw)
                                prt_hex(pri, 0, val);
                        else
                                outstring(pri, pri->iob_buf);
                }
        }
}
#undef  ISREAD
#undef  ISWRITE

void
prt_idt(private_t *pri, int raw, long val) /* print idtype_t, waitid() arg */
{
        const char *s = raw? NULL : idtype_enum(pri, val);

        if (s == NULL)
                prt_dec(pri, 0, val);
        else
```

```
653                 outstring(pri, s);
654 }
655
656 void
657 prt_wop(private_t *pri, int raw, long val)      /* print waitid() options */
658 {
659         const char *s = raw? NULL : woptions(pri, (int)val);
660
661         if (s == NULL)
662                 prt_oct(pri, 0, val);
663         else
664                 outstring(pri, s);
665 }
666
667 void
668 prt_whn(private_t *pri, int raw, long val) /* print lseek() whence argument */
669 {
670         const char *s = raw? NULL : whencearg(val);
671
672         if (s == NULL)
673                 prt_dec(pri, 0, val);
674         else
675                 outstring(pri, s);
676 }
677
678 /*ARGSUSED*/
679 void
680 prt_spm(private_t *pri, int raw, long val)      /* print sigprocmask argument */
681 {
682         const char *s = NULL;
683
684         if (!raw) {
685                 switch (val) {
686                 case SIG_BLOCK:         s = "SIG_BLOCK";        break;
687                 case SIG_UNBLOCK:       s = "SIG_UNBLOCK";      break;
688                 case SIG_SETMASK:       s = "SIG_SETMASK";      break;
689                 }
690         }
691
692         if (s == NULL)
693                 prt_dec(pri, 0, val);
694         else
695                 outstring(pri, s);
696 }
697
698 const char *
699 mmap_protect(private_t *pri, long arg)
700 {
701         char *str = pri->code_buf;
702
703         if (arg & ~(PROT_READ|PROT_WRITE|PROT_EXEC))
704                 return ((char *)NULL);
705
706         if (arg == PROT_NONE)
707                 return ("PROT_NONE");
708
709         *str = '\0';
710         if (arg & PROT_READ)
711                 (void) strlcat(str, "|PROT_READ", sizeof (pri->code_buf));
712         if (arg & PROT_WRITE)
713                 (void) strlcat(str, "|PROT_WRITE", sizeof (pri->code_buf));
714         if (arg & PROT_EXEC)
715                 (void) strlcat(str, "|PROT_EXEC", sizeof (pri->code_buf));
716         return ((const char *)(str + 1));
717 }
```



```
719 const char *
720 mmap_type(private_t *pri, long arg)
721 {
722         char *str = pri->code_buf;
723         size_t used;
724
725 #define CBSIZE  sizeof (pri->code_buf)
726         switch (arg & MAP_TYPE) {
727         case MAP_SHARED:
728                 used = strlcpy(str, "MAP_SHARED", CBSIZE);
729                 break;
730         case MAP_PRIVATE:
731                 used = strlcpy(str, "MAP_PRIVATE", CBSIZE);
732                 break;
733         default:
734                 used = snprintf(str, CBSIZE, "%ld", arg&MAP_TYPE);
735                 break;
736         }
737
738         arg &= ~(_MAP_NEW|MAP_TYPE);
739
740         if (arg & ~(MAP_FIXED|MAP_RENAME|MAP_NORESERVE|MAP_ANON|MAP_ALIGN|
741             MAP_TEXT|MAP_INITDATA|MAP_32BIT))
742                 (void) snprintf(str + used, sizeof (pri->code_buf) - used,
743                     "|0x%lX", arg);
744         else {
745                 if (arg & MAP_FIXED)
746                         (void) strlcat(str, "|MAP_FIXED", CBSIZE);
747                 if (arg & MAP_RENAME)
748                         (void) strlcat(str, "|MAP_RENAME", CBSIZE);
749                 if (arg & MAP_NORESERVE)
750                         (void) strlcat(str, "|MAP_NORESERVE", CBSIZE);
751                 if (arg & MAP_ANON)
752                         (void) strlcat(str, "|MAP_ANON", CBSIZE);
753                 if (arg & MAP_ALIGN)
754                         (void) strlcat(str, "|MAP_ALIGN", CBSIZE);
755                 if (arg & MAP_TEXT)
756                         (void) strlcat(str, "|MAP_TEXT", CBSIZE);
757                 if (arg & MAP_INITDATA)
758                         (void) strlcat(str, "|MAP_INITDATA", CBSIZE);
759                 if (arg & MAP_32BIT)
760                         (void) strlcat(str, "|MAP_32BIT", CBSIZE);
761         }
762
763         return ((const char *)str);
764 #undef CBSIZE
765 }
766
767 void
768 prt_mpr(private_t *pri, int raw, long val) /* print mmap()/mprotect() flags */
769 {
770         const char *s = raw? NULL : mmap_protect(pri, val);
771
772         if (s == NULL)
773                 prt_hhx(pri, 0, val);
774         else
775                 outstring(pri, s);
776 }
777
778 void
779 prt_mty(private_t *pri, int raw, long val) /* print mmap() mapping type flags */
780 {
781         const char *s = raw? NULL : mmap_type(pri, val);
782
783         if (s == NULL)
784                 prt_hhx(pri, 0, val);
```

```
785         else
786                 outstring(pri, s);
787 }

789 void
790 prt_mob(private_t *pri, int raw, long val) /* print mmapobj() flags */
791 {
792         if (val == 0)
793                 prt_dec(pri, 0, val);
794         else if (raw || (val & ~(MMOBJ_PADDING|MMOBJ_INTERPRET)) != 0)
795                 prt_hhx(pri, 0, val);
796         else {
797 #define CBSIZE  sizeof (pri->code_buf)
798                 char *s = pri->code_buf;

800                 *s = '\0';
801                 if (val & MMOBJ_PADDING)
802                         (void) strlcat(s, "|MMOBJ_PADDING", CBSIZE);
803                 if (val & MMOBJ_INTERPRET)
804                         (void) strlcat(s, "|MMOBJ_INTERPRET", CBSIZE);
805                 outstring(pri, s + 1);
806 #undef CBSIZE
807         }
808 }

810 /*ARGSUSED*/
811 void
812 prt_mcf(private_t *pri, int raw, long val)      /* print memcntl() function */
813 {
814         const char *s = NULL;

816         if (!raw) {
817                 switch (val) {
818                 case MC_SYNC:           s = "MC_SYNC";          break;
819                 case MC_LOCK:           s = "MC_LOCK";          break;
820                 case MC_UNLOCK:         s = "MC_UNLOCK";        break;
821                 case MC_ADVISE:         s = "MC_ADVISE";        break;
822                 case MC_LOCKAS:         s = "MC_LOCKAS";        break;
823                 case MC_UNLOCKAS:       s = "MC_UNLOCKAS";      break;
824                 case MC_HAT_ADVISE:     s = "MC_HAT_ADVISE";    break;
825                 }
826         }

828         if (s == NULL)
829                 prt_dec(pri, 0, val);
830         else
831                 outstring(pri, s);
832 }

834 void
835 prt_mad(private_t *pri, int raw, long val)      /* print madvise() argument */
836 {
837         const char *s = NULL;

839         if (!raw) {
840                 switch (val) {
841                 case MADV_NORMAL:       s = "MADV_NORMAL";      break;
842                 case MADV_RANDOM:       s = "MADV_RANDOM";      break;
843                 case MADV_SEQUENTIAL:   s = "MADV_SEQUENTIAL";  break;
844                 case MADV_WILLNEED:     s = "MADV_WILLNEED";    break;
845                 case MADV_DONTNEED:     s = "MADV_DONTNEED";    break;
846                 case MADV_FREE:         s = "MADV_FREE";        break;
847                 case MADV_ACCESS_DEFAULT: s = "MADV_ACCESS_DEFAULT";    break;
848                 case MADV_ACCESS_LWP:   s = "MADV_ACCESS_LWP";  break;
849                 case MADV_ACCESS_MANY:  s = "MADV_ACCESS_MANY"; break;
850                 }
```



```
851         }

853         if (s == NULL)
854                 prt_dec(pri, 0, val);
855         else
856                 outstring(pri, s);
857 }

859 void
860 prt_mc4(private_t *pri, int raw, long val) /* print memcntl() (4th) argument */
861 {
862         if (val == 0)
863                 prt_dec(pri, 0, val);
864         else if (raw)
865                 prt_hhx(pri, 0, val);
866         else {
867                 char *s = NULL;

869 #define CBSIZE  sizeof (pri->code_buf)
870                 /* cheating -- look at memcntl func */
871                 switch (pri->sys_args[2]) {
872                 case MC_ADVISE:
873                         prt_mad(pri, 0, val);
874                         return;

876                 case MC_SYNC:
877                         if ((val & ~(MS_SYNC|MS_ASYNC|MS_INVALIDATE)) == 0) {
878                                 *(s = pri->code_buf) = '\0';
879                                 if (val & MS_SYNC)
880                                         (void) strlcat(s, "|MS_SYNC", CBSIZE);
881                                 if (val & MS_ASYNC)
882                                         (void) strlcat(s, "|MS_ASYNC", CBSIZE);
883                                 if (val & MS_INVALIDATE)
884                                         (void) strlcat(s, "|MS_INVALIDATE",
885                                             CBSIZE);
886                         }
887                         break;

889                 case MC_LOCKAS:
890                 case MC_UNLOCKAS:
891                         if ((val & ~(MCL_CURRENT|MCL_FUTURE)) == 0) {
892                                 *(s = pri->code_buf) = '\0';
893                                 if (val & MCL_CURRENT)
894                                         (void) strlcat(s, "|MCL_CURRENT",
895                                             CBSIZE);
896                                 if (val & MCL_FUTURE)
897                                         (void) strlcat(s, "|MCL_FUTURE",
898                                             CBSIZE);
899                         }
900                         break;
901                 }
902 #undef CBSIZE

904                 if (s == NULL || *s == '\0')
905                         prt_hhx(pri, 0, val);
906                 else
907                         outstring(pri, ++s);
908         }
909 }

911 void
912 prt_mc5(private_t *pri, int raw, long val) /* print memcntl() (5th) argument */
913 {
914         char *s;

916 #define CBSIZE  sizeof (pri->code_buf)
```

```
917         if (val == 0)
918                 prt_dec(pri, 0, val);
919         else if (raw || (val & ~VALID_ATTR))
920                 prt_hhx(pri, 0, val);
921         else {
922                 s = pri->code_buf;
923                 *s = '\0';
924                 if (val & SHARED)
925                         (void) strlcat(s, "|SHARED", CBSIZE);
926                 if (val & PRIVATE)
927                         (void) strlcat(s, "|PRIVATE", CBSIZE);
928                 if (val & PROT_READ)
929                         (void) strlcat(s, "|PROT_READ", CBSIZE);
930                 if (val & PROT_WRITE)
931                         (void) strlcat(s, "|PROT_WRITE", CBSIZE);
932                 if (val & PROT_EXEC)
933                         (void) strlcat(s, "|PROT_EXEC", CBSIZE);
934                 if (*s == '\0')
935                         prt_hhx(pri, 0, val);
936                 else
937                         outstring(pri, ++s);
938         }
939 #undef CBSIZE
940 }

942 void
943 prt_ulm(private_t *pri, int raw, long val)      /* print ulimit() argument */
944 {
945         const char *s = NULL;

947         if (!raw) {
948                 switch (val) {
949                 case UL_GFILLIM:        s = "UL_GFILLIM";       break;
950                 case UL_SFILLIM:        s = "UL_SFILLIM";       break;
951                 case UL_GMEMLIM:        s = "UL_GMEMLIM";       break;
952                 case UL_GDESLIM:        s = "UL_GDESLIM";       break;
953                 }
954         }

956         if (s == NULL)
957                 prt_dec(pri, 0, val);
958         else
959                 outstring(pri, s);
960 }

962 void
963 prt_rlm(private_t *pri, int raw, long val) /* print get/setrlimit() argument */
964 {
965         const char *s = NULL;

967         if (!raw) {
968                 switch (val) {
969                 case RLIMIT_CPU:        s = "RLIMIT_CPU";       break;
970                 case RLIMIT_FSIZE:      s = "RLIMIT_FSIZE";     break;
971                 case RLIMIT_DATA:       s = "RLIMIT_DATA";      break;
972                 case RLIMIT_STACK:      s = "RLIMIT_STACK";     break;
973                 case RLIMIT_CORE:       s = "RLIMIT_CORE";      break;
974                 case RLIMIT_NOFILE:     s = "RLIMIT_NOFILE";    break;
975                 case RLIMIT_VMEM:       s = "RLIMIT_VMEM";      break;
976                 }
977         }

979         if (s == NULL)
980                 prt_dec(pri, 0, val);
981         else
982                 outstring(pri, s);
```



```
983 }

985 void
986 prt_cnf(private_t *pri, int raw, long val)      /* print sysconfig code */
987 {
988         const char *s = raw? NULL : sconfname(val);

990         if (s == NULL)
991                 prt_dec(pri, 0, val);
992         else
993                 outstring(pri, s);
994 }

996 void
997 prt_inf(private_t *pri, int raw, long val)      /* print sysinfo code */
998 {
999         const char *s = NULL;

1001        if (!raw) {
1002                switch (val) {
1003                case SI_SYSNAME:        s = "SI_SYSNAME";       break;
1004                case SI_HOSTNAME:       s = "SI_HOSTNAME";      break;
1005                case SI_RELEASE:        s = "SI_RELEASE";       break;
1006                case SI_VERSION:        s = "SI_VERSION";       break;
1007                case SI_MACHINE:        s = "SI_MACHINE";       break;
1008                case SI_ARCHITECTURE:   s = "SI_ARCHITECTURE";  break;
1009                case SI_ARCHITECTURE_32:s = "SI_ARCHITECTURE_32"; break;
1010                case SI_ARCHITECTURE_64:s = "SI_ARCHITECTURE_64"; break;
1011                case SI_ARCHITECTURE_K: s = "SI_ARCHITECTURE_K"; break;
1012                case SI_HW_SERIAL:      s = "SI_HW_SERIAL";     break;
1013                case SI_HW_PROVIDER:    s = "SI_HW_PROVIDER";   break;
1014                case SI_SRPC_DOMAIN:    s = "SI_SRPC_DOMAIN";   break;
1015                case SI_SET_HOSTNAME:   s = "SI_SET_HOSTNAME";  break;
1016                case SI_SET_SRPC_DOMAIN: s = "SI_SET_SRPC_DOMAIN"; break;
1017                case SI_PLATFORM:       s = "SI_PLATFORM";      break;
1018                case SI_ISALIST:        s = "SI_ISALIST";       break;
1019                case SI_DHCP_CACHE:     s = "SI_DHCP_CACHE";    break;
1020                }
1021        }

1023        if (s == NULL)
1024                prt_dec(pri, 0, val);
1025        else
1026                outstring(pri, s);
1027 }

1029 void
1030 prt_ptc(private_t *pri, int raw, long val)      /* print pathconf code */
1031 {
1032        const char *s = raw? NULL : pathconfname(val);

1034        if (s == NULL)
1035                prt_dec(pri, 0, val);
1036        else
1037                outstring(pri, s);
1038 }

1040 void
1041 prt_fui(private_t *pri, int raw, long val) /* print fusers() input argument */
1042 {
1043        const char *s = raw? NULL : fuiname(val);

1045        if (s == NULL)
1046                prt_hhx(pri, 0, val);
1047        else
1048                outstring(pri, s);
```

```
}

void
prt_lwf(private_t *pri, int raw, long val)      /* print lwp_create() flags */
{
        char *s;

        if (val == 0)
                prt_dec(pri, 0, val);
        else if (raw ||
            (val & ~(LWP_DAEMON|LWP_DETACHED|LWP_SUSPENDED)))
                prt_hhx(pri, 0, val);
        else {
#define CBSIZE  sizeof (pri->code_buf)
                s = pri->code_buf;
                *s = '\0';
                if (val & LWP_DAEMON)
                        (void) strlcat(s, "|LWP_DAEMON", CBSIZE);
                if (val & LWP_DETACHED)
                        (void) strlcat(s, "|LWP_DETACHED", CBSIZE);
                if (val & LWP_SUSPENDED)
                        (void) strlcat(s, "|LWP_SUSPENDED", CBSIZE);
                outstring(pri, ++s);
#undef CBSIZE
        }
}

void
prt_itm(private_t *pri, int raw, long val) /* print [get|set]itimer() arg */
{
        const char *s = NULL;

        if (!raw) {
                switch (val) {
                case ITIMER_REAL:       s = "ITIMER_REAL";      break;
                case ITIMER_VIRTUAL:    s = "ITIMER_VIRTUAL";   break;
                case ITIMER_PROF:       s = "ITIMER_PROF";      break;
#ifdef ITIMER_REALPROF
                case ITIMER_REALPROF:   s = "ITIMER_REALPROF";  break;
#endif
                }
        }

        if (s == NULL)
                prt_dec(pri, 0, val);
        else
                outstring(pri, s);
}

void
prt_mod(private_t *pri, int raw, long val)      /* print modctl() code */
{
        const char *s = NULL;

        if (!raw) {
                switch (val) {
                case MODLOAD:           s = "MODLOAD";          break;
                case MODUNLOAD:         s = "MODUNLOAD";        break;
                case MODINFO:           s = "MODINFO";          break;
                case MODRESERVED:       s = "MODRESERVED";      break;
                case MODSETMINIROOT:    s = "MODSETMINIROOT";   break;
                case MODADDMAJBIND:     s = "MODADDMAJBIND";    break;
                case MODGETPATH:        s = "MODGETPATH";       break;
                case MODGETPATHLEN:     s = "MODGETPATHLEN";    break;
                case MODREADSYSBIND:    s = "MODREADSYSBIND";   break;
                case MODGETMAJBIND:     s = "MODGETMAJBIND";    break;
```



```
                case MODGETNAME:        s = "MODGETNAME";       break;
                case MODSIZEOF_DEVID:   s = "MODSIZEOF_DEVID";  break;
                case MODGETDEVID:       s = "MODGETDEVID";      break;
                case MODSIZEOF_MINORNAME: s = "MODSIZEOF_MINORNAME"; break;
                case MODGETMINORNAME:   s = "MODGETMINORNAME";  break;
                case MODGETFBNAME:      s = "MODGETFBNAME";     break;
                case MODEVENTS:         s = "MODEVENTS";        break;
                case MODREREADDACF:     s = "MODREREADDACF";    break;
                case MODLOADDRVCONF:    s = "MODLOADDRVCONF";   break;
                case MODUNLOADDRVCONF:  s = "MODUNLOADDRVCONF"; break;
                case MODREMMAJBIND:     s = "MODREMMAJBIND";    break;
                case MODDEVT2INSTANCE:  s = "MODDEVT2INSTANCE"; break;
                case MODGETDEVFSPATH_LEN: s = "MODGETDEVFSPATH_LEN"; break;
                case MODGETDEVFSPATH:   s = "MODGETDEVFSPATH";  break;
                case MODDEVID2PATHS:    s = "MODDEVID2PATHS";   break;
                case MODSETDEVPOLICY:   s = "MODSETDEVPOLICY";  break;
                case MODGETDEVPOLICY:   s = "MODGETDEVPOLICY";  break;
                case MODALLOCPRIV:      s = "MODALLOCPRIV";     break;
                case MODGETDEVPOLICYBYNAME:
                                        s = "MODGETDEVPOLICYBYNAME"; break;
                case MODLOADMINORPERM:  s = "MODLOADMINORPERM"; break;
                case MODADDMINORPERM:   s = "MODADDMINORPERM"; break;
                case MODREMMINORPERM:   s = "MODREMMINORPERM"; break;
                case MODREMDRVCLEANUP:  s = "MODREMDRVCLEANUP"; break;
                case MODDEVEXISTS:      s = "MODDEVEXISTS"; break;
                case MODDEVREADDIR:     s = "MODDEVREADDIR"; break;
                case MODDEVEMPTYDIR:    s = "MODDEVEMPTYDIR"; break;
                case MODDEVNAME:        s = "MODDEVNAME"; break;
                case MODGETDEVFSPATH_MI_LEN:
                                        s = "MODGETDEVFSPATH_MI_LEN"; break;
                case MODGETDEVFSPATH_MI:
                                        s = "MODGETDEVFSPATH_MI"; break;
                case MODREMDRVALIAS:    s = "MODREMDRVALIAS"; break;
                case MODHPOPS:  s = "MODHPOPS"; break;
                }
        }

        if (s == NULL)
                prt_dec(pri, 0, val);
        else
                outstring(pri, s);
}

void
prt_acl(private_t *pri, int raw, long val)      /* print acl() code */
{
        const char *s = NULL;

        if (!raw) {
                switch (val) {
                case GETACL:            s = "GETACL";           break;
                case SETACL:            s = "SETACL";           break;
                case GETACLCNT:         s = "GETACLCNT";        break;
                case ACE_GETACL:        s = "ACE_GETACL";       break;
                case ACE_SETACL:        s = "ACE_SETACL";       break;
                case ACE_GETACLCNT:     s = "ACE_GETACLCNT";    break;
                }
        }

        if (s == NULL)
                prt_dec(pri, 0, val);
        else
                outstring(pri, s);
}

void
```

```
1181 prt_aio(private_t *pri, int raw, long val)      /* print kaio() code */
1182 {
1183         const char *s = NULL;
1184         char buf[32];

1186         if (!raw) {
1187                 switch (val & ~AIO_POLL_BIT) {
1188                 case AIOREAD:           s = "AIOREAD";          break;
1189                 case AIOWRITE:          s = "AIOWRITE";         break;
1190                 case AIOWAIT:           s = "AIOWAIT";          break;
1191                 case AIOCANCEL:         s = "AIOCANCEL";        break;
1192                 case AIONOTIFY:         s = "AIONOTIFY";        break;
1193                 case AIOINIT:           s = "AIOINIT";          break;
1194                 case AIOSTART:          s = "AIOSTART";         break;
1195                 case AIOLIO:            s = "AIOLIO";           break;
1196                 case AIOSUSPEND:        s = "AIOSUSPEND";       break;
1197                 case AIOERROR:          s = "AIOERROR";         break;
1198                 case AIOLIOWAIT:        s = "AIOLIOWAIT";       break;
1199                 case AIOAREAD:          s = "AIOAREAD";         break;
1200                 case AIOAWRITE:         s = "AIOAWRITE";        break;
1201                 /*
1202                  * We have to hardcode the values for the 64-bit versions of
1203                  * these calls, because <sys/aio.h> defines them to be identical
1204                  * when compiled 64-bit.  If our target is 32-bit, we still need
1205                  * to decode them correctly.
1206                  */
1207                 case 13:                s = "AIOLIO64";         break;
1208                 case 14:                s = "AIOSUSPEND64";     break;
1209                 case 15:                s = "AUIOERROR64";      break;
1210                 case 16:                s = "AIOLIOWAIT64";     break;
1211                 case 17:                s = "AIOAREAD64";       break;
1212                 case 18:                s = "AIOAWRITE64";      break;
1213                 case 19:                s = "AIOCANCEL64";      break;

1215                 /*
1216                  * AIOFSYNC doesn't correspond to a syscall.
1217                  */
1218                 case AIOWAITN:          s = "AIOWAITN";         break;
1219                 }
1220                 if (s != NULL && (val & AIO_POLL_BIT)) {
1221                         (void) strlcpy(buf, s, sizeof (buf));
1222                         (void) strlcat(buf, "|AIO_POLL_BIT", sizeof (buf));
1223                         s = (const char *)buf;
1224                 }
1225         }

1227         if (s == NULL)
1228                 prt_dec(pri, 0, val);
1229         else
1230                 outstring(pri, s);
1231 }

1233 void
1234 prt_aud(private_t *pri, int raw, long val)      /* print auditsys() code */
1235 {
1236         const char *s = NULL;

1238         if (!raw) {
1239                 switch (val) {
1240                 case BSM_GETAUID:       s = "BSM_GETAUID";      break;
1241                 case BSM_SETAUID:       s = "BSM_SETAUID";      break;
1242                 case BSM_GETAUDIT:      s = "BSM_GETAUDIT";     break;
1243                 case BSM_SETAUDIT:      s = "BSM_SETAUDIT";     break;
1244                 case BSM_AUDIT:         s = "BSM_AUDIT";        break;
1245                 case BSM_AUDITCTL:      s = "BSM_AUDITCTL";     break;
1246                 case BSM_GETAUDIT_ADDR: s = "BSM_GETAUDIT_ADDR"; break;
```



```
1247                 case BSM_SETAUDIT_ADDR: s = "BSM_SETAUDIT_ADDR"; break;
1248                 }
1249         }

1251         if (s == NULL)
1252                 prt_dec(pri, 0, val);
1253         else
1254                 outstring(pri, s);
1255 }

1257 void
1258 prt_cor(private_t *pri, int raw, long val)      /* print corectl() subcode */
1259 {
1260         const char *s = NULL;

1262         if (!raw) {
1263                 switch (val) {
1264                 case CC_SET_OPTIONS:
1265                         s = "CC_SET_OPTIONS";           break;
1266                 case CC_GET_OPTIONS:
1267                         s = "CC_GET_OPTIONS";           break;
1268                 case CC_SET_GLOBAL_PATH:
1269                         s = "CC_SET_GLOBAL_PATH";       break;
1270                 case CC_GET_GLOBAL_PATH:
1271                         s = "CC_GET_GLOBAL_PATH";       break;
1272                 case CC_SET_PROCESS_PATH:
1273                         s = "CC_SET_PROCESS_PATH";      break;
1274                 case CC_GET_PROCESS_PATH:
1275                         s = "CC_GET_PROCESS_PATH";      break;
1276                 case CC_SET_GLOBAL_CONTENT:
1277                         s = "CC_SET_GLOBAL_CONTENT";    break;
1278                 case CC_GET_GLOBAL_CONTENT:
1279                         s = "CC_GET_GLOBAL_CONTENT";    break;
1280                 case CC_SET_PROCESS_CONTENT:
1281                         s = "CC_SET_PROCESS_CONTENT";   break;
1282                 case CC_GET_PROCESS_CONTENT:
1283                         s = "CC_GET_PROCESS_CONTENT";   break;
1284                 case CC_SET_DEFAULT_PATH:
1285                         s = "CC_SET_DEFAULT_PATH";      break;
1286                 case CC_GET_DEFAULT_PATH:
1287                         s = "CC_GET_DEFAULT_PATH";      break;
1288                 case CC_SET_DEFAULT_CONTENT:
1289                         s = "CC_SET_DEFAULT_CONTENT";   break;
1290                 case CC_GET_DEFAULT_CONTENT:
1291                         s = "CC_GET_DEFAULT_CONTENT";   break;
1292                 }
1293         }

1295         if (s == NULL)
1296                 prt_dec(pri, 0, val);
1297         else
1298                 outstring(pri, s);
1299 }

1301 void
1302 prt_cco(private_t *pri, int raw, long val)      /* print corectl() options */
1303 {
1304         char *s;

1306         if (val == 0)
1307                 prt_dec(pri, 0, val);
1308         else if (raw || (val & ~CC_OPTIONS))
1309                 prt_hhx(pri, 0, val);
1310         else {
1311 #define CBSIZE  sizeof (pri->code_buf)
1312                 s = pri->code_buf;
```

```
1313                *s = '\0';
1314                if (val & CC_GLOBAL_PATH)
1315                        (void) strlcat(s, "|CC_GLOBAL_PATH", CBSIZE);
1316                if (val & CC_PROCESS_PATH)
1317                        (void) strlcat(s, "|CC_PROCESS_PATH", CBSIZE);
1318                if (val & CC_GLOBAL_SETID)
1319                        (void) strlcat(s, "|CC_GLOBAL_SETID", CBSIZE);
1320                if (val & CC_PROCESS_SETID)
1321                        (void) strlcat(s, "|CC_PROCESS_SETID", CBSIZE);
1322                if (val & CC_GLOBAL_LOG)
1323                        (void) strlcat(s, "|CC_GLOBAL_LOG", CBSIZE);
1324                if (*s == '\0')
1325                        prt_hhx(pri, 0, val);
1326                else
1327                        outstring(pri, ++s);
1328 #undef CBSIZE
1329        }
1330 }

1332 void
1333 prt_ccc(private_t *pri, int raw, long val)      /* print corectl() content */
1334 {
1335        core_content_t ccc;

1337        if (Pread(Proc, &ccc, sizeof (ccc), val) != sizeof (ccc))
1338                prt_hex(pri, 0, val);
1339        else if (!raw && proc_content2str(ccc, pri->code_buf,
1340            sizeof (pri->code_buf)) >= 0)
1341                outstring(pri, pri->code_buf);
1342        else
1343                prt_hhx(pri, 0, (long)ccc);
1344 }

1346 void
1347 prt_rcc(private_t *pri, int raw, long val)      /* print corectl() ret. cont. */
1348 {
1349        core_content_t ccc;

1351        if (pri->Errno || Pread(Proc, &ccc, sizeof (ccc), val) != sizeof (ccc))
1352                prt_hex(pri, 0, val);
1353        else if (!raw && proc_content2str(ccc, pri->code_buf,
1354            sizeof (pri->code_buf)) >= 0)
1355                outstring(pri, pri->code_buf);
1356        else
1357                prt_hhx(pri, 0, (long)ccc);
1358 }

1360 void
1361 prt_cpc(private_t *pri, int raw, long val)      /* print cpc() subcode */
1362 {
1363        const char *s = NULL;

1365        if (!raw) {
1366                switch (val) {
1367                case CPC_BIND:          s = "CPC_BIND";         break;
1368                case CPC_SAMPLE:        s = "CPC_SAMPLE";       break;
1369                case CPC_INVALIDATE:    s = "CPC_INVALIDATE";   break;
1370                case CPC_RELE:          s = "CPC_RELE";         break;
1371                case CPC_EVLIST_SIZE:   s = "CPC_EVLIST_SIZE";  break;
1372                case CPC_LIST_EVENTS:   s = "CPC_LIST_EVENTS";  break;
1373                case CPC_ATTRLIST_SIZE: s = "CPC_ATTRLIST_SIZE"; break;
1374                case CPC_LIST_ATTRS:    s = "CPC_LIST_ATTRS";   break;
1375                case CPC_IMPL_NAME:     s = "CPC_IMPL_NAME";    break;
1376                case CPC_CPUREF:        s = "CPC_CPUREF";       break;
1377                case CPC_USR_EVENTS:    s = "CPC_USR_EVENTS";   break;
1378                case CPC_SYS_EVENTS:    s = "CPC_SYS_EVENTS";   break;
```



```
1379                case CPC_NPIC:          s = "CPC_NPIC";         break;
1380                case CPC_CAPS:          s = "CPC_CAPS";         break;
1381                case CPC_ENABLE:        s = "CPC_ENABLE";       break;
1382                case CPC_DISABLE:       s = "CPC_DISABLE";      break;
1383                }
1384        }

1386        if (s == NULL)
1387                prt_dec(pri, 0, val);
1388        else
1389                outstring(pri, s);
1390 }

1392 void
1393 outstring(private_t *pri, const char *s)
1394 {
1395        int len = strlen(s);

1397        GROW(len);
1398        (void) strcpy(pri->sys_string + pri->sys_leng, s);
1399        pri->sys_leng += len;
1400 }

1402 void
1403 grow(private_t *pri, int nbyte) /* reallocate format buffer if necessary */
1404 {
1405        while (pri->sys_leng + nbyte >= pri->sys_ssize)
1406                pri->sys_string = my_realloc(pri->sys_string,
1407                    pri->sys_ssize *= 2, "format buffer");
1408 }

1410 void
1411 prt_clc(private_t *pri, int raw, long val)
1412 {
1413        const char *s = NULL;

1415        if (!raw) {
1416                switch (val) {
1417                case CL_INITIALIZE:     s = "CL_INITIALIZE";    break;
1418                case CL_CONFIG:         s = "CL_CONFIG";        break;
1419                }
1420        }

1422        if (s == NULL)
1423                prt_dec(pri, 0, val);
1424        else
1425                outstring(pri, s);
1426 }

1428 void
1429 prt_clf(private_t *pri, int raw, long val)
1430 {
1431        const char *s = NULL;

1433        if (!raw) {
1434                switch (pri->sys_args[0]) {
1435                case CL_CONFIG:
1436                        switch (pri->sys_args[1]) {
1437                        case CL_NODEID:
1438                                s = "CL_NODEID";                break;
1439                        case CL_HIGHEST_NODEID:
1440                                s = "CL_HIGHEST_NODEID";        break;
1441                        }
1442                        break;
1443                case CL_INITIALIZE:
1444                        switch (pri->sys_args[1]) {
```

```
1445                        case CL_GET_BOOTFLAG:
1446                                s = "CL_GET_BOOTFLAG";          break;
1447                        }
1448                        break;
1449                }
1450        }

1452        if (s == NULL)
1453                prt_dec(pri, 0, val);
1454        else
1455                outstring(pri, s);
1456 }

1458 void
1459 prt_sqc(private_t *pri, int raw, long val)      /* print sigqueue() si_code */
1460 {
1461        const char *s = NULL;

1463        if (!raw) {
1464                switch ((int)val) {
1465                case SI_QUEUE:          s = "SI_QUEUE";         break;
1466                case SI_TIMER:          s = "SI_TIMER";         break;
1467                case SI_ASYNCIO:        s = "SI_ASYNCIO";       break;
1468                case SI_MESGQ:          s = "SI_MESGQ";         break;
1469                }
1470        }

1472        if (s == NULL)
1473                prt_dec(pri, 0, val);
1474        else
1475                outstring(pri, s);
1476 }

1478 /*
1479  * print priocntlsys() (key, value) pair key.
1480  */
1481 void
1482 print_pck(private_t *pri, int raw, long val)
1483 {
1484        const char      *s = NULL;
1485        char            clname[PC_CLNMSZ];

1487        if ((pri->sys_args[2] != PC_GETXPARMS &&
1488            pri->sys_args[2] != PC_SETXPARMS) || val == 0 || raw) {
1489                prt_dec(pri, 0, val);
1490                return;
1491        }

1493        if (pri->sys_args[3] == NULL) {
1494                if (val == PC_KY_CLNAME) {
1495                        s = "PC_KY_CLNAME";
1496                        outstring(pri, s);
1497                } else
1498                        prt_dec(pri, 0, val);
1499                return;
1500        }

1502        if (Pread(Proc, &clname, PC_CLNMSZ, pri->sys_args[3]) != PC_CLNMSZ) {
1503                prt_dec(pri, 0, val);
1504                return;
1505        }

1507        if (strcmp(clname, "TS") == 0) {
1508                switch (val) {
1509                case TS_KY_UPRILIM:     s = "TS_KY_UPRILIM";    break;
1510                case TS_KY_UPRI:        s = "TS_KY_UPRI";       break;
```



```
1511                default:                                        break;
1512                }
1513        } else if (strcmp(clname, "IA") == 0) {
1514                switch (val) {
1515                case IA_KY_UPRILIM:     s = "IA_KY_UPRILIM";    break;
1516                case IA_KY_UPRI:        s = "IA_KY_UPRI";       break;
1517                case IA_KY_MODE:        s = "IA_KY_MODE";       break;
1518                default:                                        break;
1519                }
1520        } else if (strcmp(clname, "RT") == 0) {
1521                switch (val) {
1522                case RT_KY_PRI:         s = "RT_KY_PRI";        break;
1523                case RT_KY_TQSECS:      s = "RT_KY_TQSECS";     break;
1524                case RT_KY_TQNSECS:     s = "RT_KY_TQNSECS";    break;
1525                case RT_KY_TQSIG:       s = "RT_KY_TQSIG";      break;
1526                default:                                        break;
1527                }
1528        } else if (strcmp(clname, "FSS") == 0) {
1529                switch (val) {
1530                case FSS_KY_UPRILIM:    s = "FSS_KY_UPRILIM";   break;
1531                case FSS_KY_UPRI:       s = "FSS_KY_UPRI";      break;
1532                default:                                        break;
1533                }
1534        } else if (strcmp(clname, "FX") == 0) {
1535                switch (val) {
1536                case FX_KY_UPRILIM:     s = "FX_KY_UPRILIM";    break;
1537                case FX_KY_UPRI:        s = "FX_KY_UPRI";       break;
1538                case FX_KY_TQSECS:      s = "FX_KY_TQSECS";     break;
1539                case FX_KY_TQNSECS:     s = "FX_KY_TQNSECS";    break;
1540                default:                                        break;
1541                }
1542        }

1544        if (s == NULL)
1545                prt_dec(pri, 0, val);
1546        else
1547                outstring(pri, s);
1548 }

1550 /*
1551  * print priocntlsys() fourth argument.
1552  */
1553 /*ARGSUSED*/
1554 void
1555 prt_pc4(private_t *pri, int raw, long val)
1556 {
1557        /* look at pricntlsys function */
1558        if ((pri->sys_args[2] != PC_GETXPARMS &&
1559            pri->sys_args[2] != PC_SETXPARMS))
1560                prt_hex(pri, 0, val);
1561        else if (val)
1562                prt_stg(pri, 0, val);
1563        else
1564                prt_dec(pri, 0, val);
1565 }

1567 /*
1568  * print priocntlsys() (key, value) pairs (5th argument).
1569  */
1570 /*ARGSUSED*/
1571 void
1572 prt_pc5(private_t *pri, int raw, long val)
1573 {
1574        pc_vaparms_t    prms;
1575        pc_vaparm_t     *vpp = &prms.pc_parms[0];
1576        uint_t          cnt;
```

```
1579         /* look at pricntlsys function */
1580         if ((pri->sys_args[2] != PC_GETXPARMS &&
1581             pri->sys_args[2] != PC_SETXPARMS) || val == 0) {
1582                 prt_dec(pri, 0, 0);
1583                 return;
1584         }

1586         if (Pread(Proc, &prms, sizeof (prms), val) != sizeof (prms)) {
1587                 prt_hex(pri, 0, val);
1588                 return;
1589         }

1591         if ((cnt = prms.pc_vaparmscnt) > PC_VAPARMCNT)
1592                 return;

1594         for (; cnt--; vpp++) {
1595                 print_pck(pri, 0, vpp->pc_key);
1596                 outstring(pri, ", ");
1597                 prt_hex(pri, 0, (long)vpp->pc_parm);
1598                 outstring(pri, ", ");
1599         }

1601         prt_dec(pri, 0, PC_KY_NULL);
1602 }

1604 /*
1605  * Print a psecflags(2) command
1606  */
1607 void
1608 prt_psfcmd(private_t *pri, int raw, long val)
1609 {
1610         const char *s = NULL;

1612         if (raw == 0) {
1613                 switch ((psecflags_cmd_t)val) {
1614                 case PSECFLAGS_SET:
1615                         s = "PSECFLAGS_SET";
1616                         break;
1617                 case PSECFLAGS_DISABLE:
1618                         s = "PSECFLAGS_DISABLE";
1619                         break;
1620                 case PSECFLAGS_ENABLE:
1621                         s = "PSECFLAGS_ENABLE";
1622                         break;
1623                 }
1624         }

1626         if (s == NULL)
1627                 prt_dec(pri, 0, val);
1628         else
1629                 outstring(pri, s);
1630 }

1632 void
1633 prt_psflags(private_t *pri, int raw, long val)
1634 {
1635         char *str = pri->code_buf;

1637         if (raw == 1) {
1638                 prt_hex(pri, 0, val);
1639                 return;
1640         }

1642         *str = '\0';
```



```
1643         if (val & PROC_SEC_ASLR) {
1644                 (void) strlcat(str, "|PROC_SEC_ASLR", sizeof (pri->code_buf));
1645                 val &= ~PROC_SEC_ASLR;
1646         }

1648         if (val != 0)
1649                 (void) snprintf(str, sizeof (pri->code_buf), "%s|%x", str, val);

1651         outstring(pri, str + 1);
1652 }

1654 /*
1655 #endif /* ! codereview */
1656  * Print processor set id, including logical expansion of "special" ids.
1657  */
1658 void
1659 prt_pst(private_t *pri, int raw, long val)
1660 {
1661         const char *s = NULL;

1663         if (!raw) {
1664                 switch ((psetid_t)val) {
1665                 case PS_NONE:           s = "PS_NONE";          break;
1666                 case PS_QUERY:          s = "PS_QUERY";         break;
1667                 case PS_MYID:           s = "PS_MYID";          break;
1668                 }
1669         }

1671         if (s == NULL)
1672                 prt_dec(pri, 0, val);
1673         else
1674                 outstring(pri, s);
1675 }

1677 /*
1678  * Print meminfo() argument.
1679  */
1680 /*ARGSUSED*/
1681 void
1682 prt_mif(private_t *pri, int raw, long val)
1683 {
1684         struct meminfo  minfo;

1686 #ifdef _LP64
1687         if (data_model == PR_MODEL_ILP32) {
1688                 struct meminfo32 minfo32;

1690                 if (Pread(Proc, &minfo32, sizeof (struct meminfo32), val) !=
1691                         sizeof (struct meminfo32)) {
1692                         prt_dec(pri, 0, pri->sys_args[1]);      /* addr_count */
1693                         outstring(pri, ", ");
1694                         prt_hex(pri, 0, val);
1695                         return;
1696                 }
1697                 /*
1698                  * arrange the arguments in the order that user calls with
1699                  */
1700                 prt_hex(pri, 0, minfo32.mi_inaddr);
1701                 outstring(pri, ", ");
1702                 prt_dec(pri, 0, pri->sys_args[1]);      /* addr_count */
1703                 outstring(pri, ", ");
1704                 prt_hex(pri, 0, minfo32.mi_info_req);
1705                 outstring(pri, ", ");
1706                 prt_dec(pri, 0, minfo32.mi_info_count);
1707                 outstring(pri, ", ");
1708                 prt_hex(pri, 0, minfo32.mi_outdata);
```

```
1709                 outstring(pri, ", ");
1710                 prt_hex(pri, 0, minfo32.mi_validity);
1711                 return;
1712         }
1713 #endif
1714         if (Pread(Proc, &minfo, sizeof (struct meminfo), val) !=
1715                 sizeof (struct meminfo)) {
1716                 prt_dec(pri, 0, pri->sys_args[1]);      /* addr_count */
1717                 outstring(pri, ", ");
1718                 prt_hex(pri, 0, val);
1719                 return;
1720         }
1721         /*
1722          * arrange the arguments in the order that user calls with
1723          */
1724         prt_hex(pri, 0, (long)minfo.mi_inaddr);
1725         outstring(pri, ", ");
1726         prt_dec(pri, 0, pri->sys_args[1]);      /* addr_count */
1727         outstring(pri, ", ");
1728         prt_hex(pri, 0, (long)minfo.mi_info_req);
1729         outstring(pri, ", ");
1730         prt_dec(pri, 0, minfo.mi_info_count);
1731         outstring(pri, ", ");
1732         prt_hex(pri, 0, (long)minfo.mi_outdata);
1733         outstring(pri, ", ");
1734         prt_hex(pri, 0, (long)minfo.mi_validity);
1735 }


1738 /*
1739  * Print so_socket() 1st argument.
1740  */
1741 /*ARGSUSED*/
1742 void
1743 prt_pfm(private_t *pri, int raw, long val)
1744 {
1745         /* Protocol Families have same names as Address Families */
1746         if ((ulong_t)val < MAX_AFCODES) {
1747                 outstring(pri, "PF_");
1748                 outstring(pri, afcodes[val]);
1749         } else {
1750                 prt_dec(pri, 0, val);
1751         }
1752 }

1754 /*
1755  * Print sockconfig() subcode.
1756  */
1757 /*ARGSUSED*/
1758 void
1759 prt_skc(private_t *pri, int raw, long val)
1760 {
1761         const char *s = NULL;

1763         if (!raw) {
1764                 switch (val) {
1765                 case SOCKCONFIG_ADD_SOCK:
1766                         s = "SOCKCONFIG_ADD_SOCK"; break;
1767                 case SOCKCONFIG_REMOVE_SOCK:
1768                         s = "SOCKCONFIG_REMOVE_SOCK"; break;
1769                 case SOCKCONFIG_ADD_FILTER:
1770                         s = "SOCKCONFIG_ADD_FILTER"; break;
1771                 case SOCKCONFIG_REMOVE_FILTER:
1772                         s = "SOCKCONFIG_REMOVE_FILTER"; break;
1773                 }
1774         }
```



```
1775         if (s == NULL)
1776                 prt_dec(pri, 0, val);
1777         else
1778                 outstring(pri, s);
1779 }
1780 /*
1781  * Print so_socket() 2nd argument.
1782  */
1783 /*ARGSUSED*/
1784 void
1785 prt_skt(private_t *pri, int raw, long val)
1786 {
1787         const char *s;
1788         long type = val & SOCK_TYPE_MASK;

1790         if ((ulong_t)type <= MAX_SOCKTYPES &&
1791             (s = socktype_codes[type]) != NULL) {
1792                 outstring(pri, s);
1793                 if ((val & SOCK_CLOEXEC) != 0) {
1794                         outstring(pri, "|SOCK_CLOEXEC");
1795                 }
1796         } else {
1797                 prt_dec(pri, 0, val);
1798         }
1799 }


1802 /*
1803  * Print so_socket() 3rd argument.
1804  */
1805 /*ARGSUSED*/
1806 void
1807 prt_skp(private_t *pri, int raw, long val)
1808 {
1809         const char *s;

1811         /* cheating -- look at the protocol-family */
1812         switch (pri->sys_args[0]) {
1813         case PF_INET6:
1814         case PF_INET:
1815         case PF_NCA:    if ((s = ipprotos((int)val)) != NULL) {
1816                                 outstring(pri, s);
1817                                 break;
1818                         }
1819                         /* FALLTHROUGH */
1820         default:        prt_dec(pri, 0, val);
1821                         break;
1822         }
1823 }


1826 /*
1827  * Print so_socket() 5th argument.
1828  */
1829 /*ARGSUSED*/
1830 void
1831 prt_skv(private_t *pri, int raw, long val)
1832 {
1833         switch (val) {
1834         case SOV_STREAM:        outstring(pri, "SOV_STREAM");   break;
1835         case SOV_DEFAULT:       outstring(pri, "SOV_DEFAULT");  break;
1836         case SOV_SOCKSTREAM:    outstring(pri, "SOV_SOCKSTREAM");       break;
1837         case SOV_SOCKBSD:       outstring(pri, "SOV_SOCKBSD");  break;
1838         case SOV_XPG4_2:        outstring(pri, "SOV_XPG4_2");   break;
1839         default:                prt_dec(pri, 0, val);           break;
1840         }
```

```
}

/*
 * Print accept4() flags argument.
 */
void
prt_acf(private_t *pri, int raw, long val)
{
        int first = 1;
        if (raw || !val ||
            (val & ~(SOCK_CLOEXEC|SOCK_NDELAY|SOCK_NONBLOCK))) {
                prt_dex(pri, 0, val);
                return;
        }

        if (val & SOCK_CLOEXEC) {
                outstring(pri, "|SOCK_CLOEXEC" + first);
                first = 0;
        }
        if (val & SOCK_NDELAY) {
                outstring(pri, "|SOCK_NDELAY" + first);
                first = 0;
        }
        if (val & SOCK_NONBLOCK) {
                outstring(pri, "|SOCK_NONBLOCK" + first);
        }
}


/*
 * Print setsockopt()/getsockopt() 2nd argument.
 */
/*ARGSUSED*/
void
prt_sol(private_t *pri, int raw, long val)
{
        if (val == SOL_SOCKET) {
                outstring(pri, "SOL_SOCKET");
        } else if (val == SOL_ROUTE) {
                outstring(pri, "SOL_ROUTE");
        } else {
                const struct protoent *p;
                struct protoent res;
                char buf[NSS_BUFLEN_PROTOCOLS];

                if ((p = getprotobynumber_r(val, &res,
                    (char *)buf, sizeof (buf))) != NULL)
                        outstring(pri, p->p_name);
                else
                        prt_dec(pri, 0, val);
        }
}


const char *
sol_optname(private_t *pri, long val)
{
#define CBSIZE  sizeof (pri->code_buf)
        if (val >= SO_SNDBUF) {
                switch (val) {
                case SO_SNDBUF:         return ("SO_SNDBUF");
                case SO_RCVBUF:         return ("SO_RCVBUF");
                case SO_SNDLOWAT:       return ("SO_SNDLOWAT");
                case SO_RCVLOWAT:       return ("SO_RCVLOWAT");
                case SO_SNDTIMEO:       return ("SO_SNDTIMEO");
                case SO_RCVTIMEO:       return ("SO_RCVTIMEO");
```



```
                case SO_ERROR:          return ("SO_ERROR");
                case SO_TYPE:           return ("SO_TYPE");
                case SO_PROTOTYPE:      return ("SO_PROTOTYPE");
                case SO_ANON_MLP:       return ("SO_ANON_MLP");
                case SO_MAC_EXEMPT:     return ("SO_MAC_EXEMPT");
                case SO_ALLZONES:       return ("SO_ALLZONES");
                case SO_MAC_IMPLICIT:   return ("SO_MAC_IMPLICIT");
                case SO_VRRP:           return ("SO_VRRP");
                case SO_EXCLBIND:       return ("SO_EXCLBIND");
                case SO_DOMAIN:         return ("SO_DOMAIN");

                default:                (void) snprintf(pri->code_buf, CBSIZE,
                                            "0x%lx", val);
                                        return (pri->code_buf);
                }
        } else {
                char *s = pri->code_buf;
                size_t used = 1;
                long val2;

                *s = '\0';
                val2 = val & ~(SO_DEBUG|SO_ACCEPTCONN|SO_REUSEADDR|SO_KEEPALIVE|
                    SO_DONTROUTE|SO_BROADCAST|SO_USELOOPBACK|SO_LINGER|
                    SO_OOBINLINE|SO_DGRAM_ERRIND|SO_RECVUCRED);
                if (val2)
                        used = snprintf(s, CBSIZE, "|0x%lx", val2);
                if (val & SO_DEBUG)
                        used = strlcat(s, "|SO_DEBUG", CBSIZE);
                if (val & SO_ACCEPTCONN)
                        used = strlcat(s, "|SO_ACCEPTCONN", CBSIZE);
                if (val & SO_REUSEADDR)
                        used = strlcat(s, "|SO_REUSEADDR", CBSIZE);
                if (val & SO_KEEPALIVE)
                        used = strlcat(s, "|SO_KEEPALIVE", CBSIZE);
                if (val & SO_DONTROUTE)
                        used = strlcat(s, "|SO_DONTROUTE", CBSIZE);
                if (val & SO_BROADCAST)
                        used = strlcat(s, "|SO_BROADCAST", CBSIZE);
                if (val & SO_USELOOPBACK)
                        used = strlcat(s, "|SO_USELOOPBACK", CBSIZE);
                if (val & SO_LINGER)
                        used = strlcat(s, "|SO_LINGER", CBSIZE);
                if (val & SO_OOBINLINE)
                        used = strlcat(s, "|SO_OOBINLINE", CBSIZE);
                if (val & SO_DGRAM_ERRIND)
                        used = strlcat(s, "|SO_DGRAM_ERRIND", CBSIZE);
                if (val & SO_RECVUCRED)
                        used = strlcat(s, "|SO_RECVUCRED", CBSIZE);
                if (used >= CBSIZE || val == 0)
                        (void) snprintf(s + 1, CBSIZE-1, "0x%lx", val);
                return ((const char *)(s + 1));
        }
#undef CBSIZE
}

const char *
route_optname(private_t *pri, long val)
{
        switch (val) {
        case RT_AWARE:
                return ("RT_AWARE");
        default:
                (void) snprintf(pri->code_buf, sizeof (pri->code_buf),
                    "0x%lx", val);
                return (pri->code_buf);
        }
```

```
1973 }
1975 const char *
1976 tcp_optname(private_t *pri, long val)
1977 {
1978         switch (val) {
1979         case TCP_NODELAY:               return ("TCP_NODELAY");
1980         case TCP_MAXSEG:                return ("TCP_MAXSEG");
1981         case TCP_KEEPALIVE:             return ("TCP_KEEPALIVE");
1982         case TCP_NOTIFY_THRESHOLD:      return ("TCP_NOTIFY_THRESHOLD");
1983         case TCP_ABORT_THRESHOLD:       return ("TCP_ABORT_THRESHOLD");
1984         case TCP_CONN_NOTIFY_THRESHOLD: return ("TCP_CONN_NOTIFY_THRESHOLD");
1985         case TCP_CONN_ABORT_THRESHOLD:  return ("TCP_CONN_ABORT_THRESHOLD");
1986         case TCP_RECVDSTADDR:           return ("TCP_RECVDSTADDR");
1987         case TCP_ANONPRIVBIND:          return ("TCP_ANONPRIVBIND");
1988         case TCP_EXCLBIND:              return ("TCP_EXCLBIND");
1989         case TCP_INIT_CWND:             return ("TCP_INIT_CWND");
1990         case TCP_KEEPALIVE_THRESHOLD:   return ("TCP_KEEPALIVE_THRESHOLD");
1991         case TCP_KEEPALIVE_ABORT_THRESHOLD:
1992                 return ("TCP_KEEPALIVE_ABORT_THRESHOLD");
1993         case TCP_CORK:                  return ("TCP_CORK");
1994         case TCP_RTO_INITIAL:           return ("TCP_RTO_INITIAL");
1995         case TCP_RTO_MIN:               return ("TCP_RTO_MIN");
1996         case TCP_RTO_MAX:               return ("TCP_RTO_MAX");
1997         case TCP_LINGER2:               return ("TCP_LINGER2");

1999         default:                        (void) snprintf(pri->code_buf,
2000                                             sizeof (pri->code_buf),
2001                                             "0x%lx", val);
2002                                         return (pri->code_buf);
2003         }
2004 }


2007 const char *
2008 sctp_optname(private_t *pri, long val)
2009 {
2010         switch (val) {
2011         case SCTP_RTOINFO:              return ("SCTP_RTOINFO");
2012         case SCTP_ASSOCINFO:            return ("SCTP_ASSOCINFO");
2013         case SCTP_INITMSG:              return ("SCTP_INITMSG");
2014         case SCTP_NODELAY:              return ("SCTP_NODELAY");
2015         case SCTP_AUTOCLOSE:            return ("SCTP_AUTOCLOSE");
2016         case SCTP_SET_PEER_PRIMARY_ADDR:
2017                 return ("SCTP_SET_PEER_PRIMARY_ADDR");
2018         case SCTP_PRIMARY_ADDR:         return ("SCTP_PRIMARY_ADDR");
2019         case SCTP_ADAPTATION_LAYER:     return ("SCTP_ADAPTATION_LAYER");
2020         case SCTP_DISABLE_FRAGMENTS:    return ("SCTP_DISABLE_FRAGMENTS");
2021         case SCTP_PEER_ADDR_PARAMS:     return ("SCTP_PEER_ADDR_PARAMS");
2022         case SCTP_DEFAULT_SEND_PARAM:   return ("SCTP_DEFAULT_SEND_PARAM");
2023         case SCTP_EVENTS:               return ("SCTP_EVENTS");
2024         case SCTP_I_WANT_MAPPED_V4_ADDR:
2025                 return ("SCTP_I_WANT_MAPPED_V4_ADDR");
2026         case SCTP_MAXSEG:               return ("SCTP_MAXSEG");
2027         case SCTP_STATUS:               return ("SCTP_STATUS");
2028         case SCTP_GET_PEER_ADDR_INFO:   return ("SCTP_GET_PEER_ADDR_INFO");

2030         case SCTP_ADD_ADDR:             return ("SCTP_ADD_ADDR");
2031         case SCTP_REM_ADDR:             return ("SCTP_REM_ADDR");

2033         default:                        (void) snprintf(pri->code_buf,
2034                                             sizeof (pri->code_buf),
2035                                             "0x%lx", val);
2036                                         return (pri->code_buf);
2037         }
2038 }
```



```
2041 const char *
2042 udp_optname(private_t *pri, long val)
2043 {
2044         switch (val) {
2045         case UDP_CHECKSUM:              return ("UDP_CHECKSUM");
2046         case UDP_ANONPRIVBIND:          return ("UDP_ANONPRIVBIND");
2047         case UDP_EXCLBIND:              return ("UDP_EXCLBIND");
2048         case UDP_RCVHDR:                return ("UDP_RCVHDR");
2049         case UDP_NAT_T_ENDPOINT:        return ("UDP_NAT_T_ENDPOINT");

2051         default:                        (void) snprintf(pri->code_buf,
2052                                             sizeof (pri->code_buf), "0x%lx",
2053                                             val);
2054                                         return (pri->code_buf);
2055         }
2056 }


2059 /*
2060  * Print setsockopt()/getsockopt() 3rd argument.
2061  */
2062 /*ARGSUSED*/
2063 void
2064 prt_son(private_t *pri, int raw, long val)
2065 {
2066         /* cheating -- look at the level */
2067         switch (pri->sys_args[1]) {
2068         case SOL_SOCKET:        outstring(pri, sol_optname(pri, val));
2069                                 break;
2070         case SOL_ROUTE:         outstring(pri, route_optname(pri, val));
2071                                 break;
2072         case IPPROTO_TCP:       outstring(pri, tcp_optname(pri, val));
2073                                 break;
2074         case IPPROTO_UDP:       outstring(pri, udp_optname(pri, val));
2075                                 break;
2076         case IPPROTO_SCTP:      outstring(pri, sctp_optname(pri, val));
2077                                 break;
2078         default:                prt_dec(pri, 0, val);
2079                                 break;
2080         }
2081 }


2084 /*
2085  * Print utrap type
2086  */
2087 /*ARGSUSED*/
2088 void
2089 prt_utt(private_t *pri, int raw, long val)
2090 {
2091         const char *s = NULL;

2093 #ifdef __sparc
2094         if (!raw) {
2095                 switch (val) {
2096                 case UT_INSTRUCTION_DISABLED:
2097                         s = "UT_INSTRUCTION_DISABLED"; break;
2098                 case UT_INSTRUCTION_ERROR:
2099                         s = "UT_INSTRUCTION_ERROR"; break;
2100                 case UT_INSTRUCTION_PROTECTION:
2101                         s = "UT_INSTRUCTION_PROTECTION"; break;
2102                 case UT_ILLTRAP_INSTRUCTION:
2103                         s = "UT_ILLTRAP_INSTRUCTION"; break;
2104                 case UT_ILLEGAL_INSTRUCTION:
```

```
2105                        s = "UT_ILLEGAL_INSTRUCTION"; break;
2106                case UT_PRIVILEGED_OPCODE:
2107                        s = "UT_PRIVILEGED_OPCODE"; break;
2108                case UT_FP_DISABLED:
2109                        s = "UT_FP_DISABLED"; break;
2110                case UT_FP_EXCEPTION_IEEE_754:
2111                        s = "UT_FP_EXCEPTION_IEEE_754"; break;
2112                case UT_FP_EXCEPTION_OTHER:
2113                        s = "UT_FP_EXCEPTION_OTHER"; break;
2114                case UT_TAG_OVERFLOW:
2115                        s = "UT_TAG_OVERFLOW"; break;
2116                case UT_DIVISION_BY_ZERO:
2117                        s = "UT_DIVISION_BY_ZERO"; break;
2118                case UT_DATA_EXCEPTION:
2119                        s = "UT_DATA_EXCEPTION"; break;
2120                case UT_DATA_ERROR:
2121                        s = "UT_DATA_ERROR"; break;
2122                case UT_DATA_PROTECTION:
2123                        s = "UT_DATA_PROTECTION"; break;
2124                case UT_MEM_ADDRESS_NOT_ALIGNED:
2125                        s = "UT_MEM_ADDRESS_NOT_ALIGNED"; break;
2126                case UT_PRIVILEGED_ACTION:
2127                        s = "UT_PRIVILEGED_ACTION"; break;
2128                case UT_ASYNC_DATA_ERROR:
2129                        s = "UT_ASYNC_DATA_ERROR"; break;
2130                case UT_TRAP_INSTRUCTION_16:
2131                        s = "UT_TRAP_INSTRUCTION_16"; break;
2132                case UT_TRAP_INSTRUCTION_17:
2133                        s = "UT_TRAP_INSTRUCTION_17"; break;
2134                case UT_TRAP_INSTRUCTION_18:
2135                        s = "UT_TRAP_INSTRUCTION_18"; break;
2136                case UT_TRAP_INSTRUCTION_19:
2137                        s = "UT_TRAP_INSTRUCTION_19"; break;
2138                case UT_TRAP_INSTRUCTION_20:
2139                        s = "UT_TRAP_INSTRUCTION_20"; break;
2140                case UT_TRAP_INSTRUCTION_21:
2141                        s = "UT_TRAP_INSTRUCTION_21"; break;
2142                case UT_TRAP_INSTRUCTION_22:
2143                        s = "UT_TRAP_INSTRUCTION_22"; break;
2144                case UT_TRAP_INSTRUCTION_23:
2145                        s = "UT_TRAP_INSTRUCTION_23"; break;
2146                case UT_TRAP_INSTRUCTION_24:
2147                        s = "UT_TRAP_INSTRUCTION_24"; break;
2148                case UT_TRAP_INSTRUCTION_25:
2149                        s = "UT_TRAP_INSTRUCTION_25"; break;
2150                case UT_TRAP_INSTRUCTION_26:
2151                        s = "UT_TRAP_INSTRUCTION_26"; break;
2152                case UT_TRAP_INSTRUCTION_27:
2153                        s = "UT_TRAP_INSTRUCTION_27"; break;
2154                case UT_TRAP_INSTRUCTION_28:
2155                        s = "UT_TRAP_INSTRUCTION_28"; break;
2156                case UT_TRAP_INSTRUCTION_29:
2157                        s = "UT_TRAP_INSTRUCTION_29"; break;
2158                case UT_TRAP_INSTRUCTION_30:
2159                        s = "UT_TRAP_INSTRUCTION_30"; break;
2160                case UT_TRAP_INSTRUCTION_31:
2161                        s = "UT_TRAP_INSTRUCTION_31"; break;
2162                }
2163        }
2164 #endif /* __sparc */

2166        if (s == NULL)
2167                prt_dec(pri, 0, val);
2168        else
2169                outstring(pri, s);
2170 }
```



```
2173 /*
2174  * Print utrap handler
2175  */
2176 void
2177 prt_uth(private_t *pri, int raw, long val)
2178 {
2179        const char *s = NULL;

2181        if (!raw) {
2182                switch (val) {
2183                case (long)UTH_NOCHANGE:        s = "UTH_NOCHANGE"; break;
2184                }
2185        }

2187        if (s == NULL)
2188                prt_hex(pri, 0, val);
2189        else
2190                outstring(pri, s);
2191 }

2193 const char *
2194 access_flags(private_t *pri, long arg)
2195 {
2196 #define E_OK 010
2197        char *str = pri->code_buf;

2199        if (arg & ~(R_OK|W_OK|X_OK|E_OK))
2200                return (NULL);

2202        /* NB: F_OK == 0 */
2203        if (arg == F_OK)
2204                return ("F_OK");
2205        if (arg == E_OK)
2206                return ("F_OK|E_OK");

2208        *str = '\0';
2209        if (arg & R_OK)
2210                (void) strlcat(str, "|R_OK", sizeof (pri->code_buf));
2211        if (arg & W_OK)
2212                (void) strlcat(str, "|W_OK", sizeof (pri->code_buf));
2213        if (arg & X_OK)
2214                (void) strlcat(str, "|X_OK", sizeof (pri->code_buf));
2215        if (arg & E_OK)
2216                (void) strlcat(str, "|E_OK", sizeof (pri->code_buf));
2217        return ((const char *)(str + 1));
2218 #undef E_OK
2219 }

2221 /*
2222  * Print access() flags.
2223  */
2224 void
2225 prt_acc(private_t *pri, int raw, long val)
2226 {
2227        const char *s = raw? NULL : access_flags(pri, val);

2229        if (s == NULL)
2230                prt_dex(pri, 0, val);
2231        else
2232                outstring(pri, s);
2233 }

2235 /*
2236  * Print shutdown() "how" (2nd) argument
```

```
 */
void
prt_sht(private_t *pri, int raw, long val)
{
        if (raw) {
                prt_dex(pri, 0, val);
                return;
        }
        switch (val) {
        case SHUT_RD:   outstring(pri, "SHUT_RD");      break;
        case SHUT_WR:   outstring(pri, "SHUT_WR");      break;
        case SHUT_RDWR: outstring(pri, "SHUT_RDWR");    break;
        default:        prt_dec(pri, 0, val);           break;
        }
}

/*
 * Print fcntl() F_SETFL flags (3rd) argument or fdsync flag (2nd arg)
 */
static struct fcntl_flags {
        long            val;
        const char      *name;
} fcntl_flags[] = {
#define FC_FL(flag)     { (long)flag, "|" # flag }
        FC_FL(FREVOKED),
        FC_FL(FREAD),
        FC_FL(FWRITE),
        FC_FL(FNDELAY),
        FC_FL(FAPPEND),
        FC_FL(FSYNC),
        FC_FL(FDSYNC),
        FC_FL(FRSYNC),
        FC_FL(FOFFMAX),
        FC_FL(FNONBLOCK),
        FC_FL(FCREAT),
        FC_FL(FTRUNC),
        FC_FL(FEXCL),
        FC_FL(FNOCTTY),
        FC_FL(FXATTR),
        FC_FL(FASYNC),
        FC_FL(FNODSYNC)
#undef FC_FL
};

void
prt_ffg(private_t *pri, int raw, long val)
{
#define CBSIZE  sizeof (pri->code_buf)
        char *s = pri->code_buf;
        size_t used = 1;
        struct fcntl_flags *fp;

        if (raw) {
                (void) snprintf(s, CBSIZE, "0x%lx", val);
                outstring(pri, s);
                return;
        }
        if (val == 0) {
                outstring(pri, "(no flags)");
                return;
        }

        *s = '\0';
        for (fp = fcntl_flags;
            fp < &fcntl_flags[sizeof (fcntl_flags) / sizeof (*fp)]; fp++) {
                if (val & fp->val) {
```



```
                        used = strlcat(s, fp->name, CBSIZE);
                        val &= ~fp->val;
                }
        }

        if (val != 0 && used <= CBSIZE)
                used += snprintf(s + used, CBSIZE - used, "|0x%lx", val);

        if (used >= CBSIZE)
                (void) snprintf(s + 1, CBSIZE-1, "0x%lx", val);
        outstring(pri, s + 1);
#undef CBSIZE
}

void
prt_prs(private_t *pri, int raw, long val)
{
        static size_t setsize;
        priv_set_t *set = priv_allocset();

        if (setsize == 0) {
                const priv_impl_info_t *info = getprivimplinfo();
                if (info != NULL)
                        setsize = info->priv_setsize * sizeof (priv_chunk_t);
        }

        if (setsize != 0 && !raw && set != NULL &&
            Pread(Proc, set, setsize, val) == setsize) {
                int i;

                outstring(pri, "{");
                for (i = 0; i < setsize / sizeof (priv_chunk_t); i++) {
                        char buf[9];    /* 8 hex digits + '\0' */
                        (void) snprintf(buf, sizeof (buf), "%08x",
                            ((priv_chunk_t *)set)[i]);
                        outstring(pri, buf);
                }

                outstring(pri, "}");
        } else {
                prt_hex(pri, 0, val);
        }

        if (set != NULL)
                priv_freeset(set);
}

/*
 * Print privilege set operation.
 */
void
prt_pro(private_t *pri, int raw, long val)
{
        const char *s = NULL;

        if (!raw) {
                switch ((priv_op_t)val) {
                case PRIV_ON:           s = "PRIV_ON";          break;
                case PRIV_OFF:          s = "PRIV_OFF";         break;
                case PRIV_SET:          s = "PRIV_SET";         break;
                }
        }

        if (s == NULL)
                prt_dec(pri, 0, val);
        else
```

```
2369                outstring(pri, s);
2370 }

2372 /*
2373  * Print privilege set name
2374  */
2375 void
2376 prt_prn(private_t *pri, int raw, long val)
2377 {
2378         const char *s = NULL;

2380         if (!raw)
2381                 s = priv_getsetbynum((int)val);

2383         if (s == NULL)
2384                 prt_dec(pri, 0, val);
2385         else {
2386                 char *dup = strdup(s);
2387                 char *q;

2389                 /* Do the best we can in this case */
2390                 if (dup == NULL) {
2391                         outstring(pri, s);
2392                         return;
2393                 }

2395                 outstring(pri, "PRIV_");

2397                 q = dup;

2399                 while (*q != '\0') {
2400                         *q = toupper(*q);
2401                         q++;
2402                 }
2403                 outstring(pri, dup);
2404                 free(dup);
2405         }
2406 }

2408 /*
2409  * Print process flag names.
2410  */
2411 void
2412 prt_pfl(private_t *pri, int raw, long val)
2413 {
2414         const char *s = NULL;

2416         if (!raw) {
2417                 switch ((int)val) {
2418                 case PRIV_DEBUG:        s = "PRIV_DEBUG";       break;
2419                 case PRIV_AWARE:        s = "PRIV_AWARE";       break;
2420                 case PRIV_XPOLICY:      s = "PRIV_XPOLICY";     break;
2421                 case PRIV_AWARE_RESET:  s = "PRIV_AWARE_RESET"; break;
2422                 case PRIV_PFEXEC:       s = "PRIV_PFEXEC";      break;
2423                 case NET_MAC_AWARE:     s =  "NET_MAC_AWARE";   break;
2424                 case NET_MAC_AWARE_INHERIT:
2425                         s = "NET_MAC_AWARE_INHERIT";
2426                         break;
2427                 }
2428         }

2430         if (s == NULL)
2431                 prt_dec(pri, 0, val);
2432         else
2433                 outstring(pri, s);
2434 }
```



```
2436 /*
2437  * Print lgrp_affinity_{get,set}() arguments.
2438  */
2439 /*ARGSUSED*/
2440 void
2441 prt_laf(private_t *pri, int raw, long val)
2442 {
2443         lgrp_affinity_args_t    laff;

2445         if (Pread(Proc, &laff, sizeof (lgrp_affinity_args_t), val) !=
2446             sizeof (lgrp_affinity_args_t)) {
2447                 prt_hex(pri, 0, val);
2448                 return;
2449         }
2450         /*
2451          * arrange the arguments in the order that user calls with
2452          */
2453         prt_dec(pri, 0, laff.idtype);
2454         outstring(pri, ", ");
2455         prt_dec(pri, 0, laff.id);
2456         outstring(pri, ", ");
2457         prt_dec(pri, 0, laff.lgrp);
2458         outstring(pri, ", ");
2459         if (pri->sys_args[0] == LGRP_SYS_AFFINITY_SET)
2460                 prt_dec(pri, 0, laff.aff);
2461 }

2463 /*
2464  * Print a key_t as IPC_PRIVATE if it is 0.
2465  */
2466 void
2467 prt_key(private_t *pri, int raw, long val)
2468 {
2469         if (!raw && val == 0)
2470                 outstring(pri, "IPC_PRIVATE");
2471         else
2472                 prt_dec(pri, 0, val);
2473 }


2476 /*
2477  * Print zone_getattr() attribute types.
2478  */
2479 void
2480 prt_zga(private_t *pri, int raw, long val)
2481 {
2482         const char *s = NULL;

2484         if (!raw) {
2485                 switch ((int)val) {
2486                 case ZONE_ATTR_NAME:    s = "ZONE_ATTR_NAME";   break;
2487                 case ZONE_ATTR_ROOT:    s = "ZONE_ATTR_ROOT";   break;
2488                 case ZONE_ATTR_STATUS:  s = "ZONE_ATTR_STATUS"; break;
2489                 case ZONE_ATTR_PRIVSET: s = "ZONE_ATTR_PRIVSET"; break;
2490                 case ZONE_ATTR_UNIQID:  s = "ZONE_ATTR_UNIQID"; break;
2491                 case ZONE_ATTR_POOLID:  s = "ZONE_ATTR_POOLID"; break;
2492                 case ZONE_ATTR_INITPID: s = "ZONE_ATTR_INITPID"; break;
2493                 case ZONE_ATTR_SLBL:    s = "ZONE_ATTR_SLBL"; break;
2494                 case ZONE_ATTR_INITNAME:        s = "ZONE_ATTR_INITNAME"; break;
2495                 case ZONE_ATTR_BOOTARGS:        s = "ZONE_ATTR_BOOTARGS"; break;
2496                 case ZONE_ATTR_BRAND:   s = "ZONE_ATTR_BRAND"; break;
2497                 case ZONE_ATTR_FLAGS:   s = "ZONE_ATTR_FLAGS"; break;
2498                 case ZONE_ATTR_PHYS_MCAP: s = "ZONE_ATTR_PHYS_MCAP"; break;
2499                 }
2500         }
```

```
2502         if (s == NULL)
2503                 prt_dec(pri, 0, val);
2504         else
2505                 outstring(pri, s);
2506 }

2508 /*
2509  * Print a file descriptor as AT_FDCWD if necessary
2510  */
2511 void
2512 prt_atc(private_t *pri, int raw, long val)
2513 {
2514         if ((int)val == AT_FDCWD) {
2515                 if (raw)
2516                         prt_hex(pri, 0, (uint_t)AT_FDCWD);
2517                 else
2518                         outstring(pri, "AT_FDCWD");
2519         } else {
2520                 prt_dec(pri, 0, val);
2521         }
2522 }

2524 /*
2525  * Print Trusted Networking database operation codes (labelsys; tn*)
2526  */
2527 static void
2528 prt_tnd(private_t *pri, int raw, long val)
2529 {
2530         const char *s = NULL;

2532         if (!raw) {
2533                 switch ((tsol_dbops_t)val) {
2534                 case TNDB_NOOP:         s = "TNDB_NOOP";        break;
2535                 case TNDB_LOAD:         s = "TNDB_LOAD";        break;
2536                 case TNDB_DELETE:       s = "TNDB_DELETE";      break;
2537                 case TNDB_FLUSH:        s = "TNDB_FLUSH";       break;
2538                 case TNDB_GET:          s = "TNDB_GET";         break;
2539                 }
2540         }

2542         if (s == NULL)
2543                 prt_dec(pri, 0, val);
2544         else
2545                 outstring(pri, s);
2546 }

2548 /*
2549  * Print LIO_XX flags
2550  */
2551 void
2552 prt_lio(private_t *pri, int raw, long val)
2553 {
2554         if (raw)
2555                 prt_dec(pri, 0, val);
2556         else if (val == LIO_WAIT)
2557                 outstring(pri, "LIO_WAIT");
2558         else if (val == LIO_NOWAIT)
2559                 outstring(pri, "LIO_NOWAIT");
2560         else
2561                 prt_dec(pri, 0, val);
2562 }

2564 const char *
2565 door_flags(private_t *pri, long val)
2566 {
```



```
2567         door_attr_t attr = (door_attr_t)val;
2568         char *str = pri->code_buf;

2570         *str = '\0';
2571 #define PROCESS_FLAG(flg)                                               \
2572         if (attr & flg) {                                               \
2573                 (void) strlcat(str, "|" #flg, sizeof (pri->code_buf));  \
2574                 attr &= ~flg;                                           \
2575         }

2577         PROCESS_FLAG(DOOR_UNREF);
2578         PROCESS_FLAG(DOOR_UNREF_MULTI);
2579         PROCESS_FLAG(DOOR_PRIVATE);
2580         PROCESS_FLAG(DOOR_REFUSE_DESC);
2581         PROCESS_FLAG(DOOR_NO_CANCEL);
2582         PROCESS_FLAG(DOOR_LOCAL);
2583         PROCESS_FLAG(DOOR_REVOKED);
2584         PROCESS_FLAG(DOOR_IS_UNREF);
2585 #undef PROCESS_FLAG

2587         if (attr != 0 || *str == '\0') {
2588                 size_t len = strlen(str);
2589                 (void) snprintf(str + len, sizeof (pri->code_buf) - len,
2590                     "|0x%X", attr);
2591         }

2593         return (str + 1);
2594 }

2596 /*
2597  * Print door_create() flags
2598  */
2599 void
2600 prt_dfl(private_t *pri, int raw, long val)
2601 {
2602         if (raw)
2603                 prt_hex(pri, 0, val);
2604         else
2605                 outstring(pri, door_flags(pri, val));
2606 }

2608 /*
2609  * Print door_*param() param argument
2610  */
2611 void
2612 prt_dpm(private_t *pri, int raw, long val)
2613 {
2614         if (raw)
2615                 prt_hex(pri, 0, val);
2616         else if (val == DOOR_PARAM_DESC_MAX)
2617                 outstring(pri, "DOOR_PARAM_DESC_MAX");
2618         else if (val == DOOR_PARAM_DATA_MIN)
2619                 outstring(pri, "DOOR_PARAM_DATA_MIN");
2620         else if (val == DOOR_PARAM_DATA_MAX)
2621                 outstring(pri, "DOOR_PARAM_DATA_MAX");
2622         else
2623                 prt_hex(pri, 0, val);
2624 }

2626 /*
2627  * Print rctlsys subcodes
2628  */
2629 void
2630 prt_rsc(private_t *pri, int raw, long val)      /* print utssys code */
2631 {
2632         const char *s = raw? NULL : rctlsyscode(val);
```

```
2634         if (s == NULL)
2635                 prt_dec(pri, 0, val);
2636         else
2637                 outstring(pri, s);
2638 }

2640 /*
2641  * Print getrctl flags
2642  */
2643 void
2644 prt_rgf(private_t *pri, int raw, long val)
2645 {
2646         long action = val & (~RCTLSYS_ACTION_MASK);

2648         if (raw)
2649                 prt_hex(pri, 0, val);
2650         else if (action == RCTL_FIRST)
2651                 outstring(pri, "RCTL_FIRST");
2652         else if (action == RCTL_NEXT)
2653                 outstring(pri, "RCTL_NEXT");
2654         else if (action == RCTL_USAGE)
2655                 outstring(pri, "RCTL_USAGE");
2656         else
2657                 prt_hex(pri, 0, val);
2658 }

2660 /*
2661  * Print setrctl flags
2662  */
2663 void
2664 prt_rsf(private_t *pri, int raw, long val)
2665 {
2666         long action = val & (~RCTLSYS_ACTION_MASK);
2667         long pval = val & RCTL_LOCAL_ACTION_MASK;
2668         char *s = pri->code_buf;

2670         if (raw) {
2671                 prt_hex(pri, 0, val);
2672                 return;
2673         } else if (action == RCTL_INSERT)
2674                 (void) strcpy(s, "RCTL_INSERT");
2675         else if (action == RCTL_DELETE)
2676                 (void) strcpy(s, "RCTL_DELETE");
2677         else if (action == RCTL_REPLACE)
2678                 (void) strcpy(s, "RCTL_REPLACE");
2679         else {
2680                 prt_hex(pri, 0, val);
2681                 return;
2682         }

2684         if (pval & RCTL_USE_RECIPIENT_PID) {
2685                 pval ^= RCTL_USE_RECIPIENT_PID;
2686                 (void) strlcat(s, "|RCTL_USE_RECIPIENT_PID",
2687                     sizeof (pri->code_buf));
2688         }

2690         if ((pval & RCTLSYS_ACTION_MASK) != 0)
2691                 prt_hex(pri, 0, val);
2692         else if (*s != '\0')
2693                 outstring(pri, s);
2694         else
2695                 prt_hex(pri, 0, val);
2696 }

2698 /*
```



```
2699  * Print rctlctl flags
2700  */
2701 void
2702 prt_rcf(private_t *pri, int raw, long val)
2703 {
2704         long action = val & (~RCTLSYS_ACTION_MASK);

2706         if (raw)
2707                 prt_hex(pri, 0, val);
2708         else if (action == RCTLCTL_GET)
2709                 outstring(pri, "RCTLCTL_GET");
2710         else if (action == RCTLCTL_SET)
2711                 outstring(pri, "RCTLCTL_SET");
2712         else
2713                 prt_hex(pri, 0, val);
2714 }

2716 /*
2717  * Print setprojrctl flags
2718  */
2719 void
2720 prt_spf(private_t *pri, int raw, long val)
2721 {
2722         long action = val & TASK_PROJ_MASK;

2724         if (!raw && (action == TASK_PROJ_PURGE))
2725                 outstring(pri, "TASK_PROJ_PURGE");
2726         else
2727                 prt_hex(pri, 0, val);
2728 }

2730 /*
2731  * Print forkx() flags
2732  */
2733 void
2734 prt_fxf(private_t *pri, int raw, long val)
2735 {
2736         char *str;

2738         if (val == 0)
2739                 outstring(pri, "0");
2740         else if (raw || (val & ~(FORK_NOSIGCHLD | FORK_WAITPID)))
2741                 prt_hhx(pri, 0, val);
2742         else {
2743                 str = pri->code_buf;
2744                 *str = '\0';
2745                 if (val & FORK_NOSIGCHLD)
2746                         (void) strlcat(str, "|FORK_NOSIGCHLD",
2747                             sizeof (pri->code_buf));
2748                 if (val & FORK_WAITPID)
2749                         (void) strlcat(str, "|FORK_WAITPID",
2750                             sizeof (pri->code_buf));
2751                 outstring(pri, str + 1);
2752         }
2753 }

2755 /*
2756  * Print faccessat() flag
2757  */
2758 void
2759 prt_fat(private_t *pri, int raw, long val)
2760 {
2761         if (val == 0)
2762                 outstring(pri, "0");
2763         else if (!raw && val == AT_EACCESS)
2764                 outstring(pri, "AT_EACCESS");
```

```
2765         else
2766                 prt_hex(pri, 0, val);
2767 }

2769 /*
2770  * Print unlinkat() flag
2771  */
2772 void
2773 prt_uat(private_t *pri, int raw, long val)
2774 {
2775         if (val == 0)
2776                 outstring(pri, "0");
2777         else if (!raw && val == AT_REMOVEDIR)
2778                 outstring(pri, "AT_REMOVEDIR");
2779         else
2780                 prt_hex(pri, 0, val);
2781 }

2783 /*
2784  * Print AT_SYMLINK_NOFOLLOW / AT_SYMLINK_FOLLOW flag
2785  */
2786 void
2787 prt_snf(private_t *pri, int raw, long val)
2788 {
2789         if (val == 0)
2790                 outstring(pri, "0");
2791         else if (!raw && val == AT_SYMLINK_NOFOLLOW)
2792                 outstring(pri, "AT_SYMLINK_NOFOLLOW");
2793         else if (!raw && val == AT_SYMLINK_FOLLOW)
2794                 outstring(pri, "AT_SYMLINK_FOLLOW");
2795         else
2796                 prt_hex(pri, 0, val);
2797 }

2799 void
2800 prt_grf(private_t *pri, int raw, long val)
2801 {
2802         int first = 1;

2804         if (raw != 0 || val == 0 ||
2805             (val & ~(GRND_NONBLOCK | GRND_RANDOM)) != 0) {
2806                 outstring(pri, "0");
2807                 return;
2808         }

2810         if (val & GRND_NONBLOCK) {
2811                 outstring(pri, "|GRND_NONBLOCK" + first);
2812                 first = 0;
2813         }
2814         if (val & GRND_RANDOM) {
2815                 outstring(pri, "|GRND_RANDOM" + first);
2816                 first = 0;
2817         }
2818 }

2820 /*
2821  * Array of pointers to print functions, one for each format.
2822  */
2823 void (* const Print[])() = {
2824         prt_nov,        /* NOV -- no value */
2825         prt_dec,        /* DEC -- print value in decimal */
2826         prt_oct,        /* OCT -- print value in octal */
2827         prt_hex,        /* HEX -- print value in hexadecimal */
2828         prt_dex,        /* DEX -- print value in hexadecimal if big enough */
2829         prt_stg,        /* STG -- print value as string */
2830         prt_ioc,        /* IOC -- print ioctl code */
```



```
2831         prt_fcn,        /* FCN -- print fcntl code */
2832         prt_s86,        /* S86 -- print sysi86 code */
2833         prt_uts,        /* UTS -- print utssys code */
2834         prt_opn,        /* OPN -- print open code */
2835         prt_sig,        /* SIG -- print signal name plus flags */
2836         prt_uat,        /* UAT -- print unlinkat() flag */
2837         prt_msc,        /* MSC -- print msgsys command */
2838         prt_msf,        /* MSF -- print msgsys flags */
2839         prt_smc,        /* SMC -- print semsys command */
2840         prt_sef,        /* SEF -- print semsys flags */
2841         prt_shc,        /* SHC -- print shmsys command */
2842         prt_shf,        /* SHF -- print shmsys flags */
2843         prt_fat,        /* FAT -- print faccessat( flag */
2844         prt_sfs,        /* SFS -- print sysfs code */
2845         prt_rst,        /* RST -- print string returned by syscall */
2846         prt_smf,        /* SMF -- print streams message flags */
2847         prt_ioa,        /* IOA -- print ioctl argument */
2848         prt_pip,        /* PIP -- print pipe flags */
2849         prt_mtf,        /* MTF -- print mount flags */
2850         prt_mft,        /* MFT -- print mount file system type */
2851         prt_iob,        /* IOB -- print contents of I/O buffer */
2852         prt_hhx,        /* HHX -- print value in hexadecimal (half size) */
2853         prt_wop,        /* WOP -- print waitsys() options */
2854         prt_spm,        /* SPM -- print sigprocmask argument */
2855         prt_rlk,        /* RLK -- print readlink buffer */
2856         prt_mpr,        /* MPR -- print mmap()/mprotect() flags */
2857         prt_mty,        /* MTY -- print mmap() mapping type flags */
2858         prt_mcf,        /* MCF -- print memcntl() function */
2859         prt_mc4,        /* MC4 -- print memcntl() (fourth) argument */
2860         prt_mc5,        /* MC5 -- print memcntl() (fifth) argument */
2861         prt_mad,        /* MAD -- print madvise() argument */
2862         prt_ulm,        /* ULM -- print ulimit() argument */
2863         prt_rlm,        /* RLM -- print get/setrlimit() argument */
2864         prt_cnf,        /* CNF -- print sysconfig() argument */
2865         prt_inf,        /* INF -- print sysinfo() argument */
2866         prt_ptc,        /* PTC -- print pathconf/fpathconf() argument */
2867         prt_fui,        /* FUI -- print fusers() input argument */
2868         prt_idt,        /* IDT -- print idtype_t, waitid() argument */
2869         prt_lwf,        /* LWF -- print lwp_create() flags */
2870         prt_itm,        /* ITM -- print [get|set]itimer() arg */
2871         prt_llo,        /* LLO -- print long long offset arg */
2872         prt_mod,        /* MOD -- print modctl() subcode */
2873         prt_whn,        /* WHN -- print lseek() whence arguiment */
2874         prt_acl,        /* ACL -- print acl() code */
2875         prt_aio,        /* AIO -- print kaio() code */
2876         prt_aud,        /* AUD -- print auditsys() code */
2877         prt_uns,        /* DEC -- print value in unsigned decimal */
2878         prt_clc,        /* CLC -- print cladm command argument */
2879         prt_clf,        /* CLF -- print cladm flag argument */
2880         prt_cor,        /* COR -- print corectl() subcode */
2881         prt_cco,        /* CCO -- print corectl() options */
2882         prt_ccc,        /* CCC -- print corectl() content */
2883         prt_rcc,        /* RCC -- print corectl() returned content */
2884         prt_cpc,        /* CPC -- print cpc() subcode */
2885         prt_sqc,        /* SQC -- print sigqueue() si_code argument */
2886         prt_pc4,        /* PC4 -- print priocntlsys() (fourth) argument */
2887         prt_pc5,        /* PC5 -- print priocntlsys() (key, value) pairs */
2888         prt_pst,        /* PST -- print processor set id */
2889         prt_mif,        /* MIF -- print meminfo() arguments */
2890         prt_pfm,        /* PFM -- print so_socket() proto-family (1st) arg */
2891         prt_skt,        /* SKT -- print so_socket() socket-type (2nd) arg */
2892         prt_skp,        /* SKP -- print so_socket() protocol (3rd) arg */
2893         prt_skv,        /* SKV -- print socket version arg */
2894         prt_sol,        /* SOL -- print [sg]etsockopt() level (2nd) arg */
2895         prt_son,        /* SON -- print [sg]etsockopt() opt-name (3rd) arg */
2896         prt_utt,        /* UTT -- print utrap type */
```

```
2897         prt_uth,        /* UTH -- print utrap handler */
2898         prt_acc,        /* ACC -- print access() flags */
2899         prt_sht,        /* SHT -- print shutdown() how (2nd) argument */
2900         prt_ffg,        /* FFG -- print fcntl() flags (3rd) argument */
2901         prt_prs,        /* PRS -- print privilege set */
2902         prt_pro,        /* PRO -- print privilege set operation */
2903         prt_prn,        /* PRN -- print privilege set name */
2904         prt_pfl,        /* PFL -- print privilege/process flag name */
2905         prt_laf,        /* LAF -- print lgrp_affinity arguments */
2906         prt_key,        /* KEY -- print key_t 0 as IPC_PRIVATE */
2907         prt_zga,        /* ZGA -- print zone_getattr attribute types */
2908         prt_atc,        /* ATC -- print AT_FDCWD or file descriptor */
2909         prt_lio,        /* LIO -- print LIO_XX flags */
2910         prt_dfl,        /* DFL -- print door_create() flags */
2911         prt_dpm,        /* DPM -- print DOOR_PARAM_XX flags */
2912         prt_tnd,        /* TND -- print trusted network data base opcode */
2913         prt_rsc,        /* RSC -- print rctlsys() subcodes */
2914         prt_rgf,        /* RGF -- print getrctl() flags */
2915         prt_rsf,        /* RSF -- print setrctl() flags */
2916         prt_rcf,        /* RCF -- print rctlsys_ctl() flags */
2917         prt_fxf,        /* FXF -- print forkx() flags */
2918         prt_spf,        /* SPF -- print rctlsys_projset() flags */
2919         prt_un1,        /* UN1 -- as prt_uns except for -1 */
2920         prt_mob,        /* MOB -- print mmapobj() flags */
2921         prt_snf,        /* SNF -- print AT_SYMLINK_[NO]FOLLOW flag */
2922         prt_skc,        /* SKC -- print sockconfig() subcode */
2923         prt_acf,        /* ACF -- print accept4 flags */
2924         prt_pfd,        /* PFD -- print pipe fds */
2925         prt_grf,        /* GRF -- print getrandom flags */
2926         prt_psfcmd,     /* PSFCMD -- print psecflags(2) command */
2927         prt_psflags,    /* PSFLG -- print psecflags(2) flags */
2928 #endif /* ! codereview */
2929         prt_dec,        /* HID -- hidden argument, make this the last one */
2930 };
```

```
**********************************************************
    6433 Wed May 27 19:49:08 2015
new/usr/src/cmd/truss/print.h
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright (c) 1989, 2010, Oracle and/or its affiliates. All rights reserved.
 * Copyright (c) 2015, Joyent, Inc.
 */

/*      Copyright (c) 1984, 1986, 1987, 1988, 1989 AT&T */
/*        All Rights Reserved   */

/* Copyright (c) 2013, OmniTI Computer Consulting, Inc. All right reserved. */

#ifndef _TRUSS_PRINT_H
#define _TRUSS_PRINT_H

#ifdef  __cplusplus
extern "C" {
#endif

/*
 * Argument & return value print codes.
 */
#define NOV     0               /* no value */
#define DEC     1               /* print value in decimal */
#define OCT     2               /* print value in octal */
#define HEX     3               /* print value in hexadecimal */
#define DEX     4               /* print value in hexadecimal if big enough */
#define STG     5               /* print value as string */
#define IOC     6               /* print ioctl code */
#define FCN     7               /* print fcntl code */
#define S86     8               /* print sysi86 code */
#define UTS     9               /* print utssys code */
#define OPN     10              /* print open code */
#define SIG     11              /* print signal name plus flags */
#define UAT     12              /* print unlinkat() flag */
#define MSC     13              /* print msgsys command */
#define MSF     14              /* print msgsys flags */
#define SMC     15              /* print semsys command */
#define SEF     16              /* print semsys flags */
```



```
#define SHC     17              /* print shmsys command */
#define SHF     18              /* print shmsys flags */
#define FAT     19              /* print faccessat() flag */
#define SFS     20              /* print sysfs code */
#define RST     21              /* print string returned by sys call */
#define SMF     22              /* print streams message flags */
#define IOA     23              /* print ioctl argument */
#define PIP     24              /* print pipe flags */
#define MTF     25              /* print mount flags */
#define MFT     26              /* print mount file system type */
#define IOB     27              /* print contents of I/O buffer */
#define HHX     28              /* print value in hexadecimal (half size) */
#define WOP     29              /* print waitsys() options */
#define SPM     30              /* print sigprocmask argument */
#define RLK     31              /* print readlink buffer */
#define MPR     32              /* print mmap()/mprotect() flags */
#define MTY     33              /* print mmap() mapping type flags */
#define MCF     34              /* print memcntl() function */
#define MC4     35              /* print memcntl() (fourth) argument */
#define MC5     36              /* print memcntl() (fifth) argument */
#define MAD     37              /* print madvise() argument */
#define ULM     38              /* print ulimit() argument */
#define RLM     39              /* print get/setrlimit() argument */
#define CNF     40              /* print sysconfig() argument */
#define INF     41              /* print sysinfo() argument */
#define PTC     42              /* print pathconf/fpathconf() argument */
#define FUI     43              /* print fusers() input argument */
#define IDT     44              /* print idtype_t, waitid() argument */
#define LWF     45              /* print lwp_create() flags */
#define ITM     46              /* print [get|set]itimer() arg */
#define LLO     47              /* print long long offset */
#define MOD     48              /* print modctl() code */
#define WHN     49              /* print lseek() whence argument */
#define ACL     50              /* print acl() code */
#define AIO     51              /* print kaio() code */
#define AUD     52              /* print auditsys() code */
#define UNS     53              /* print value in unsigned decimal */
#define CLC     54              /* print cladm() command argument */
#define CLF     55              /* print cladm() flag argument */
#define COR     56              /* print corectl() subcode */
#define CCO     57              /* print corectl() options */
#define CCC     58              /* print corectl() content */
#define RCC     59              /* print corectl() content */
#define CPC     60              /* print cpc() subcode */
#define SQC     61              /* print sigqueue() si_code argument */
#define PC4     62              /* print priocntlsys() (fourth) argument */
#define PC5     63              /* print priocntlsys() (key-value) pairs */
#define PST     64              /* print processor set id */
#define MIF     65              /* print meminfo() argument */
#define PFM     66              /* print so_socket() proto-family (1st) arg */
#define SKT     67              /* print so_socket() socket type (2nd) arg */
#define SKP     68              /* print so_socket() protocol (3rd) arg */
#define SKV     69              /* print so_socket() version (5th) arg */
#define SOL     70              /* print [sg]etsockopt() level (2nd) arg */
#define SON     71              /* print [sg]etsockopt() name (3rd) arg */
#define UTT     72              /* print utrap type */
#define UTH     73              /* print utrap handler */
#define ACC     74              /* print access flags */
#define SHT     75              /* print shutdown() "how" (2nd) arg */
#define FFG     76              /* print fcntl() flags (3rd) arg */
#define PRS     77              /* privilege set */
#define PRO     78              /* privilege set operation */
#define PRN     79              /* privilege set name */
#define PFL     80              /* privilege/process flag name */
#define LAF     81              /* print lgrp_affinity arguments */
#define KEY     82              /* print key_t 0 as IPC_PRIVATE */
```

```
125 #define ZGA     83              /* print zone_getattr attribute types */
126 #define ATC     84              /* print AT_FDCWD or file descriptor */
127 #define LIO     85              /* print LIO_XX flags */
128 #define DFL     86              /* print door_create() flags */
129 #define DPM     87              /* print DOOR_PARAM_XX flags */
130 #define TND     88              /* print trusted network data base opcode */
131 #define RSC     89              /* print rctlsys subcode */
132 #define RGF     90              /* print rctlsys_get flags */
133 #define RSF     91              /* print rctlsys_set flags */
134 #define RCF     92              /* print rctlsys_ctl flags */
135 #define FXF     93              /* print forkx flags */
136 #define SPF     94              /* print rctlsys_projset flags */
137 #define UN1     95              /* unsigned except for -1 */
138 #define MOB     96              /* print mmapobj() flags */
139 #define SNF     97              /* print AT_SYMLINK_[NO]FOLLOW flag */
140 #define SKC     98              /* print sockconfig subcode */
141 #define ACF     99              /* accept4 flags */
142 #define PFD     100             /* pipe fds[2] */
143 #define GRF     101             /* getrandom flags */
144 #define PSFCMD  102             /* psecflags(2) command */
145 #define PSFLG   103             /* psecflags(2) flags */
146 #define HID     104             /* hidden argument, don't print */
144 #define HID     102             /* hidden argument, don't print */
147                                 /* make sure HID is always the last member */

149 /*
150  * Print routines, indexed by print codes.
151  */
152 extern void (* const Print[])();

154 #ifdef  __cplusplus
155 }
```

*unchanged_portion_omitted_*

```
**********************************************************
   58034 Wed May 27 19:49:08 2015
new/usr/src/cmd/truss/systable.c
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
______unchanged_portion_omitted_


 221 const struct systable systable[] = {
 222 { NULL,         8, HEX, HEX, HEX, HEX, HEX, HEX, HEX, HEX, HEX, HEX},
 223 {"_exit",       1, DEC, NOV, DEC},                                  /*   1 */
 224 {"psecflags",   3, DEC, NOV, HEX, PSFCMD, PSFLG},                   /*   2 */
 224 { NULL,         8, HEX, HEX, HEX, HEX, HEX, HEX, HEX, HEX, HEX, HEX},
 225 {"read",        3, DEC, NOV, DEC, IOB, UNS},                        /*   3 */
 226 {"write",       3, DEC, NOV, DEC, IOB, UNS},                        /*   4 */
 227 {"open",        3, DEC, NOV, STG, OPN, OCT},                        /*   5 */
 228 {"close",       1, DEC, NOV, DEC},                                  /*   6 */
 229 {"linkat",      5, DEC, NOV, ATC, STG, ATC, STG, SNF},              /*   7 */
 230 { NULL,         8, HEX, HEX, HEX, HEX, HEX, HEX, HEX, HEX, HEX, HEX},
 231 {"link",        2, DEC, NOV, STG, STG},                             /*   9 */
 232 {"unlink",      1, DEC, NOV, STG},                                  /*  10 */
 233 {"symlinkat",   3, DEC, NOV, STG, ATC, STG},                        /*  11 */
 234 {"chdir",       1, DEC, NOV, STG},                                  /*  12 */
 235 {"time",        0, DEC, NOV},                                       /*  13 */
 236 {"mknod",       3, DEC, NOV, STG, OCT, HEX},                        /*  14 */
 237 {"chmod",       2, DEC, NOV, STG, OCT},                             /*  15 */
 238 {"chown",       3, DEC, NOV, STG, DEC, DEC},                        /*  16 */
 239 {"brk",         1, DEC, NOV, HEX},                                  /*  17 */
 240 {"stat",        2, DEC, NOV, STG, HEX},                             /*  18 */
 241 {"lseek",       3, DEC, NOV, DEC, DEX, WHN},                        /*  19 */
 242 {"getpid",      0, DEC, DEC},                                       /*  20 */
 243 {"mount",       8, DEC, NOV, STG, STG, MTF, MFT, HEX, DEC, HEX, DEC}, /* 21 */
 244 {"readlinkat",  4, DEC, NOV, ATC, STG, RLK, UNS},                   /*  22 */
 245 {"setuid",      1, DEC, NOV, UNS},                                  /*  23 */
 246 {"getuid",      0, UNS, UNS},                                       /*  24 */
 247 {"stime",       1, DEC, NOV, DEC},                                  /*  25 */
 248 {"pcsample",    2, DEC, NOV, HEX, DEC},                             /*  26 */
 249 {"alarm",       1, DEC, NOV, UNS},                                  /*  27 */
 250 {"fstat",       2, DEC, NOV, DEC, HEX},                             /*  28 */
 251 {"pause",       0, DEC, NOV},                                       /*  29 */
 252 { NULL,         8, HEX, HEX, HEX, HEX, HEX, HEX, HEX, HEX, HEX, HEX},
 253 {"stty",        2, DEC, NOV, DEC, DEC},                             /*  31 */
 254 {"gtty",        2, DEC, NOV, DEC, DEC},                             /*  32 */
 255 {"access",      2, DEC, NOV, STG, ACC},                             /*  33 */
 256 {"nice",        1, DEC, NOV, DEC},                                  /*  34 */
 257 {"statfs",      4, DEC, NOV, STG, HEX, DEC, DEC},                   /*  35 */
 258 {"sync",        0, DEC, NOV},                                       /*  36 */
 259 {"kill",        2, DEC, NOV, DEC, SIG},                             /*  37 */
 260 {"fstatfs",     4, DEC, NOV, DEC, HEX, DEC, DEC},                   /*  38 */
 261 {"pgrpsys",     3, DEC, NOV, DEC, DEC, DEC},                        /*  39 */
 262 {"uucopystr",   3, DEC, NOV, STG, RST, UNS},                        /*  40 */
 263 { NULL,         8, HEX, HEX, HEX, HEX, HEX, HEX, HEX, HEX, HEX, HEX},
 264 {"pipe",        2, DEC, NOV, PFD, PIP},                             /*  42 */
 265 {"times",       1, DEC, NOV, HEX},                                  /*  43 */
 266 {"profil",      4, DEC, NOV, HEX, UNS, HEX, OCT},                   /*  44 */
 267 {"faccessat",   4, DEC, NOV, ATC, STG, ACC, FAT},                   /*  45 */
 268 {"setgid",      1, DEC, NOV, UNS},                                  /*  46 */
 269 {"getgid",      0, UNS, UNS},                                       /*  47 */
 270 {"mknodat",     4, DEC, NOV, ATC, STG, OCT, HEX},                   /*  48 */
 271 {"msgsys",      6, DEC, NOV, DEC, DEC, DEC, DEC, DEC, DEC},         /*  49 */
 272 {"sysi86",      4, HEX, NOV, S86, HEX, HEX, HEX, DEC, DEC},         /*  50 */
 273 {"acct",        1, DEC, NOV, STG},                                  /*  51 */
 274 {"shmsys",      4, DEC, NOV, DEC, HEX, HEX, HEX},                   /*  52 */
```



```
 275 {"semsys",      5, DEC, NOV, DEC, HEX, HEX, HEX, HEX},              /*  53 */
 276 {"ioctl",       3, DEC, NOV, DEC, IOC, IOA},                        /*  54 */
 277 {"uadmin",      3, DEC, NOV, DEC, DEC, DEC},                        /*  55 */
 278 {"fchownat",    5, DEC, NOV, ATC, STG, DEC, DEC, SNF},              /*  56 */
 279 {"utssys",      4, DEC, NOV, HEX, DEC, UTS, HEX},                   /*  57 */
 280 {"fdsync",      2, DEC, NOV, DEC, FFG},                             /*  58 */
 281 {"execve",      3, DEC, NOV, STG, HEX, HEX},                        /*  59 */
 282 {"umask",       1, OCT, NOV, OCT},                                  /*  60 */
 283 {"chroot",      1, DEC, NOV, STG},                                  /*  61 */
 284 {"fcntl",       3, DEC, NOV, DEC, FCN, HEX},                        /*  62 */
 285 {"ulimit",      2, DEX, NOV, ULM, DEC},                             /*  63 */
 286 {"renameat",    4, DEC, NOV, ATC, STG, ATC, STG},                   /*  64 */
 287 {"unlinkat",    3, DEC, NOV, ATC, STG, UAT},                        /*  65 */
 288 {"fstatat",     4, DEC, NOV, ATC, STG, HEX, SNF},                   /*  66 */
 289 {"fstatat64",   4, DEC, NOV, ATC, STG, HEX, SNF},                   /*  67 */
 290 {"openat",      4, DEC, NOV, ATC, STG, OPN, OCT},                   /*  68 */
 291 {"openat64",    4, DEC, NOV, ATC, STG, OPN, OCT},                   /*  69 */
 292 {"tasksys",     5, DEC, NOV, DEC, DEC, DEC, HEX, DEC},              /*  70 */
 293 {"acctctl",     3, DEC, NOV, HEX, HEX, UNS},                        /*  71 */
 294 {"exacctsys",   6, DEC, NOV, DEC, IDT, DEC, HEX, DEC, HEX},         /*  72 */
 295 {"getpagesizes", 2, DEC, NOV, HEX, DEC},                            /*  73 */
 296 {"rctlsys",     6, DEC, NOV, RSC, STG, HEX, HEX, DEC, DEC},         /*  74 */
 297 {"sidsys",      4, UNS, UNS, DEC, DEC, DEC, DEC},                   /*  75 */
 298 { NULL,         8, HEX, HEX, HEX, HEX, HEX, HEX, HEX, HEX, HEX, HEX},
 299 {"lwp_park",    3, DEC, NOV, DEC, HEX, DEC},                        /*  77 */
 300 {"sendfilev",   5, DEC, NOV, DEC, DEC, HEX, DEC, HEX},              /*  78 */
 301 {"rmdir",       1, DEC, NOV, STG},                                  /*  79 */
 302 {"mkdir",       2, DEC, NOV, STG, OCT},                             /*  80 */
 303 {"getdents",    3, DEC, NOV, DEC, HEX, UNS},                        /*  81 */
 304 {"privsys",     5, HEX, NOV, DEC, DEC, DEC, HEX, DEC},              /*  82 */
 305 {"ucredsys",    3, DEC, NOV, DEC, DEC, HEX},                        /*  83 */
 306 {"sysfs",       3, DEC, NOV, SFS, DEX, DEX},                        /*  84 */
 307 {"getmsg",      4, DEC, NOV, DEC, HEX, HEX, HEX},                   /*  85 */
 308 {"putmsg",      4, DEC, NOV, DEC, HEX, HEX, SMF},                   /*  86 */
 309 { NULL,         8, HEX, HEX, HEX, HEX, HEX, HEX, HEX, HEX, HEX, HEX},
 310 {"lstat",       2, DEC, NOV, STG, HEX},                             /*  88 */
 311 {"symlink",     2, DEC, NOV, STG, STG},                             /*  89 */
 312 {"readlink",    3, DEC, NOV, STG, RLK, UNS},                        /*  90 */
 313 {"setgroups",   2, DEC, NOV, DEC, HEX},                             /*  91 */
 314 {"getgroups",   2, DEC, NOV, DEC, HEX},                             /*  92 */
 315 {"fchmod",      2, DEC, NOV, DEC, OCT},                             /*  93 */
 316 {"fchown",      3, DEC, NOV, DEC, DEC, DEC},                        /*  94 */
 317 {"sigprocmask", 3, DEC, NOV, SPM, HEX, HEX},                        /*  95 */
 318 {"sigsuspend",  1, DEC, NOV, HEX},                                  /*  96 */
 319 {"sigaltstack", 2, DEC, NOV, HEX, HEX},                             /*  97 */
 320 {"sigaction",   3, DEC, NOV, SIG, HEX, HEX},                        /*  98 */
 321 {"sigpendsys",  2, DEC, NOV, DEC, HEX},                             /*  99 */
 322 {"context",     2, DEC, NOV, DEC, HEX},                             /* 100 */
 323 {"fchmodat",    4, DEC, NOV, ATC, STG, OCT, SNF},                   /* 101 */
 324 {"mkdirat",     3, DEC, NOV, ATC, STG, OCT},                        /* 102 */
 325 {"statvfs",     2, DEC, NOV, STG, HEX},                             /* 103 */
 326 {"fstatvfs",    2, DEC, NOV, DEC, HEX},                             /* 104 */
 327 {"getloadavg",  2, DEC, NOV, HEX, DEC},                             /* 105 */
 328 {"nfssys",      2, DEC, NOV, DEC, HEX},                             /* 106 */
 329 {"waitid",      4, DEC, NOV, IDT, DEC, HEX, WOP},                   /* 107 */
 330 {"sigsendsys",  2, DEC, NOV, HEX, SIG},                             /* 108 */
 331 {"hrtsys",      5, DEC, NOV, DEC, HEX, HEX, HEX, HEX},              /* 109 */
 332 {"utimesys",    5, DEC, NOV, DEC, HEX, HEX, HEX, HEX},              /* 110 */
 333 {"sigresend",   3, DEC, NOV, SIG, HEX, HEX},                        /* 111 */
 334 {"priocntlsys", 5, DEC, NOV, DEC, HEX, DEC, PC4, PC5},              /* 112 */
 335 {"pathconf",    2, DEC, NOV, STG, PTC},                             /* 113 */
 336 {"mincore",     3, DEC, NOV, HEX, UNS, HEX},                        /* 114 */
 337 {"mmap",        6, HEX, NOV, HEX, UNS, MPR, MTY, DEC, DEC},         /* 115 */
 338 {"mprotect",    3, DEC, NOV, HEX, UNS, MPR},                        /* 116 */
 339 {"munmap",      2, DEC, NOV, HEX, UNS},                             /* 117 */
 340 {"fpathconf",   2, DEC, NOV, DEC, PTC},                             /* 118 */
```

```
 341 {"vfork",       0, DEC, NOV},                                   /* 119 */
 342 {"fchdir",      1, DEC, NOV, DEC},                              /* 120 */
 343 {"readv",       3, DEC, NOV, DEC, HEX, DEC},                    /* 121 */
 344 {"writev",      3, DEC, NOV, DEC, HEX, DEC},                    /* 122 */
 345 {"preadv",      4, DEC, NOV, DEC, HEX, DEC, DEC},               /* 123 */
 346 {"pwritev",     4, DEC, NOV, DEC, HEX, DEC, DEC},               /* 124 */
 347 { NULL,         8, HEX, HEX, HEX, HEX, HEX, HEX, HEX, HEX, HEX},
 348 {"getrandom",   3, DEC, NOV, IOB, UNS, GRF},                    /* 126 */
 349 {"mmapobj",     5, DEC, NOV, DEC, MOB, HEX, HEX, HEX},          /* 127 */
 350 {"setrlimit",   2, DEC, NOV, RLM, HEX},                         /* 128 */
 351 {"getrlimit",   2, DEC, NOV, RLM, HEX},                         /* 129 */
 352 {"lchown",      3, DEC, NOV, STG, DEC, DEC},                    /* 130 */
 353 {"memcntl",     6, DEC, NOV, HEX, UNS, MCF, MC4, MC5, DEC},     /* 131 */
 354 {"getpmsg",     5, DEC, NOV, DEC, HEX, HEX, HEX, HEX},          /* 132 */
 355 {"putpmsg",     5, DEC, NOV, DEC, HEX, HEX, DEC, HHX},          /* 133 */
 356 {"rename",      2, DEC, NOV, STG, STG},                         /* 134 */
 357 {"uname",       1, DEC, NOV, HEX},                              /* 135 */
 358 {"setegid",     1, DEC, NOV, UNS},                              /* 136 */
 359 {"sysconfig",   1, DEC, NOV, CNF},                              /* 137 */
 360 {"adjtime",     2, DEC, NOV, HEX, HEX},                         /* 138 */
 361 {"sysinfo",     3, DEC, NOV, INF, RST, DEC},                    /* 139 */
 362 {"sharefs",     3, DEC, NOV, DEC, HEX, DEC},                    /* 140 */
 363 {"seteuid",     1, DEC, NOV, UNS},                              /* 141 */
 364 {"forksys",     2, DEC, NOV, DEC, HHX},                         /* 142 */
 365 { NULL,         8, HEX, HEX, HEX, HEX, HEX, HEX, HEX, HEX, HEX},
 366 {"sigtimedwait", 3, DEC, NOV, HEX, HEX, HEX},                   /* 144 */
 367 {"lwp_info",    1, DEC, NOV, HEX},                              /* 145 */
 368 {"yield",       0, DEC, NOV},                                   /* 146 */
 369 { NULL,         8, HEX, HEX, HEX, HEX, HEX, HEX, HEX, HEX, HEX},
 370 {"lwp_sema_post", 1, DEC, NOV, HEX},                            /* 148 */
 371 {"lwp_sema_trywait", 1, DEC, NOV, HEX},                         /* 149 */
 372 {"lwp_detach",  1, DEC, NOV, DEC},                              /* 150 */
 373 {"corectl",     4, DEC, NOV, DEC, HEX, HEX, HEX},               /* 151 */
 374 {"modctl",      5, DEC, NOV, MOD, HEX, HEX, HEX, HEX},          /* 152 */
 375 {"fchroot",     1, DEC, NOV, DEC},                              /* 153 */
 376 { NULL,         8, HEX, HEX, HEX, HEX, HEX, HEX, HEX, HEX, HEX},
 377 {"vhangup",     0, DEC, NOV},                                   /* 155 */
 378 {"gettimeofday", 1, DEC, NOV, HEX},                             /* 156 */
 379 {"getitimer",   2, DEC, NOV, ITM, HEX},                         /* 157 */
 380 {"setitimer",   3, DEC, NOV, ITM, HEX, HEX},                    /* 158 */
 381 {"lwp_create",  3, DEC, NOV, HEX, LWF, HEX},                    /* 159 */
 382 {"lwp_exit",    0, DEC, NOV},                                   /* 160 */
 383 {"lwp_suspend", 1, DEC, NOV, DEC},                              /* 161 */
 384 {"lwp_continue", 1, DEC, NOV, DEC},                             /* 162 */
 385 {"lwp_kill",    2, DEC, NOV, DEC, SIG},                         /* 163 */
 386 {"lwp_self",    0, DEC, NOV},                                   /* 164 */
 387 {"lwp_sigmask", 5, HEX, HEX, SPM, HEX, HEX, HEX, HEX},          /* 165 */
 388 {"lwp_private", 3, HEX, NOV, DEC, DEC, HEX},                    /* 166 */
 389 {"lwp_wait",    2, DEC, NOV, DEC, HEX},                         /* 167 */
 390 {"lwp_mutex_wakeup", 2, DEC, NOV, HEX, DEC},                    /* 168 */
 391 { NULL,         8, HEX, HEX, HEX, HEX, HEX, HEX, HEX, HEX, HEX},
 392 {"lwp_cond_wait", 4, DEC, NOV, HEX, HEX, DEC},                  /* 170 */
 393 {"lwp_cond_signal", 1, DEC, NOV, HEX},                          /* 171 */
 394 {"lwp_cond_broadcast", 1, DEC, NOV, HEX},                       /* 172 */
 395 {"pread",       4, DEC, NOV, DEC, IOB, UNS, DEX},               /* 173 */
 396 {"pwrite",      4, DEC, NOV, DEC, IOB, UNS, DEX},               /* 174 */
 397 {"llseek",      4, LLO, NOV, DEC, LLO, HID, WHN},               /* 175 */
 398 {"inst_sync",   2, DEC, NOV, STG, DEC},                         /* 176 */
 399 {"brand",       6, DEC, NOV, DEC, HEX, HEX, HEX, HEX, HEX},     /* 177 */
 400 {"kaio",        7, DEC, NOV, AIO, HEX, HEX, HEX, HEX, HEX, HEX}, /* 178 */
 401 {"cpc",         5, DEC, NOV, CPC, DEC, HEX, HEX, HEX},          /* 179 */
 402 {"lgrpsys",     3, DEC, NOV, DEC, DEC, HEX},                    /* 180 */
 403 {"rusagesys",   5, DEC, NOV, DEC, HEX, DEC, HEX, HEX},          /* 181 */
 404 {"portfs",      6, HEX, HEX, DEC, HEX, HEX, HEX, HEX, HEX},     /* 182 */
 405 {"pollsys",     4, DEC, NOV, HEX, DEC, HEX, HEX},               /* 183 */
 406 {"labelsys",    2, DEC, NOV, DEC, HEX},                         /* 184 */
```



```
 407 {"acl",         4, DEC, NOV, STG, ACL, DEC, HEX},               /* 185 */
 408 {"auditsys",    4, DEC, NOV, AUD, HEX, HEX, HEX},               /* 186 */
 409 {"processor_bind", 4, DEC, NOV, IDT, DEC, DEC, HEX},           /* 187 */
 410 {"processor_info", 2, DEC, NOV, DEC, HEX},                      /* 188 */
 411 {"p_online",    2, DEC, NOV, DEC, DEC},                         /* 189 */
 412 {"sigqueue",    5, DEC, NOV, DEC, SIG, HEX, SQC, DEC},          /* 190 */
 413 {"clock_gettime", 2, DEC, NOV, DEC, HEX},                       /* 191 */
 414 {"clock_settime", 2, DEC, NOV, DEC, HEX},                       /* 192 */
 415 {"clock_getres", 2, DEC, NOV, DEC, HEX},                        /* 193 */
 416 {"timer_create", 3, DEC, NOV, DEC, HEX, HEX},                   /* 194 */
 417 {"timer_delete", 1, DEC, NOV, DEC},                             /* 195 */
 418 {"timer_settime", 4, DEC, NOV, DEC, DEC, HEX, HEX},             /* 196 */
 419 {"timer_gettime", 2, DEC, NOV, DEC, HEX},                       /* 197 */
 420 {"timer_getoverrun", 1, DEC, NOV, DEC},                         /* 198 */
 421 {"nanosleep",   2, DEC, NOV, HEX, HEX},                         /* 199 */
 422 {"facl",        4, DEC, NOV, DEC, ACL, DEC, HEX},               /* 200 */
 423 {"door",        6, DEC, NOV, DEC, HEX, HEX, HEX, HEX, DEC},     /* 201 */
 424 {"setreuid",    2, DEC, NOV, UN1, UN1},                         /* 202 */
 425 {"setregid",    2, DEC, NOV, UN1, UN1},                         /* 203 */
 426 {"install_utrap", 3, DEC, NOV, DEC, HEX, HEX},                  /* 204 */
 427 {"signotify",   3, DEC, NOV, DEC, HEX, HEX},                    /* 205 */
 428 {"schedctl",    0, HEX, NOV},                                   /* 206 */
 429 {"pset",        5, DEC, NOV, DEC, HEX, HEX, HEX, HEX},          /* 207 */
 430 {"sparc_utrap_install", 5, DEC, NOV, UTT, UTH, UTH, HEX, HEX},  /* 208 */
 431 {"resolvepath", 3, DEC, NOV, STG, RLK, DEC},                    /* 209 */
 432 {"lwp_mutex_timedlock", 3, DEC, NOV, HEX, HEX, HEX},            /* 210 */
 433 {"lwp_sema_timedwait", 3, DEC, NOV, HEX, HEX, DEC},             /* 211 */
 434 {"lwp_rwlock_sys", 3, DEC, NOV, DEC, HEX, HEX},                 /* 212 */
 435 {"getdents64",  3, DEC, NOV, DEC, HEX, UNS},                    /* 213 */
 436 {"mmap64",      7, HEX, NOV, HEX, UNS, MPR, MTY, DEC, LLO, HID}, /* 214 */
 437 {"stat64",      2, DEC, NOV, STG, HEX},                         /* 215 */
 438 {"lstat64",     2, DEC, NOV, STG, HEX},                         /* 216 */
 439 {"fstat64",     2, DEC, NOV, DEC, HEX},                         /* 217 */
 440 {"statvfs64",   2, DEC, NOV, STG, HEX},                         /* 218 */
 441 {"fstatvfs64",  2, DEC, NOV, DEC, HEX},                         /* 219 */
 442 {"setrlimit64", 2, DEC, NOV, RLM, HEX},                         /* 220 */
 443 {"getrlimit64", 2, DEC, NOV, RLM, HEX},                         /* 221 */
 444 {"pread64",     5, DEC, NOV, DEC, IOB, UNS, LLO, HID},          /* 222 */
 445 {"pwrite64",    5, DEC, NOV, DEC, IOB, UNS, LLO, HID},          /* 223 */
 446 { NULL,         8, HEX, HEX, HEX, HEX, HEX, HEX, HEX, HEX, HEX},
 447 {"open64",      3, DEC, NOV, STG, OPN, OCT},                    /* 225 */
 448 {"rpcmod",      3, DEC, NOV, DEC, HEX},                         /* 226 */
 449 {"zone",        5, DEC, NOV, DEC, HEX, HEX, HEX, HEX},          /* 227 */
 450 {"autofssys",   2, DEC, NOV, DEC, HEX},                         /* 228 */
 451 {"getcwd",      3, DEC, NOV, RST, DEC},                         /* 229 */
 452 {"so_socket",   5, DEC, NOV, PFM, SKT, SKP, STG, SKV},          /* 230 */
 453 {"so_socketpair", 1, DEC, NOV, HEX},                            /* 231 */
 454 {"bind",        4, DEC, NOV, DEC, HEX, DEC, SKV},               /* 232 */
 455 {"listen",      3, DEC, NOV, DEC, DEC, SKV},                    /* 233 */
 456 {"accept",      5, DEC, NOV, DEC, HEX, HEX, SKV, ACF},          /* 234 */
 457 {"connect",     4, DEC, NOV, DEC, HEX, DEC, SKV},               /* 235 */
 458 {"shutdown",    3, DEC, NOV, DEC, SHT, SKV},                    /* 236 */
 459 {"recv",        4, DEC, NOV, DEC, IOB, DEC, DEC},               /* 237 */
 460 {"recvfrom",    6, DEC, NOV, DEC, IOB, DEC, DEC, HEX, HEX},     /* 238 */
 461 {"recvmsg",     3, DEC, NOV, DEC, HEX, DEC},                    /* 239 */
 462 {"send",        4, DEC, NOV, DEC, IOB, DEC, DEC},               /* 240 */
 463 {"sendmsg",     3, DEC, NOV, DEC, HEX, DEC},                    /* 241 */
 464 {"sendto",      6, DEC, NOV, DEC, IOB, DEC, DEC, HEX, DEC},     /* 242 */
 465 {"getpeername", 4, DEC, NOV, DEC, HEX, HEX, SKV},               /* 243 */
 466 {"getsockname", 4, DEC, NOV, DEC, HEX, HEX, SKV},               /* 244 */
 467 {"getsockopt",  6, DEC, NOV, DEC, SOL, SON, HEX, HEX, SKV},     /* 245 */
 468 {"setsockopt",  6, DEC, NOV, DEC, SOL, SON, HEX, DEC, SKV},     /* 246 */
 469 {"sockconfig",  5, DEC, NOV, DEC, HEX, HEX, HEX, HEX},          /* 247 */
 470 {"ntp_gettime", 1, DEC, NOV, HEX},                              /* 248 */
 471 {"ntp_adjtime", 1, DEC, NOV, HEX},                              /* 249 */
 472 {"lwp_mutex_unlock", 1, DEC, NOV, HEX},                         /* 250 */
```

```
 473 {"lwp_mutex_trylock", 2, DEC, NOV, HEX, HEX},                   /* 251 */
 474 {"lwp_mutex_register", 2, DEC, NOV, HEX, HEX},                  /* 252 */
 475 {"cladm",         3, DEC, NOV, CLC, CLF, HEX},                  /* 253 */
 476 {"uucopy",        3, DEC, NOV, HEX, HEX, UNS},                  /* 254 */
 477 {"umount2",       2, DEC, NOV, STG, MTF},                       /* 255 */
 478 { NULL, -1, DEC, NOV},
 479 };
```

unchanged_portion_omitted_

```
**********************************************************
    4902 Wed May 27 19:49:09 2015
new/usr/src/lib/brand/ipkg/zone/config.xml
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
```

```
<?xml version="1.0"?>

<!--
 CDDL HEADER START

 The contents of this file are subject to the terms of the
 Common Development and Distribution License (the "License").
 You may not use this file except in compliance with the License.

 You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 or http://www.opensolaris.org/os/licensing.
 See the License for the specific language governing permissions
 and limitations under the License.

 When distributing Covered Code, include this CDDL HEADER in each
 file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 If applicable, add the following below this CDDL HEADER, with the
 fields enclosed by brackets "[]" replaced with your own identifying
 information: Portions Copyright [yyyy] [name of copyright owner]

 CDDL HEADER END

 Copyright (c) 2006, 2010, Oracle and/or its affiliates. All rights reserved.

 DO NOT EDIT THIS FILE.
 Copyright 2014 Nexenta Systems, Inc. All rights reserved.
-->

<!DOCTYPE brand PUBLIC "-//Sun Microsystems Inc//DTD Brands//EN"
    "file:///usr/share/lib/xml/dtd/brand.dtd.1">

<brand name="ipkg">
        <modname></modname>

        <initname>/sbin/init</initname>
        <login_cmd>/usr/bin/login -z %Z %u</login_cmd>
        <forcedlogin_cmd>/usr/bin/login -z %Z -f %u</forcedlogin_cmd>
        <user_cmd>/usr/bin/getent passwd %u</user_cmd>

        <!-- We may not be able to do the create in pkg(1) proper. -->
        <install>/usr/lib/brand/ipkg/pkgcreatezone -z %z -R %R</install>
        <installopts>a:c:d:e:hk:P:p:suv</installopts>
        <boot></boot>
        <sysboot>/usr/lib/brand/ipkg/prestate %z %R 2 0</sysboot>
        <halt></halt>
        <shutdown>/usr/sbin/shutdown -y -g0 -i5</shutdown>
        <verify_cfg></verify_cfg>
        <verify_adm></verify_adm>
        <postclone></postclone>
        <postinstall></postinstall>
        <attach>/usr/lib/brand/ipkg/attach %z %R</attach>
        <detach>/usr/lib/brand/ipkg/detach -z %z -R %R</detach>
        <clone>/usr/lib/brand/ipkg/clone -z %z -R %R</clone>
        <uninstall>/usr/lib/brand/ipkg/uninstall %z %R</uninstall>
        <prestatechange>/usr/lib/brand/ipkg/prestate %z %R</prestatechange>
        <poststatechange>/usr/lib/brand/ipkg/poststate %z %R</poststatechange>
        <query>/usr/lib/brand/shared/query %z %R</query>
```



```
        <privilege set="default" name="contract_event" />
        <privilege set="default" name="contract_identity" />
        <privilege set="default" name="contract_observer" />
        <privilege set="default" name="file_chown" />
        <privilege set="default" name="file_chown_self" />
        <privilege set="default" name="file_dac_execute" />
        <privilege set="default" name="file_dac_read" />
        <privilege set="default" name="file_dac_search" />
        <privilege set="default" name="file_dac_write" />
        <privilege set="default" name="file_owner" />
        <privilege set="default" name="file_setid" />
        <privilege set="default" name="ipc_dac_read" />
        <privilege set="default" name="ipc_dac_write" />
        <privilege set="default" name="ipc_owner" />
        <privilege set="default" name="net_bindmlp" />
        <privilege set="default" name="net_icmpaccess" />
        <privilege set="default" name="net_mac_aware" />
        <privilege set="default" name="net_observability" />
        <privilege set="default" name="net_privaddr" />
        <privilege set="default" name="net_rawaccess" ip-type="exclusive" />
        <privilege set="default" name="proc_chroot" />
        <privilege set="default" name="sys_audit" />
        <privilege set="default" name="proc_audit" />
        <privilege set="default" name="proc_lock_memory" />
        <privilege set="default" name="proc_owner" />
        <privilege set="default" name="proc_secflags" />
#endif /* ! codereview */
        <privilege set="default" name="proc_setid" />
        <privilege set="default" name="proc_taskid" />
        <privilege set="default" name="sys_acct" />
        <privilege set="default" name="sys_admin" />
        <privilege set="default" name="sys_ip_config" ip-type="exclusive" />
        <privilege set="default" name="sys_iptun_config" ip-type="exclusive" />
        <privilege set="default" name="sys_mount" />
        <privilege set="default" name="sys_nfs" />
        <privilege set="default" name="sys_resource" />
        <privilege set="default" name="sys_ppp_config" ip-type="exclusive" />

        <privilege set="prohibited" name="dtrace_kernel" />
        <privilege set="prohibited" name="proc_zone" />
        <privilege set="prohibited" name="sys_config" />
        <privilege set="prohibited" name="sys_devices" />
        <privilege set="prohibited" name="sys_ip_config" ip-type="shared" />
        <privilege set="prohibited" name="sys_linkdir" />
        <privilege set="prohibited" name="sys_net_config" />
        <privilege set="prohibited" name="sys_res_config" />
        <privilege set="prohibited" name="sys_suser_compat" />
        <privilege set="prohibited" name="xvm_control" />
        <privilege set="prohibited" name="virt_manage" />
        <privilege set="prohibited" name="sys_ppp_config" ip-type="shared" />

        <privilege set="required" name="proc_exec" />
        <privilege set="required" name="proc_fork" />
        <privilege set="required" name="sys_ip_config" ip-type="exclusive" />
        <privilege set="required" name="sys_mount" />
</brand>
```

```
**********************************************************
    4906 Wed May 27 19:49:09 2015
new/usr/src/lib/brand/labeled/zone/config.xml
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
```

```
<?xml version="1.0"?>

<!--
 CDDL HEADER START

 The contents of this file are subject to the terms of the
 Common Development and Distribution License (the "License").
 You may not use this file except in compliance with the License.

 You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 or http://www.opensolaris.org/os/licensing.
 See the License for the specific language governing permissions
 and limitations under the License.

 When distributing Covered Code, include this CDDL HEADER in each
 file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 If applicable, add the following below this CDDL HEADER, with the
 fields enclosed by brackets "[]" replaced with your own identifying
 information: Portions Copyright [yyyy] [name of copyright owner]

 CDDL HEADER END

 Copyright (c) 2006, 2010, Oracle and/or its affiliates. All rights reserved.

 DO NOT EDIT THIS FILE.
 Copyright 2014 Nexenta Systems, Inc. All rights reserved.
-->

<!DOCTYPE brand PUBLIC "-//Sun Microsystems Inc//DTD Brands//EN"
    "file:///usr/share/lib/xml/dtd/brand.dtd.1">

<brand name="labeled">
        <modname></modname>

        <initname>/sbin/init</initname>
        <login_cmd>/usr/bin/login -z %Z %u</login_cmd>
        <forcedlogin_cmd>/usr/bin/login -z %Z -f %u</forcedlogin_cmd>

        <user_cmd>/usr/bin/getent passwd %u</user_cmd>

        <!-- We may not be able to do the create in pkg(1) proper. -->
        <install>/usr/lib/brand/ipkg/pkgcreatezone -z %z -R %R</install>
        <installopts>a:c:d:e:hk:P:p:suv</installopts>
        <boot></boot>
        <sysboot>/usr/lib/brand/ipkg/prestate %z %R 2 0</sysboot>
        <halt></halt>
        <shutdown>/usr/sbin/shutdown -y -g0 -i5</shutdown>
        <verify_cfg></verify_cfg>
        <verify_adm></verify_adm>
        <postclone></postclone>
        <postinstall></postinstall>
        <attach>/usr/lib/brand/ipkg/attach %z %R</attach>
        <detach>/usr/lib/brand/ipkg/detach -z %z -R %R</detach>
        <clone>/usr/lib/brand/ipkg/clone -z %z -R %R</clone>
        <uninstall>/usr/lib/brand/ipkg/uninstall %z %R</uninstall>
        <prestatechange>/usr/lib/brand/ipkg/prestate %z %R</prestatechange>
        <poststatechange>/usr/lib/brand/ipkg/poststate %z %R</poststatechange>
        <query>/usr/lib/brand/shared/query %z %R</query>
```



```
        <privilege set="default" name="contract_event" />
        <privilege set="default" name="contract_identity" />
        <privilege set="default" name="contract_observer" />
        <privilege set="default" name="file_chown" />
        <privilege set="default" name="file_chown_self" />
        <privilege set="default" name="file_dac_execute" />
        <privilege set="default" name="file_dac_read" />
        <privilege set="default" name="file_dac_search" />
        <privilege set="default" name="file_dac_write" />
        <privilege set="default" name="file_owner" />
        <privilege set="default" name="file_setid" />
        <privilege set="default" name="ipc_dac_read" />
        <privilege set="default" name="ipc_dac_write" />
        <privilege set="default" name="ipc_owner" />
        <privilege set="default" name="net_bindmlp" />
        <privilege set="default" name="net_icmpaccess" />
        <privilege set="default" name="net_mac_aware" />
        <privilege set="default" name="net_observability" />
        <privilege set="default" name="net_privaddr" />
        <privilege set="default" name="net_rawaccess" ip-type="exclusive" />
        <privilege set="default" name="proc_chroot" />
        <privilege set="default" name="sys_audit" />
        <privilege set="default" name="proc_audit" />
        <privilege set="default" name="proc_lock_memory" />
        <privilege set="default" name="proc_owner" />
        <privilege set="default" name="proc_secflags" />
#endif /* ! codereview */
        <privilege set="default" name="proc_setid" />
        <privilege set="default" name="proc_taskid" />
        <privilege set="default" name="sys_acct" />
        <privilege set="default" name="sys_admin" />
        <privilege set="default" name="sys_ip_config" ip-type="exclusive" />
        <privilege set="default" name="sys_iptun_config" ip-type="exclusive" />
        <privilege set="default" name="sys_mount" />
        <privilege set="default" name="sys_nfs" />
        <privilege set="default" name="sys_resource" />
        <privilege set="default" name="sys_ppp_config" ip-type="exclusive" />

        <privilege set="prohibited" name="dtrace_kernel" />
        <privilege set="prohibited" name="proc_zone" />
        <privilege set="prohibited" name="sys_config" />
        <privilege set="prohibited" name="sys_devices" />
        <privilege set="prohibited" name="sys_ip_config" ip-type="shared" />
        <privilege set="prohibited" name="sys_linkdir" />
        <privilege set="prohibited" name="sys_net_config" />
        <privilege set="prohibited" name="sys_res_config" />
        <privilege set="prohibited" name="sys_suser_compat" />
        <privilege set="prohibited" name="xvm_control" />
        <privilege set="prohibited" name="virt_manage" />
        <privilege set="prohibited" name="sys_ppp_config" ip-type="shared" />

        <privilege set="required" name="proc_exec" />
        <privilege set="required" name="proc_fork" />
        <privilege set="required" name="sys_ip_config" ip-type="exclusive" />
        <privilege set="required" name="sys_mount" />
</brand>
```

```
**********************************************************
    4650 Wed May 27 19:49:10 2015
new/usr/src/lib/brand/sn1/zone/config.xml
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
```

```
<?xml version="1.0"?>

<!--
 CDDL HEADER START

 The contents of this file are subject to the terms of the
 Common Development and Distribution License (the "License").
 You may not use this file except in compliance with the License.

 You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 or http://www.opensolaris.org/os/licensing.
 See the License for the specific language governing permissions
 and limitations under the License.

 When distributing Covered Code, include this CDDL HEADER in each
 file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 If applicable, add the following below this CDDL HEADER, with the
 fields enclosed by brackets "[]" replaced with your own identifying
 information: Portions Copyright [yyyy] [name of copyright owner]

 CDDL HEADER END

 Copyright (c) 2006, 2010, Oracle and/or its affiliates. All rights reserved.

 DO NOT EDIT THIS FILE.
-->

<!DOCTYPE brand PUBLIC "-//Sun Microsystems Inc//DTD Brands//EN"
    "file:///usr/share/lib/xml/dtd/brand.dtd.1">

<brand name="sn1">
        <modname>sn1_brand</modname>

        <initname>/sbin/init</initname>
        <login_cmd>/usr/bin/login -z %Z %u</login_cmd>
        <forcedlogin_cmd>/usr/bin/login -z %Z -f %u</forcedlogin_cmd>

        <user_cmd>/usr/bin/getent passwd %u</user_cmd>

        <install>/usr/lib/brand/ipkg/pkgcreatezone -z %z -R %R</install>
        <boot>/usr/lib/brand/sn1/sn1_boot %R</boot>
        <sysboot>/usr/lib/brand/ipkg/prestate %z %R 2 0</sysboot>
        <verify_cfg></verify_cfg>
        <verify_adm></verify_adm>
        <attach>/usr/lib/brand/ipkg/attach %z %R</attach>
        <detach>/usr/lib/brand/ipkg/detach -z %z -R %R</detach>
        <clone>/usr/lib/brand/ipkg/clone -z %z -R %R</clone>
        <uninstall>/usr/lib/brand/ipkg/uninstall %z %R</uninstall>
        <prestatechange>/usr/lib/brand/ipkg/prestate %z %R</prestatechange>
        <poststatechange>/usr/lib/brand/ipkg/poststate %z %R</poststatechange>
        <query>/usr/lib/brand/shared/query %z %R</query>

        <privilege set="default" name="contract_event" />
        <privilege set="default" name="contract_identity" />
        <privilege set="default" name="contract_observer" />
        <privilege set="default" name="file_chown" />
        <privilege set="default" name="file_chown_self" />
        <privilege set="default" name="file_dac_execute" />
```



```
        <privilege set="default" name="file_dac_read" />
        <privilege set="default" name="file_dac_search" />
        <privilege set="default" name="file_dac_write" />
        <privilege set="default" name="file_owner" />
        <privilege set="default" name="file_setid" />
        <privilege set="default" name="ipc_dac_read" />
        <privilege set="default" name="ipc_dac_write" />
        <privilege set="default" name="ipc_owner" />
        <privilege set="default" name="net_bindmlp" />
        <privilege set="default" name="net_icmpaccess" />
        <privilege set="default" name="net_mac_aware" />
        <privilege set="default" name="net_observability" />
        <privilege set="default" name="net_privaddr" />
        <privilege set="default" name="net_rawaccess" ip-type="exclusive" />
        <privilege set="default" name="proc_chroot" />
        <privilege set="default" name="sys_audit" />
        <privilege set="default" name="proc_audit" />
        <privilege set="default" name="proc_lock_memory" />
        <privilege set="default" name="proc_owner" />
        <privilege set="default" name="proc_secflags" />
#endif /* ! codereview */
        <privilege set="default" name="proc_setid" />
        <privilege set="default" name="proc_taskid" />
        <privilege set="default" name="sys_acct" />
        <privilege set="default" name="sys_admin" />
        <privilege set="default" name="sys_ip_config" ip-type="exclusive" />
        <privilege set="default" name="sys_iptun_config" ip-type="exclusive" />
        <privilege set="default" name="sys_mount" />
        <privilege set="default" name="sys_nfs" />
        <privilege set="default" name="sys_resource" />
        <privilege set="default" name="sys_ppp_config" ip-type="exclusive" />

        <privilege set="prohibited" name="dtrace_kernel" />
        <privilege set="prohibited" name="proc_zone" />
        <privilege set="prohibited" name="sys_config" />
        <privilege set="prohibited" name="sys_devices" />
        <privilege set="prohibited" name="sys_ip_config" ip-type="shared" />
        <privilege set="prohibited" name="sys_linkdir" />
        <privilege set="prohibited" name="sys_net_config" />
        <privilege set="prohibited" name="sys_res_config" />
        <privilege set="prohibited" name="sys_suser_compat" />
        <privilege set="prohibited" name="xvm_control" />
        <privilege set="prohibited" name="virt_manage" />
        <privilege set="prohibited" name="sys_ppp_config" ip-type="shared" />

        <privilege set="required" name="proc_exec" />
        <privilege set="required" name="proc_fork" />
        <privilege set="required" name="sys_ip_config" ip-type="exclusive" />
        <privilege set="required" name="sys_mount" />
</brand>
```

```
**********************************************************
   28376 Wed May 27 19:49:10 2015
new/usr/src/lib/libbsm/audit_event.txt
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
   1 #
   2 # Copyright (c) 1992, 2010, Oracle and/or its affiliates. All rights reserved.
   3 #
   4 #
   5 # CDDL HEADER START
   6 #
   7 # The contents of this file are subject to the terms of the
   8 # Common Development and Distribution License (the "License").
   9 # You may not use this file except in compliance with the License.
  10 #
  11 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
  12 # or http://www.opensolaris.org/os/licensing.
  13 # See the License for the specific language governing permissions
  14 # and limitations under the License.
  15 #
  16 # When distributing Covered Code, include this CDDL HEADER in each
  17 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  18 # If applicable, add the following below this CDDL HEADER, with the
  19 # fields enclosed by brackets "[]" replaced with your own identifying
  20 # information: Portions Copyright [yyyy] [name of copyright owner]
  21 #
  22 # CDDL HEADER END
  23 #
  24 # Audit Event Database
  25 #
  26 # File Format:
  27 #
  28 # event number:event name:event description:event classes (comma separated)
  29 #
  30 # Used to map audit events to audit classes for preselection and post-selection.
  31 # Used by TCB programs that write audit records to preselect audit events
  32 #     based on event to class mappings.
  33 #
  34 # NOTE: several events are obsolete but must continue to be defined here for
  35 # compatibility reasons. Obsolete events are defined in the "no" (invalid)
  36 # class to indicate they will not be generated.  Other events in the "no"
  37 # class which are not obsolete (but are in this class for other reasons),
  38 # are individually noted with a comment for explanation.
  39 #
  40 # System Adminstrators: Do NOT modify or add events with an event number less
  41 #   than 32768. These are reserved by the system.
  42 #
  43 #     0         Reserved as an invalid event number.
  44 #     1 - 2047  Reserved for the Solaris Kernel events.
  45 #  2048 - 32767 Reserved for the Solaris TCB programs.
  46 # 32768 - 65535 Available for third party TCB applications.
  47 #
  48 #
  49 # Allocation of reserved kernel events:
  50 # (NOTE: the kernel event table, and possibly MAX_KEVENTS, must be updated
  51 # in audit_kevents.h when changes are made to kernel events.)
  52 #           1 -   511   allocated for Solaris
  53 #         512 -  2047   (reserved but not allocated)
  54 #
  55 # Allocation of user level audit events:
  56 #        2048 -  5999   (reserved but not allocated)
  57 #        6000 -  9999   allocated for Solaris
  58 #       10000 - 32767   (reserved but not allocated)
```



```
  59 #       32768 - 65535   (Available for third party TCB applications)
  60 #
  61 #
  62 # kernel audit events
  63 #           1 -   511   allocated for Solaris
  64 #
  65 0:AUE_NULL:indir system call:no
  66 1:AUE_EXIT:exit(2):ps
  67 2:AUE_FORKALL:forkall(2):ps
  68 # AUE_OPEN is a placeholder and will not be generated
  69 3:AUE_OPEN:open(2) - place holder:no
  70 4:AUE_CREAT:creat(2):no
  71 5:AUE_LINK:link(2):fc
  72 6:AUE_UNLINK:unlink(2):fd
  73 7:AUE_EXEC:exec(2):no
  74 8:AUE_CHDIR:chdir(2):pm
  75 9:AUE_MKNOD:mknod(2):fc
  76 10:AUE_CHMOD:chmod(2):fm
  77 11:AUE_CHOWN:chown(2):fm
  78 12:AUE_UMOUNT:umount(2) - old version:as
  79 13:AUE_JUNK:junk:no
  80 14:AUE_ACCESS:access(2):fa
  81 15:AUE_KILL:kill(2):pm
  82 16:AUE_STAT:stat(2):fa
  83 17:AUE_LSTAT:lstat(2):fa
  84 18:AUE_ACCT:acct(2):as
  85 19:AUE_MCTL:mctl(2):no
  86 20:AUE_REBOOT:reboot(2):no
  87 21:AUE_SYMLINK:symlink(2):fc
  88 22:AUE_READLINK:readlink(2):fr
  89 23:AUE_EXECVE:execve(2):ps,ex
  90 24:AUE_CHROOT:chroot(2):pm
  91 25:AUE_VFORK:vfork(2):ps
  92 26:AUE_SETGROUPS:setgroups(2):pm
  93 27:AUE_SETPGRP:setpgrp(2):pm
  94 28:AUE_SWAPON:swapon(2):no
  95 29:AUE_SETHOSTNAME:sethostname(2):no
  96 30:AUE_FCNTL:fcntl(2):fm
  97 31:AUE_SETPRIORITY:setpriority(2):no
  98 32:AUE_CONNECT:connect(2):nt
  99 33:AUE_ACCEPT:accept(2):nt
 100 34:AUE_BIND:bind(2):nt
 101 35:AUE_SETSOCKOPT:setsockopt(2):nt
 102 36:AUE_VTRACE:vtrace(2):no
 103 37:AUE_SETTIMEOFDAY:settimeofday(2):no
 104 38:AUE_FCHOWN:fchown(2):fm
 105 39:AUE_FCHMOD:fchmod(2):fm
 106 40:AUE_SETREUID:setreuid(2):pm
 107 41:AUE_SETREGID:setregid(2):pm
 108 42:AUE_RENAME:rename(2):fc,fd
 109 43:AUE_TRUNCATE:truncate(2):no
 110 44:AUE_FTRUNCATE:ftruncate(2):no
 111 45:AUE_FLOCK:flock(2):no
 112 46:AUE_SHUTDOWN:shutdown(2):nt
 113 47:AUE_MKDIR:mkdir(2):fc
 114 48:AUE_RMDIR:rmdir(2):fd
 115 49:AUE_UTIMES:utimes(2):fm
 116 50:AUE_ADJTIME:adjtime(2):as
 117 51:AUE_SETRLIMIT:setrlimit(2):ua
 118 52:AUE_KILLPG:killpg(2):no
 119 53:AUE_NFS_SVC:nfs_svc(2):no
 120 54:AUE_STATFS:statfs(2):fa
 121 55:AUE_FSTATFS:fstatfs(2):fa
 122 56:AUE_UNMOUNT:unmount(2):no
 123 57:AUE_ASYNC_DAEMON:async_daemon(2):no
 124 58:AUE_NFS_GETFH:nfs_getfh(2):no
```

```
125 59:AUE_SETDOMAINNAME:setdomainname(2):no
126 60:AUE_QUOTACTL:quotactl(2):no
127 61:AUE_EXPORTFS:exportfs(2):no
128 62:AUE_MOUNT:mount(2):as
129 # AUE_SEMSYS is a placeholder and will not be generated
130 63:AUE_SEMSYS:semsys(2) - place holder:no
131 # AUE_MSGSYS is a placeholder and will not be generated
132 64:AUE_MSGSYS:msgsys(2) - place holder:no
133 # AUE_SHMSYS is a placeholder and will not be generated
134 65:AUE_SHMSYS:shmsys(2) - place holder:no
135 66:AUE_BSMSYS:bsmsys(2) - place holder:no
136 67:AUE_RFSSYS:rfssys(2) - place holder:no
137 68:AUE_FCHDIR:fchdir(2):pm
138 69:AUE_FCHROOT:fchroot(2):pm
139 70:AUE_VPIXSYS:vpixsys(2) - place holder:no
140 71:AUE_PATHCONF:pathconf(2):fa
141 72:AUE_OPEN_R:open(2) - read:fr
142 73:AUE_OPEN_RC:open(2) - read,creat:fc,fr
143 74:AUE_OPEN_RT:open(2) - read,trunc:fd,fr
144 75:AUE_OPEN_RTC:open(2) - read,creat,trunc:fc,fd,fr
145 76:AUE_OPEN_W:open(2) - write:fw
146 77:AUE_OPEN_WC:open(2) - write,creat:fc,fw
147 78:AUE_OPEN_WT:open(2) - write,trunc:fd,fw
148 79:AUE_OPEN_WTC:open(2) - write,creat,trunc:fc,fd,fw
149 80:AUE_OPEN_RW:open(2) - read,write:fr,fw
150 81:AUE_OPEN_RWC:open(2) - read,write,creat:fc,fw,fr
151 82:AUE_OPEN_RWT:open(2) - read,write,trunc:fd,fr,fw
152 83:AUE_OPEN_RWTC:open(2) - read,write,creat,trunc:fc,fd,fw,fr
153 84:AUE_MSGCTL:msgctl(2) - illegal command:ip
154 85:AUE_MSGCTL_RMID:msgctl(2) - IPC_RMID command:ip
155 86:AUE_MSGCTL_SET:msgctl(2) - IPC_SET command:ip
156 87:AUE_MSGCTL_STAT:msgctl(2) - IPC_STAT command:ip
157 88:AUE_MSGGET:msgget(2):ip
158 89:AUE_MSGRCV:msgrcv(2):ip
159 90:AUE_MSGSND:msgsnd(2):ip
160 91:AUE_SHMCTL:shmctl(2) - illegal command:ip
161 92:AUE_SHMCTL_RMID:shmctl(2) - IPC_RMID command:ip
162 93:AUE_SHMCTL_SET:shmctl(2) - IPC_SET command:ip
163 94:AUE_SHMCTL_STAT:shmctl(2) - IPC_STAT command:ip
164 95:AUE_SHMGET:shmget(2):ip
165 96:AUE_SHMAT:shmat(2):ip
166 97:AUE_SHMDT:shmdt(2):ip
167 98:AUE_SEMCTL:semctl(2) - illegal command:ip
168 99:AUE_SEMCTL_RMID:semctl(2) - IPC_RMID command:ip
169 100:AUE_SEMCTL_SET:semctl(2) - IPC_SET command:ip
170 101:AUE_SEMCTL_STAT:semctl(2) - IPC_STAT command:ip
171 102:AUE_SEMCTL_GETNCNT:semctl(2) - GETNCNT command:ip
172 103:AUE_SEMCTL_GETPID:semctl(2) - GETPID command:ip
173 104:AUE_SEMCTL_GETVAL:semctl(2) - GETVAL command:ip
174 105:AUE_SEMCTL_GETALL:semctl(2) - GETALL command:ip
175 106:AUE_SEMCTL_GETZCNT:semctl(2) - GETZCNT command:ip
176 107:AUE_SEMCTL_SETVAL:semctl(2) - SETVAL command:ip
177 108:AUE_SEMCTL_SETALL:semctl(2) - SETALL command:ip
178 109:AUE_SEMGET:semget(2):ip
179 110:AUE_SEMOP:semop(2):ip
180 111:AUE_CORE:process dumped core:fc
181 112:AUE_CLOSE:close(2):cl
182 113:AUE_SYSTEMBOOT:system booted:na
183 114:AUE_ASYNC_DAEMON_EXIT:async_daemon(2) exited:no
184 115:AUE_NFSSVC_EXIT:nfssvc(2) exited:no
185 116:AUE_PFEXEC:execve(2) with pfexec enabled:ps,ex,ua,as
186 117:AUE_OPEN_S:open(2) - search:fr
187 118:AUE_OPEN_E:open(2) - exec:fr

189 130:AUE_GETAUID:getauid(2):aa
190 131:AUE_SETAUID:setauid(2):aa
```



```
191 132:AUE_GETAUDIT:getaudit(2):aa
192 133:AUE_SETAUDIT:setaudit(2):aa
193 134:AUE_GETUSERAUDIT:getuseraudit(2):no
194 135:AUE_SETUSERAUDIT:setuseraudit(2):no
195 # AUE_AUDITSVC is a placeholder and will not be generated
196 136:AUE_AUDITSVC:auditsvc(2) - place holder:no
197 # AUE_AUDITON is a placeholder and will not be generated
198 138:AUE_AUDITON:auditon(2) - place holder:no
199 139:AUE_AUDITON_GTERMID:auditon(2) - GETTERMID command:no
200 140:AUE_AUDITON_STERMID:auditon(2) - SETTERMID command:no
201 141:AUE_AUDITON_GPOLICY:auditon(2) - get audit policy flags:aa
202 142:AUE_AUDITON_SPOLICY:auditon(2) - set audit policy flags:as
203 143:AUE_AUDITON_GESTATE:auditon(2) - GESTATE command:no
204 144:AUE_AUDITON_SESTATE:auditon(2) - SESTATE command:no
205 145:AUE_AUDITON_GQCTRL:auditon(2) - get queue control parameters:as
206 146:AUE_AUDITON_SQCTRL:auditon(2) - set queue control parameters:as
207 147:AUE_GETKERNSTATE:getkernstate(2):no
208 148:AUE_SETKERNSTATE:setkernstate(2):no
209 149:AUE_GETPORTAUDIT:getportaudit(2):no
210 150:AUE_AUDITSTAT:auditstat(2):no
211 153:AUE_ENTERPROM:enter prom:na
212 154:AUE_EXITPROM:exit prom:na
213 158:AUE_IOCTL:ioctl(2):io
214 173:AUE_ONESIDE:one-sided session record:no
215 174:AUE_MSGGETL:msggetl(2):no
216 175:AUE_MSGRCVL:msgrcvl(2):no
217 176:AUE_MSGSNDL:msgsndl(2):no
218 177:AUE_SEMGETL:semgetl(2):no
219 178:AUE_SHMGETL:shmgetl(2):no
220 183:AUE_SOCKET:socket(2):nt
221 184:AUE_SENDTO:sendto(2):nt
222 # AUE_PIPE is a potentially very high-volume event, use with caution
223 185:AUE_PIPE:pipe(2):no
224 186:AUE_SOCKETPAIR:socketpair(2):no
225 187:AUE_SEND:send(2):no
226 188:AUE_SENDMSG:sendmsg(2):nt
227 189:AUE_RECV:recv(2):no
228 190:AUE_RECVMSG:recvmsg(2):nt
229 191:AUE_RECVFROM:recvfrom(2):nt
230 # AUE_READ is a potentially very high-volume event, use with caution
231 192:AUE_READ:read(2):no
232 193:AUE_GETDENTS:getdents(2):no
233 194:AUE_LSEEK:lseek(2):no
234 # AUE_WRITE is a potentially very high-volume event, use with caution
235 195:AUE_WRITE:write(2):no
236 196:AUE_WRITEV:writev(2):no
237 197:AUE_NFS:nfs server:no
238 198:AUE_READV:readv(2):no
239 199:AUE_OSTAT:old stat(2):no
240 200:AUE_SETUID:setuid(2):pm
241 201:AUE_STIME:old stime(2):as
242 202:AUE_UTIME:old utime(2):no
243 203:AUE_NICE:old nice(2):pm
244 204:AUE_OSETPGRP:old setpgrp(2):no
245 205:AUE_SETGID:old setgid(2):pm
246 206:AUE_READL:readl(2):no
247 207:AUE_READVL:readvl(2):no
248 208:AUE_FSTAT:fstat(2):no
249 209:AUE_DUP2:dup2(2):no
250 # AUE_MMAP is a potentially very high-volume event, use with caution
251 210:AUE_MMAP:mmap(2):no
252 # AUE_AUDIT is a potentially very high-volume event, use with caution
253 211:AUE_AUDIT:audit(2):no
254 212:AUE_PRIOCNTLSYS:priocntlsys(2):pm
255 213:AUE_MUNMAP:munmap(2):cl
256 214:AUE_SETEGID:setegid(2):pm
```

```
215:AUE_SETEUID:seteuid(2):pm
216:AUE_PUTMSG:putmsg(2):nt
217:AUE_GETMSG:getmsg(2):nt
218:AUE_PUTPMSG:putpmsg(2):nt
219:AUE_GETPMSG:getpmsg(2):nt
# AUE_AUDITSYS is a placeholder and will not be generated
220:AUE_AUDITSYS:audit system calls place holder:no
221:AUE_AUDITON_GETKMASK:auditon(2) - get kernel mask:aa
222:AUE_AUDITON_SETKMASK:auditon(2) - set kernel mask:as
223:AUE_AUDITON_GETCWD:auditon(2) - get current working directory:aa,as
224:AUE_AUDITON_GETCAR:auditon(2) - get current active root:aa,as
225:AUE_AUDITON_GETSTAT:auditon(2) - get audit statistics:as
226:AUE_AUDITON_SETSTAT:auditon(2) - reset audit statistics:as
227:AUE_AUDITON_SETUMASK:auditon(2) - set mask per audit uid:as
228:AUE_AUDITON_SETSMASK:auditon(2) - set mask per session ID:as
229:AUE_AUDITON_GETCOND:auditon(2) - get audit state:aa
230:AUE_AUDITON_SETCOND:auditon(2) - set audit state:as
231:AUE_AUDITON_GETCLASS:auditon(2) - get event class:aa,as
232:AUE_AUDITON_SETCLASS:auditon(2) - set event class:as
233:AUE_FUSERS:utssys(2) - fusers:fa
234:AUE_STATVFS:statvfs(2):fa
235:AUE_XSTAT:xstat(2):no
236:AUE_LXSTAT:lxstat(2):no
237:AUE_LCHOWN:lchown(2):fm
238:AUE_MEMCNTL:memcntl(2):ot
239:AUE_SYSINFO:sysinfo(2):as
240:AUE_XMKNOD:xmknod(2):no
241:AUE_FORK1:fork1(2):ps
# AUE_MODCTL is a placeholder and will not be generated
242:AUE_MODCTL:modctl(2) system call place holder:no
243:AUE_MODLOAD:modctl(2) - load module:as
244:AUE_MODUNLOAD:modctl(2) - unload module:as
# AUE_MODCONFIG is a place holder and will not be generated
245:AUE_MODCONFIG:modctl(2) - no longer generated:no
246:AUE_MODADDMAJ:modctl(2) - bind module:as
247:AUE_SOCKACCEPT:getmsg-accept:nt
248:AUE_SOCKCONNECT:putmsg-connect:nt
249:AUE_SOCKSEND:putmsg-send:nt
250:AUE_SOCKRECEIVE:getmsg-receive:nt
251:AUE_ACLSET:acl(2) - SETACL command:fm
252:AUE_FACLSET:facl(2) - SETACL command:fm
# AUE_DOORFS is a placeholder and will not be generated
253:AUE_DOORFS:doorfs(2) - system call place holder:no
254:AUE_DOORFS_DOOR_CALL:doorfs(2) - DOOR_CALL:ip
255:AUE_DOORFS_DOOR_RETURN:doorfs(2) - DOOR_RETURN:ip
256:AUE_DOORFS_DOOR_CREATE:doorfs(2) - DOOR_CREATE:ip
257:AUE_DOORFS_DOOR_REVOKE:doorfs(2) - DOOR_REVOKE:ip
258:AUE_DOORFS_DOOR_INFO:doorfs(2) - DOOR_INFO:ip
259:AUE_DOORFS_DOOR_CRED:doorfs(2) - DOOR_CRED:ip
260:AUE_DOORFS_DOOR_BIND:doorfs(2) - DOOR_BIND:ip
261:AUE_DOORFS_DOOR_UNBIND:doorfs(2) - DOOR_UNBIND:ip
262:AUE_P_ONLINE:p_online(2):as
263:AUE_PROCESSOR_BIND:processor_bind(2):as
264:AUE_INST_SYNC:inst_sync(2):as
265:AUE_SOCKCONFIG:configure socket:nt
266:AUE_SETAUDIT_ADDR:setaudit_addr(2):aa
267:AUE_GETAUDIT_ADDR:getaudit_addr(2):aa
268:AUE_UMOUNT2:umount2(2):as
# AUE_FSAT and all AUE_OPENAT_* codes are obsolete and will not be generated
269:AUE_FSAT:fsat(2) - place holder:no
270:AUE_OPENAT_R:openat(2) - read:no
271:AUE_OPENAT_RC:openat(2) - read,creat:no
272:AUE_OPENAT_RT:openat(2) - read,trunc:no
273:AUE_OPENAT_RTC:openat(2) - read,creat,trunc:no
274:AUE_OPENAT_W:openat(2) - write:no
275:AUE_OPENAT_WC:openat(2) - write,creat:no
```



```
276:AUE_OPENAT_WT:openat(2) - write,trunc:no
277:AUE_OPENAT_WTC:openat(2) - write,creat,trunc:no
278:AUE_OPENAT_RW:openat(2) - read,write:no
279:AUE_OPENAT_RWC:openat(2) - read,write,creat:no
280:AUE_OPENAT_RWT:openat(2) - read,write,trunc:no
281:AUE_OPENAT_RWTC:openat(2) - read,write,creat,trunc:no
282:AUE_RENAMEAT:renameat(2):no
283:AUE_FSTATAT:fstatat(2):no
284:AUE_FCHOWNAT:fchownat(2):no
285:AUE_FUTIMESAT:futimesat(2):no
286:AUE_UNLINKAT:unlinkat(2):no
287:AUE_CLOCK_SETTIME:clock_settime(3RT):as
288:AUE_NTP_ADJTIME:ntp_adjtime(2):as
289:AUE_SETPPRIV:setppriv(2):pm
290:AUE_MODDEVPLCY:modctl(2) - configure device policy:as
291:AUE_MODADDPRIV:modctl(2) - configure additional privilege:as
292:AUE_CRYPTOADM:kernel cryptographic framework:as
293:AUE_CONFIGKSSL:configure kernel SSL:as
294:AUE_BRANDSYS:brandsys(2):ot
295:AUE_PF_POLICY_ADDRULE:Add IPsec policy rule:as
296:AUE_PF_POLICY_DELRULE:Delete IPsec policy rule:as
297:AUE_PF_POLICY_CLONE:Clone IPsec policy:as
298:AUE_PF_POLICY_FLIP:Flip IPsec policy:as
299:AUE_PF_POLICY_FLUSH:Flush IPsec policy rules:as
300:AUE_PF_POLICY_ALGS:Update IPsec algorithms:as
# AUE_PORTFS is a placeholder and won't be generated.
301:AUE_PORTFS:portfs(2) - file events source - place holder:no
#
302:AUE_LABELSYS_TNRH:tnrh(2) - config TN remote host cache:as
303:AUE_LABELSYS_TNRHTP:tnrhtp(2) - config TN remote host template cache:as
304:AUE_LABELSYS_TNMLP:tnmlp(2) - config TN multi-level port entry:as
#
305:AUE_PORTFS_ASSOCIATE:portfs(2) - file events source - PORT_ASSOCIATE:fa
306:AUE_PORTFS_DISSOCIATE:portfs(2) - file events source - PORT_DISSOCIATE:fa
#
307:AUE_SETSID:setsid(2):pm
308:AUE_SETPGID:setpgid(2):pm
309:AUE_FACCESSAT:faccessat(2):no
310:AUE_AUDITON_GETAMASK:auditon(2) - get default user preselection mask:aa
311:AUE_AUDITON_SETAMASK:auditon(2) - set default user preselection mask:as
312:AUE_PSECFLAGS:psecflags(2) - set process security flags:pm
#endif /* ! codereview */
#
# user level audit events
#       2048 -  6143    Reserved
#
#       6000 -  7999    allocated for Solaris
#
6144:AUE_at_create:at-create atjob:ua
6145:AUE_at_delete:at-delete atjob (at or atrm):ua
6146:AUE_at_perm:at-permission:no
6147:AUE_cron_invoke:cron-invoke:ua
6148:AUE_crontab_create:crontab-crontab created:ua
6149:AUE_crontab_delete:crontab-crontab deleted:ua
6150:AUE_crontab_perm:crontab-persmisson:no
6151:AUE_inetd_connect:inetd connect:na
6152:AUE_login:login - local:lo
6153:AUE_logout:logout:lo
6154:AUE_telnet:login - telnet:lo
6155:AUE_rlogin:login - rlogin:lo
6156:AUE_mountd_mount:mount:na
6157:AUE_mountd_umount:unmount:na
6158:AUE_rshd:rsh access:lo
6159:AUE_su:su:lo
6160:AUE_halt_solaris:halt(1m):ss
6161:AUE_reboot_solaris:reboot(1m):ss
```

```
6162:AUE_rexecd:rexecd:lo
6163:AUE_passwd:passwd:lo
6164:AUE_rexd:rexd:lo
6165:AUE_ftpd:ftp access:lo
6166:AUE_init_solaris:init(1m):ss
6167:AUE_uadmin_solaris:uadmin(1m):no
6168:AUE_shutdown_solaris:shutdown(1b):ss
6169:AUE_poweroff_solaris:poweroff(1m):ss
6170:AUE_crontab_mod:crontab-modify:ua
6171:AUE_ftpd_logout:ftp logout:lo
6172:AUE_ssh:login - ssh:lo
6173:AUE_role_login:role login:lo
6180:AUE_prof_cmd:profile command:ua,as
6181:AUE_filesystem_add:add filesystem:as
6182:AUE_filesystem_delete:delete filesystem:as
6183:AUE_filesystem_modify:modify filesystem:as
6184:AUE_network_add:add network attributes:as
6185:AUE_network_delete:delete network attributes:as
6186:AUE_network_modify:modify network attributes:as
6187:AUE_printer_add:add printer:as
6188:AUE_printer_delete:delete printer:as
6189:AUE_printer_modify:modify printer:as
6190:AUE_scheduledjob_add:add scheduled job:ua
6191:AUE_scheduledjob_delete:delete scheduled job:ua
6192:AUE_scheduledjob_modify:modify scheduled job:ua
6193:AUE_serialport_add:add serial port:as
6194:AUE_serialport_delete:delete serial port:as
6195:AUE_serialport_modify:modify serial port:as
6196:AUE_usermgr_add:add user/user attributes:ua
6197:AUE_usermgr_delete:delete user/user attributes:ua
6198:AUE_usermgr_modify:modify user/user attributes:ua
6199:AUE_uauth:authorization used:ua,as
6200:AUE_allocate_succ:allocate-device success:ot
6201:AUE_allocate_fail:allocate-device failure:ot
6202:AUE_deallocate_succ:deallocate-device success:ot
6203:AUE_deallocate_fail:deallocate-device failure:ot
6205:AUE_listdevice_succ:allocate-list devices success:ot
6206:AUE_listdevice_fail:allocate-list devices failure:ot
6207:AUE_create_user:create user:no
6208:AUE_modify_user:modify user:no
6209:AUE_delete_user:delete user:no
6210:AUE_disable_user:disable user:no
6211:AUE_enable_user:enable user:no
6212:AUE_newgrp_login:newgrp login:lo
6213:AUE_admin_authenticate:admin login:lo
6214:AUE_kadmind_auth:authenticated kadmind request:ua
6215:AUE_kadmind_unauth:unauthenticated kadmind req:ua
6216:AUE_krb5kdc_as_req:kdc authentication svc request:ap
6217:AUE_krb5kdc_tgs_req:kdc tkt-grant svc request:ap
6218:AUE_krb5kdc_tgs_req_2ndtktmm:kdc tgs 2ndtkt mismtch:ap
6219:AUE_krb5kdc_tgs_req_alt_tgt:kdc tgs issue alt tgt:ap
6220:AUE_smserverd:smserverd:ot
6221:AUE_screenlock:screenlock - lock:lo
6222:AUE_screenunlock:screenlock - unlock:lo
6223:AUE_zone_state:zoneadmd:ss
6224:AUE_inetd_copylimit:inetd copylimit:na
6225:AUE_inetd_failrate:inetd failrate:na
6226:AUE_inetd_ratelimit:inetd ratelimit:na
6227:AUE_zlogin:login - zlogin:lo
6228:AUE_su_logout:su logout:lo
6229:AUE_role_logout:role logout:lo
6230:AUE_attach:attach device:ot
6231:AUE_detach:detach device:ot
6232:AUE_remove:remove/eject device:ot
6233:AUE_pool_import:import device into pool:ot
6234:AUE_pool_export:export device from pool:ot
```



```
6235:AUE_dladm_create_secobj:create network security object:as,cy
6236:AUE_dladm_delete_secobj:delete network security object:as,cy
6237:AUE_uadmin_shutdown:uadmin(1m) - shutdown:ss
6238:AUE_uadmin_reboot:uadmin(1m) - reboot:ss
6239:AUE_uadmin_dump:uadmin(1m) - dump:ss
6240:AUE_uadmin_freeze:uadmin(1m) - freeze:ss
6241:AUE_uadmin_remount:uadmin(1m) - remount:ss
6242:AUE_uadmin_ftrace:uadmin(1m) - ftrace:ss
6243:AUE_uadmin_swapctl:uadmin(1m) - swapctl:ss
6244:AUE_smbd_session:smbd(1m) session setup:lo
6245:AUE_smbd_logoff:smbd(1m) session logoff:lo
6246:AUE_vscan_quarantine:vscand(1m) quarantine infected file:na
6247:AUE_ndmp_connect:ndmp connect:na
6248:AUE_ndmp_disconnect:ndmp disconnect:na
6249:AUE_ndmp_backup:ndmp backup:na
6250:AUE_ndmp_restore:ndmp restore:na
6251:AUE_cpu_ondemand:set ondemand CPU freq governor:ss
6252:AUE_cpu_performance:set max CPU freq governor:ss
6253:AUE_cpu_threshold:set CPU freq threshold:ss
6254:AUE_uadmin_thaw:uadmin(1m) - thaw after freeze:ss,na
6255:AUE_uadmin_config:uadmin(1m) - config:ss

#
# SMF(5) svc.configd events (svcadm(1M) related)
#
6260:AUE_smf_enable:persistently enable service instance:ss
6261:AUE_smf_tmp_enable:temporarily enable service instance:ss
6262:AUE_smf_disable:persistently disable service instance:ss
6263:AUE_smf_tmp_disable:temporarily disable service instance:ss
6264:AUE_smf_restart:restart service instance:ss
6265:AUE_smf_refresh:refresh service instance:ss
6266:AUE_smf_clear:clear service instance state:ss
6267:AUE_smf_degrade:set service instance degraded state:ss
6268:AUE_smf_immediate_degrade:immediately set service instance degraded state:s
6269:AUE_smf_maintenance:set service instance persistent maintenance state:ss
6270:AUE_smf_immediate_maintenance:immediately set service instance persistent m
6271:AUE_smf_immtmp_maintenance:immediately set service instance temporary maint
6272:AUE_smf_tmp_maintenance:set service instance maintenance temporary state:ss
6273:AUE_smf_milestone:set service management facility milestone:ss
#
# SMF(5) svc.configd miscellaneous events
#
6275:AUE_smf_read_prop:read restricted access property value:as
#
# SMF(5) svc.configd events (svccfg(1M) related)
#
6280:AUE_smf_create:create service instance object:as
6281:AUE_smf_delete:delete service instance object:as
6282:AUE_smf_create_pg:create persistent service property group:as
6283:AUE_smf_create_npg:create non-persistent service property group:as
6284:AUE_smf_delete_pg:delete persistent service property group:as
6285:AUE_smf_delete_npg:delete non-persistent service property group:as
6286:AUE_smf_create_snap:create repository snapshot:as
6287:AUE_smf_delete_snap:delete repository snapshot:as
6288:AUE_smf_attach_snap:attach repository snapshot:as
6289:AUE_smf_annotation:annotate transaction:as,ss
6290:AUE_smf_create_prop:create service instance property:as
6291:AUE_smf_change_prop:change service instance property:as
6292:AUE_smf_delete_prop:delete service instance property:as
#
# nwamd(1M) events
#
6300:AUE_nwam_enable:enable nwam profile object:ss
6301:AUE_nwam_disable:disable nwam profile object:ss
#
# ilbd(1M) events
```

```
#
6310:AUE_ilb_create_healthcheck:create ILB health check:as
6311:AUE_ilb_delete_healthcheck:delete ILB health check:as
6312:AUE_ilb_create_rule:create ILB rule:as
6313:AUE_ilb_delete_rule:delete ILB rule:as
6314:AUE_ilb_disable_rule:disable ILB rule:as
6315:AUE_ilb_enable_rule:enable ILB rule:as
6316:AUE_ilb_add_server:add ILB server:as
6317:AUE_ilb_disable_server:disable ILB server:as
6318:AUE_ilb_enable_server:enable ILB server:as
6319:AUE_ilb_remove_server:remove ILB server:as
6320:AUE_ilb_create_servergroup:create ILB server group:as
6321:AUE_ilb_delete_servergroup:delete ILB server group:as
#
# netcfgd(1M) events
#
6330:AUE_netcfg_update:create or modify configuration object:ss
6331:AUE_netcfg_remove:remove configuration object from repository:ss
#
# TCSD(8) events
#
6400:AUE_tpm_takeownership:take ownership of TPM:as
6401:AUE_tpm_clearowner:clear ownership of TPM:as
6402:AUE_tpm_setoperatorauth:set TPM operator authorization:as
6403:AUE_tpm_setownerinstall:set TPM ownership flag:as
6404:AUE_tpm_selftestfull:test all TPM protected capabilities:as
6405:AUE_tpm_certifyselftest:perform full TPM self-test:as
6406:AUE_tpm_continueselftest:complete TPM self-test:as
6407:AUE_tpm_ownersetdisable:change the status of TPM disable flag:as
6408:AUE_tpm_ownerclear:perform the clear operation under TPM owner auth:as
6409:AUE_tpm_disableownerclear:disable TPM_OwnerClear command permanently:as
6410:AUE_tpm_forceclear:perform TPM clear operation under physical access:as
6411:AUE_tpm_disableforceclear:disable ForceClear execution until next startup:a
6412:AUE_tpm_physicaldisable:disable TPM physical presence:as
6413:AUE_tpm_physicalenable:enable TPM physical presence:as
6414:AUE_tpm_physicaldeactivate:set TPM deactivated flag:as
6415:AUE_tpm_settempdeactivated:set volatile TPM deactivated flag to TRUE:as
6416:AUE_tpm_settempdeactivated2:set volatile TPM deactivated flag TRUE with aut
6417:AUE_tpm_physicalpresence:set the TPM physical presence flag:as
6418:AUE_tpm_fieldupgrade:update TPM protected capabilities:as
6419:AUE_tpm_resetlockvalue:reset TPM failed authorization attempt lock:as
#
# hotplugd(1m) events
#
6500:AUE_hotplug_state:change hotplug connection state:ss
6501:AUE_hotplug_set:set hotplug bus private options:ss

#
# Trusted Extensions events:
#
9035:AUE_sl_change:Workspace label change:ap
9036:AUE_file_relabel:relabel file:fm
9037:AUE_file_copy:file copy:fm
9038:AUE_file_move:file move:no
9039:AUE_sel_mgr_xfer:selection manager transfer:fm
9101:AUE_ClientConnect:client connection to x server:lo
9102:AUE_ClientDisconnect:client disconn. from x server:lo
9120:AUE_ChangeProperty:XChangeProperty(3X11):xc
9121:AUE_DeleteProperty:XDeleteProperty(3X11):xc
9137:AUE_GrabServer:XGrabServer(3X11):ot
9138:AUE_UngrabServer:XUngrabServer(3X11):ot
9146:AUE_SetFontPath:XSetFontPath(3X11):ot
9173:AUE_InstallColormap:XInstallColormap(3X11):ot
9174:AUE_UninstallColormap:XUninstallColormap(3X11):xp
9193:AUE_SetScreenSaver:XSetScreenSaver(3X11):xp
9194:AUE_ChangeHosts:XChangeHosts(3X11):ot
```



```
9195:AUE_SetAccessControl:XSetAccessControl(3X11):xp
9196:AUE_SetCloseDownMode:XSetCloseDownMode(3X11):xs
9197:AUE_KillClient:XKillClient(3X11):xc
9202:AUE_XExtensions:X server extensions:xp
9103:AUE_CreateWindow:XCreateWindow(3X11):xc
9104:AUE_ChangeWindowAttributes:XChangeWindowAttributes(3X11):xp
9105:AUE_GetWindowAttributes:XGetWindowAttributes(3X11):xp
9106:AUE_DestroyWindow:XDestroyWindow(3X11):xc
9107:AUE_DestroySubwindows:XDestroySubwindows(3X11):xc
9108:AUE_ChangeSaveSet:XChangeSaveSet(3X11):xp
9109:AUE_ReparentWindow:XReparentWindow(3X11):xp
9110:AUE_MapWindow:XMapWindow(3X11):xp
9111:AUE_MapSubwindows:XMapSubwindows(3X11):xp
9112:AUE_UnmapWindow:XUnmapWindow(3X11):xp
9113:AUE_UnmapSubwindows:XUnmapSubwindows(3X11):xp
9114:AUE_ConfigureWindow:XConfigureWindow(3X11):xp
9115:AUE_CirculateWindow:XCirculateWindow(3X11):xp
9116:AUE_GetGeometry:XGetGeometry(3X11):xp
9117:AUE_QueryTree:XQueryTree(3X11):xp
9118:AUE_InternAtom:XInternAtom(3X11):xs
9119:AUE_GetAtomName:XGetAtomName(3X11):xs
9122:AUE_GetProperty:XGetProperty(3X11):xp
9123:AUE_ListProperties:XListProperties(3X11):xp
9124:AUE_SetSelectionOwner:XSetSelectionOwner(3X11):xp
9125:AUE_GetSelectionOwner:XGetSelectionOwner(3X11):xs
9126:AUE_ConvertSelection:XConvertSelection(3X11):xs
9127:AUE_SendEvent:XSendEvent(3X11):xs
9128:AUE_GrabPointer:XGrabPointer(3X11):xs
9129:AUE_UngrabPointer:XUngrabPointer(3X11):xs
9130:AUE_GrabButton:XGrabButton(3X11):xp
9131:AUE_UngrabButton:XUngrabButton(3X11):xs
9132:AUE_ChangeActivePointerGrab:XChangeActivePointerGrab(3X11):xs
9133:AUE_GrabKeyboard:XGrabKeyboard(3X11):xp
9134:AUE_UngrabKeyboard:XUngrabKeyboard(3X11):xs
9135:AUE_GrabKey:XGrabKey(3X11):xp
9136:AUE_UngrabKey:XUngrabKey(3X11):xp
9139:AUE_QueryPointer:XQueryPointer(3X11):xp
9140:AUE_GetMotionEvents:XGetMotionEvents(3X11):xp
9141:AUE_TranslateCoords:XTranslateCoords(3X11):xp
9142:AUE_WarpPointer:XWarpPointer(3X11):xs
9143:AUE_SetInputFocus:XSetInputFocus(3X11):xs
9144:AUE_GetInputFocus:XGetInputFocus(3X11):xs
9145:AUE_QueryKeymap:XQueryKeymap(3X11):xp
9147:AUE_FreePixmap:XFreePixmap(3X11):xc
9148:AUE_ChangeGC:XChangeGC(3X11):xp
9149:AUE_CopyGC:XCopyGC(3X11):xp
9150:AUE_SetDashes:XSetDashes(3X11):xp
9151:AUE_SetClipRectangles:XSetClipRectangles(3X11):xp
9152:AUE_FreeGC:XFreeGC(3X11):xc
9153:AUE_ClearArea:XClearArea(3X11):xp
9154:AUE_CopyArea:XCopyArea(3X11):xs
9155:AUE_CopyPlane:XCopyPlane(3X11):xs
9156:AUE_PolyPoint:XPolyPoint(3X11):xp
9157:AUE_PolyLine:XPolyLine(3X11):xp
9158:AUE_PolySegment:XPolySegment(3X11):xp
9159:AUE_PolyRectangle:XPolyRectangle(3X11):xs
9160:AUE_PolyArc:XPolyArc(3X11):xp
9161:AUE_FillPolygon:XFillPolygon(3X11):xp
9162:AUE_PolyFillRectangle:XPolyFillRectangle(3X11):xp
9163:AUE_PolyFillArc:XPolyFillArc(3X11):xp
9164:AUE_PutImage:XPutImage(3X11):xp
9165:AUE_GetImage:XGetImage(3X11):xs
9166:AUE_PolyText8:XPolyText8(3X11):xp
9167:AUE_PolyText16:XPolyText16(3X11):xp
9168:AUE_ImageText8:XImageText8(3X11):xp
9169:AUE_ImageText16:XImageText16(3X11):xp
```

```
653 9170:AUE_CreateColormap:XCreateColormap(3X11):xc
654 9171:AUE_FreeColormap:XFreeColormap(3X11):xc
655 9172:AUE_CopyColormapAndFree:XCopyColormapAndFree(3X11):xp
656 9175:AUE_ListInstalledColormaps:XListInstalledColormaps(3X11):xs
657 9176:AUE_AllocColor:XAllocColor(3X11):xc
658 9177:AUE_AllocNamedColor:XAllocNamedColor(3X11):xc
659 9178:AUE_AllocColorCells:XAllocColorCells(3X11):xc
660 9179:AUE_AllocColorPlanes:XAllocColorPlanes(3X11):xc
661 9180:AUE_FreeColors:XFreeColors(3X11):xc
662 9181:AUE_StoreColors:XStoreColors(3X11):xp
663 9182:AUE_StoreNamedColor:XStoreNamedColor(3X11):xp
664 9183:AUE_QueryColors:XQueryColors(3X11):xp
665 9184:AUE_LookupColor:XLookupColor(3X11):xp
666 9185:AUE_CreateCursor:XCreateCursor(3X11):xc
667 9186:AUE_CreateGlyphCursor:XCreateGlyphCursor(3X11):xc
668 9187:AUE_FreeCursor:XFreeCursor(3X11):xc
669 9188:AUE_RecolorCursor:XRecolorCursor(3X11):xp
670 9189:AUE_ChangeKeyboardMapping:XChangeKeyboardMapping(3X11):xs
671 9190:AUE_ChangeKeyboardControl:XChangeKeyboardControl(3X11):xs
672 9191:AUE_Bell:XBell(3X11):xs
673 9192:AUE_ChangePointerControl:XChangePointerControl(3X11):xs
674 9198:AUE_RotateProperties:XRotateProperties(3X11):xp
675 9199:AUE_ForceScreenSaver:XForceScreenSaver(3X11):xp
676 9200:AUE_SetPointerMapping:XSetPointerMapping(3X11):xs
677 9201:AUE_SetModifierMapping:XSetModifierMapping(3X11):xs
```

```
**********************************************************
   22820 Wed May 27 19:49:11 2015
new/usr/src/lib/libc/amd64/Makefile
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
#
# Copyright (c) 2004, 2010, Oracle and/or its affiliates. All rights reserved.
# Copyright (c) 2015, Joyent, Inc. All rights reserved.
#
# Copyright (c) 2013, OmniTI Computer Consulting, Inc. All rights reserved.
# Copyright 2013 Garrett D'Amore <garrett@damore.org>
# Copyright 2011 Nexenta Systems, Inc.  All rights reserved.
# Use is subject to license terms.
#

LIBCBASE=       .
LIBCDIR=        $(SRC)/lib/libc
LIBRARY=        libc.a
LIB_PIC=        libc_pic.a
VERS=           .1
CPP=            /usr/lib/cpp
TARGET_ARCH=    amd64

# objects are grouped by source directory

# local objects
STRETS=

CRTOBJS=                        \
        cerror.o

DYNOBJS=

FPOBJS=                         \
        fpgetmask.o             \
        fpgetround.o            \
        fpsetmask.o             \
        fpsetround.o            \
        fpstart.o               \
        ieee.o

I386FPOBJS=                     \
        _D_cplx_div.o           \
```



```
        _D_cplx_div_ix.o        \
        _D_cplx_div_rx.o        \
        _D_cplx_lr_div.o        \
        _D_cplx_lr_div_ix.o     \
        _D_cplx_lr_div_rx.o     \
        _D_cplx_mul.o           \
        _F_cplx_div.o           \
        _F_cplx_div_ix.o        \
        _F_cplx_div_rx.o        \
        _F_cplx_lr_div.o        \
        _F_cplx_lr_div_ix.o     \
        _F_cplx_lr_div_rx.o     \
        _F_cplx_mul.o           \
        _X_cplx_div.o           \
        _X_cplx_div_ix.o        \
        _X_cplx_div_rx.o        \
        _X_cplx_lr_div.o        \
        _X_cplx_lr_div_ix.o     \
        _X_cplx_lr_div_rx.o     \
        _X_cplx_mul.o

FPASMOBJS=                      \
        __xgetRD.o              \
        _xtoll.o                \
        _xtoull.o               \
        _base_il.o              \
        fpcw.o                  \
        fpgetsticky.o           \
        fpsetsticky.o

ATOMICOBJS=                     \
        atomic.o

CHACHAOBJS=                     \
        chacha.o

XATTROBJS=                      \
        xattr_common.o
COMOBJS=                        \
        bcmp.o                  \
        bcopy.o                 \
        bsearch.o               \
        bzero.o                 \
        qsort.o                 \
        strtol.o                \
        strtoul.o               \
        strtoll.o               \
        strtoull.o

GENOBJS=                        \
        _getsp.o                \
        abs.o                   \
        alloca.o                \
        arc4random.o            \
        arc4random_uniform.o    \
        attrat.o                \
        byteorder.o             \
        cuexit.o                \
        ecvt.o                  \
        errlst.o                \
        amd64_data.o            \
        ldivide.o               \
        lock.o                  \
        makectxt.o              \
        memccpy.o               \
        memchr.o                \
```

```
125         memcmp.o                \
126         memcpy.o                \
127         memset.o                \
128         new_list.o              \
129         proc64_id.o             \
130         proc64_support.o        \
131         setjmp.o                \
132         siginfolst.o            \
133         siglongjmp.o            \
134         strcmp.o                \
135         strcpy.o                \
136         strlen.o                \
137         strncmp.o               \
138         strncpy.o               \
139         strnlen.o               \
140         sync_instruction_memory.o

142 # Preserved solely to ease maintenance of 32-bit and 64-bit library builds
143 # This macro should ALWAYS be empty; native APIs are already 'large file'.
144 COMSYSOBJS64=

146 SYSOBJS64=

148 COMSYSOBJS=                     \
149         __clock_timer.o         \
150         __getloadavg.o          \
151         __rusagesys.o           \
152         __signotify.o           \
153         __sigrt.o               \
154         __time.o                \
155         _lgrp_home_fast.o       \
156         _lgrpsys.o              \
157         _nfssys.o               \
158         _portfs.o               \
159         _pset.o                 \
160         _rpcsys.o               \
161         _sigaction.o            \
162         _so_accept.o            \
163         _so_bind.o              \
164         _so_connect.o           \
165         _so_getpeername.o       \
166         _so_getsockname.o       \
167         _so_getsockopt.o        \
168         _so_listen.o            \
169         _so_recv.o              \
170         _so_recvfrom.o          \
171         _so_recvmsg.o           \
172         _so_send.o              \
173         _so_sendmsg.o           \
174         _so_sendto.o            \
175         _so_setsockopt.o        \
176         _so_shutdown.o          \
177         _so_socket.o            \
178         _so_socketpair.o        \
179         _sockconfig.o           \
180         acct.o                  \
181         acl.o                   \
182         adjtime.o               \
183         alarm.o                 \
184         brk.o                   \
185         chdir.o                 \
186         chroot.o                \
187         cladm.o                 \
188         close.o                 \
189         execve.o                \
190         exit.o                  \
```



```
191         facl.o                  \
192         fchdir.o                \
193         fchroot.o               \
194         fdsync.o                \
195         fpathconf.o             \
196         fstatfs.o               \
197         fstatvfs.o              \
198         getcpuid.o              \
199         getdents.o              \
200         getegid.o               \
201         geteuid.o               \
202         getgid.o                \
203         getgroups.o             \
204         gethrtime.o             \
205         getitimer.o             \
206         getmsg.o                \
207         getpid.o                \
208         getpmsg.o               \
209         getppid.o               \
210         getrandom.o             \
211         getrlimit.o             \
212         getuid.o                \
213         gtty.o                  \
214         install_utrap.o         \
215         ioctl.o                 \
216         kaio.o                  \
217         kill.o                  \
218         llseek.o                \
219         lseek.o                 \
220         mmapobjsys.o            \
221         memcntl.o               \
222         mincore.o               \
223         mmap.o                  \
224         modctl.o                \
225         mount.o                 \
226         mprotect.o              \
227         munmap.o                \
228         nice.o                  \
229         ntp_adjtime.o           \
230         ntp_gettime.o           \
231         p_online.o              \
232         pathconf.o              \
233         pause.o                 \
234         pcsample.o              \
235         pipe2.o                 \
236         pollsys.o               \
237         pread.o                 \
238         preadv.o                \
239         priocntlset.o           \
240         processor_bind.o        \
241         processor_info.o        \
242         profil.o                \
243         psecflagsset.o          \
244 #endif /* ! codereview */
245         putmsg.o                \
246         putpmsg.o               \
247         pwrite.o                \
248         pwritev.o               \
249         read.o                  \
250         readv.o                 \
251         resolvepath.o           \
252         seteguid.o              \
253         setgid.o                \
254         setgroups.o             \
255         setitimer.o             \
256         setreid.o               \
```

```
257         setrlimit.o             \
258         setuid.o                \
259         sigaltstk.o             \
260         sigprocmsk.o            \
261         sigsendset.o            \
262         sigsuspend.o            \
263         statfs.o                \
264         statvfs.o               \
265         stty.o                  \
266         sync.o                  \
267         sysconfig.o             \
268         sysfs.o                 \
269         sysinfo.o               \
270         syslwp.o                \
271         times.o                 \
272         ulimit.o                \
273         umask.o                 \
274         umount2.o               \
275         utssys.o                \
276         uucopy.o                \
277         vhangup.o               \
278         waitid.o                \
279         write.o                 \
280         writev.o                \
281         yield.o

283 SYSOBJS=                        \
284         __clock_gettime.o       \
285         __getcontext.o          \
286         __uadmin.o              \
287         _lwp_mutex_unlock.o     \
288         _stack_grow.o           \
289         door.o                  \
290         forkx.o                 \
291         forkallx.o              \
292         getcontext.o            \
293         gettimeofday.o          \
294         lwp_private.o           \
295         nuname.o                \
296         syscall.o               \
297         sysi86.o                \
298         tls_get_addr.o          \
299         uadmin.o                \
300         umount.o                \
301         uname.o                 \
302         vforkx.o

304 # Preserved solely to ease maintenance of 32-bit and 64-bit library builds
305 # This macro should ALWAYS be empty; native APIs are already 'large file'.
306 PORTGEN64=

308 # objects from source under $(LIBCDIR)/port
309 PORTFP=                         \
310         __flt_decim.o           \
311         __flt_rounds.o          \
312         __tbl_10_b.o            \
313         __tbl_10_h.o            \
314         __tbl_10_s.o            \
315         __tbl_2_b.o             \
316         __tbl_2_h.o             \
317         __tbl_2_s.o             \
318         __tbl_fdq.o             \
319         __tbl_tens.o            \
320         __x_power.o             \
321         _base_sup.o             \
322         aconvert.o              \
```



```
323         decimal_bin.o           \
324         double_decim.o          \
325         econvert.o              \
326         fconvert.o              \
327         file_decim.o            \
328         finite.o                \
329         fp_data.o               \
330         func_decim.o            \
331         gconvert.o              \
332         hex_bin.o               \
333         ieee_globals.o          \
334         pack_float.o            \
335         sigfpe.o                \
336         string_decim.o

338 PORTGEN=                        \
339         _env_data.o             \
340         _xftw.o                 \
341         a64l.o                  \
342         abort.o                 \
343         addsev.o                \
344         ascii_strcasecmp.o      \
345         ascii_strncasecmp.o     \
346         assert.o                \
347         atof.o                  \
348         atoi.o                  \
349         atol.o                  \
350         atoll.o                 \
351         attropen.o              \
352         atexit.o                \
353         atfork.o                \
354         basename.o              \
355         calloc.o                \
356         catgets.o               \
357         catopen.o               \
358         cfgetispeed.o           \
359         cfgetospeed.o           \
360         cfree.o                 \
361         cfsetispeed.o           \
362         cfsetospeed.o           \
363         cftime.o                \
364         clock.o                 \
365         closedir.o              \
366         closefrom.o             \
367         confstr.o               \
368         crypt.o                 \
369         csetlen.o               \
370         ctime.o                 \
371         ctime_r.o               \
372         daemon.o                \
373         deflt.o                 \
374         directio.o              \
375         dirname.o               \
376         div.o                   \
377         drand48.o               \
378         dup.o                   \
379         env_data.o              \
380         err.o                   \
381         errno.o                 \
382         euclen.o                \
383         event_port.o            \
384         execvp.o                \
385         explicit_bzero.o        \
386         fattach.o               \
387         fdetach.o               \
388         fdopendir.o             \
```

```
389         ffs.o                   \
390         fls.o                   \
391         fmtmsg.o                \
392         ftime.o                 \
393         ftok.o                  \
394         ftw.o                   \
395         gcvt.o                  \
396         getauxv.o               \
397         getcwd.o                \
398         getdate_err.o           \
399         getdtblsize.o           \
400         getentropy.o            \
401         getenv.o                \
402         getexecname.o           \
403         getgrnam.o              \
404         getgrnam_r.o            \
405         gethostid.o             \
406         gethostname.o           \
407         gethz.o                 \
408         getisax.o               \
409         getloadavg.o            \
410         getlogin.o              \
411         getmntent.o             \
412         getnetgrent.o           \
413         get_nprocs.o            \
414         getopt.o                \
415         getopt_long.o           \
416         getpagesize.o           \
417         getpw.o                 \
418         getpwnam.o              \
419         getpwnam_r.o            \
420         getrusage.o             \
421         getspent.o              \
422         getspent_r.o            \
423         getsubopt.o             \
424         gettxt.o                \
425         getusershell.o          \
426         getut.o                 \
427         getutx.o                \
428         getvfsent.o             \
429         getwd.o                 \
430         getwidth.o              \
431         getxby_door.o           \
432         gtxt.o                  \
433         hsearch.o               \
434         iconv.o                 \
435         imaxabs.o               \
436         imaxdiv.o               \
437         index.o                 \
438         initgroups.o            \
439         insque.o                \
440         isaexec.o               \
441         isastream.o             \
442         isatty.o                \
443         killpg.o                \
444         klpdlib.o               \
445         l64a.o                  \
446         lckpwdf.o               \
447         lconstants.o            \
448         lexp10.o                \
449         lfind.o                 \
450         lfmt.o                  \
451         lfmt_log.o              \
452         lldiv.o                 \
453         llog10.o                \
454         lltostr.o               \
```



```
455         lmath.o                 \
456         localtime.o             \
457         lsearch.o               \
458         madvise.o               \
459         malloc.o                \
460         memalign.o              \
461         memmem.o                \
462         mkdev.o                 \
463         mkdtemp.o               \
464         mkfifo.o                \
465         mkstemp.o               \
466         mktemp.o                \
467         mlock.o                 \
468         mlockall.o              \
469         mon.o                   \
470         msync.o                 \
471         munlock.o               \
472         munlockall.o            \
473         ndbm.o                  \
474         nftw.o                  \
475         nlspath_checks.o        \
476         nsparse.o               \
477         nss_common.o            \
478         nss_dbdefs.o            \
479         nss_deffinder.o         \
480         opendir.o               \
481         opt_data.o              \
482         perror.o                \
483         pfmt.o                  \
484         pfmt_data.o             \
485         pfmt_print.o            \
486         pipe.o                  \
487         plock.o                 \
488         poll.o                  \
489         posix_fadvise.o         \
490         posix_fallocate.o       \
491         posix_madvise.o         \
492         posix_memalign.o        \
493         priocntl.o              \
494         privlib.o               \
495         priv_str_xlate.o        \
496         psecflags.o             \
497 #endif /* ! codereview */
498         psiginfo.o              \
499         psignal.o               \
500         pt.o                    \
501         putpwent.o              \
502         putspent.o              \
503         raise.o                 \
504         rand.o                  \
505         random.o                \
506         rctlops.o               \
507         readdir.o               \
508         readdir_r.o             \
509         realpath.o              \
510         reboot.o                \
511         regexpr.o               \
512         remove.o                \
513         rewinddir.o             \
514         rindex.o                \
515         scandir.o               \
516         seekdir.o               \
517         select.o                \
518         setlabel.o              \
519         setpriority.o           \
520         settimeofday.o          \
```

```
521         sh_locks.o              \
522         sigflag.o               \
523         siglist.o               \
524         sigsend.o               \
525         sigsetops.o             \
526         ssignal.o               \
527         stack.o                 \
528         stpcpy.o                \
529         stpncpy.o               \
530         str2sig.o               \
531         strcase_charmap.o       \
532         strcat.o                \
533         strchr.o                \
534         strchrnul.o             \
535         strcspn.o               \
536         strdup.o                \
537         strerror.o              \
538         strlcat.o               \
539         strlcpy.o               \
540         strncat.o               \
541         strndup.o               \
542         strpbrk.o               \
543         strrchr.o               \
544         strsep.o                \
545         strsignal.o             \
546         strspn.o                \
547         strstr.o                \
548         strtod.o                \
549         strtoimax.o             \
550         strtok.o                \
551         strtok_r.o              \
552         strtoumax.o             \
553         swab.o                  \
554         swapctl.o               \
555         sysconf.o               \
556         syslog.o                \
557         tcdrain.o               \
558         tcflow.o                \
559         tcflush.o               \
560         tcgetattr.o             \
561         tcgetpgrp.o             \
562         tcgetsid.o              \
563         tcsendbreak.o           \
564         tcsetattr.o             \
565         tcsetpgrp.o             \
566         tell.o                  \
567         telldir.o               \
568         tfind.o                 \
569         time_data.o             \
570         time_gdata.o            \
571         tls_data.o              \
572         truncate.o              \
573         tsdalloc.o              \
574         tsearch.o               \
575         ttyname.o               \
576         ttyslot.o               \
577         ualarm.o                \
578         ucred.o                 \
579         valloc.o                \
580         vlfmt.o                 \
581         vpfmt.o                 \
582         waitpid.o               \
583         walkstack.o             \
584         wdata.o                 \
585         xgetwidth.o             \
586         xpg4.o                  \
```



```
587         xpg6.o

589 PORTPRINT_W=                   \
590         doprnt_w.o

592 PORTPRINT=                     \
593         asprintf.o              \
594         doprnt.o                \
595         fprintf.o               \
596         printf.o                \
597         snprintf.o              \
598         sprintf.o               \
599         vfprintf.o              \
600         vprintf.o               \
601         vsnprintf.o             \
602         vsprintf.o              \
603         vwprintf.o              \
604         wprintf.o

606 # Preserved solely to ease maintenance of 32-bit and 64-bit library builds
607 # This macro should ALWAYS be empty; native APIs are already 'large file'.
608 PORTSTDIO64=

610 PORTSTDIO_W=                    \
611         doscan_w.o

613 PORTSTDIO=                      \
614         __extensions.o          \
615         _endopen.o              \
616         _filbuf.o               \
617         _findbuf.o              \
618         _flsbuf.o               \
619         _wrtchk.o               \
620         clearerr.o              \
621         ctermid.o               \
622         ctermid_r.o             \
623         cuserid.o               \
624         data.o                  \
625         doscan.o                \
626         fdopen.o                \
627         feof.o                  \
628         ferror.o                \
629         fgetc.o                 \
630         fgets.o                 \
631         fileno.o                \
632         flockf.o                \
633         flush.o                 \
634         fopen.o                 \
635         fpos.o                  \
636         fputc.o                 \
637         fputs.o                 \
638         fread.o                 \
639         fseek.o                 \
640         fseeko.o                \
641         ftell.o                 \
642         ftello.o                \
643         fwrite.o                \
644         getc.o                  \
645         getchar.o               \
646         getline.o               \
647         getpass.o               \
648         gets.o                  \
649         getw.o                  \
650         mse.o                   \
651         popen.o                 \
652         putc.o                  \
```

```
653         putchar.o               \
654         puts.o                  \
655         putw.o                  \
656         rewind.o                \
657         scanf.o                 \
658         setbuf.o                \
659         setbuffer.o             \
660         setvbuf.o               \
661         system.o                \
662         tempnam.o               \
663         tmpfile.o               \
664         tmpnam_r.o              \
665         ungetc.o                \
666         vscanf.o                \
667         vwscanf.o               \
668         wscanf.o

670 PORTI18N=                       \
671         getwchar.o              \
672         putwchar.o              \
673         putws.o                 \
674         strtows.o               \
675         wcsnlen.o               \
676         wcsstr.o                \
677         wcstoimax.o             \
678         wcstol.o                \
679         wcstoul.o               \
680         wcswcs.o                \
681         wmemchr.o               \
682         wmemcmp.o               \
683         wmemcpy.o               \
684         wmemmove.o              \
685         wmemset.o               \
686         wscat.o                 \
687         wschr.o                 \
688         wscmp.o                 \
689         wscpy.o                 \
690         wscspn.o                \
691         wsdup.o                 \
692         wslen.o                 \
693         wsncat.o                \
694         wsncmp.o                \
695         wsncpy.o                \
696         wspbrk.o                \
697         wsprintf.o              \
698         wsrchr.o                \
699         wsscanf.o               \
700         wsspn.o                 \
701         wstod.o                 \
702         wstok.o                 \
703         wstol.o                 \
704         wstoll.o                \
705         wsxfrm.o                \
706         gettext.o               \
707         gettext_gnu.o           \
708         gettext_real.o          \
709         gettext_util.o          \
710         plural_parser.o         \
711         wdresolve.o             \
712         _ctype.o                \
713         isascii.o               \
714         toascii.o

716 PORTI18N_COND=                  \
717         wcstol_longlong.o       \
718         wcstoul_longlong.o
```



```
720 PORTLOCALE=                    \
721         big5.o                  \
722         btowc.o                 \
723         collate.o               \
724         collcmp.o               \
725         euc.o                   \
726         fnmatch.o               \
727         fgetwc.o                \
728         fgetws.o                \
729         fix_grouping.o          \
730         fputwc.o                \
731         fputws.o                \
732         fwide.o                 \
733         gb18030.o               \
734         gb2312.o                \
735         gbk.o                   \
736         getdate.o               \
737         isdigit.o               \
738         iswctype.o              \
739         ldpart.o                \
740         lmessages.o             \
741         lnumeric.o              \
742         lmonetary.o             \
743         localeconv.o            \
744         localeimpl.o            \
745         mbftowc.o               \
746         mblen.o                 \
747         mbrlen.o                \
748         mbrtowc.o               \
749         mbsinit.o               \
750         mbsnrtowcs.o            \
751         mbsrtowcs.o             \
752         mbstowcs.o              \
753         mbtowc.o                \
754         mskanji.o               \
755         nextwctype.o            \
756         nl_langinfo.o           \
757         none.o                  \
758         regcomp.o               \
759         regfree.o               \
760         regerror.o              \
761         regexec.o               \
762         rune.o                  \
763         runetype.o              \
764         setlocale.o             \
765         setrunelocale.o         \
766         strcasecmp.o            \
767         strcasestr.o            \
768         strcoll.o               \
769         strfmon.o               \
770         strftime.o              \
771         strncasecmp.o           \
772         strptime.o              \
773         strxfrm.o               \
774         table.o                 \
775         timelocal.o             \
776         tolower.o               \
777         towlower.o              \
778         ungetwc.o               \
779         utf8.o                  \
780         wcrtomb.o               \
781         wcscasecmp.o            \
782         wcscoll.o               \
783         wcsftime.o              \
784         wcsnrtombs.o            \
```

```
785         wcsrtombs.o             \
786         wcswidth.o              \
787         wcstombs.o              \
788         wcsxfrm.o               \
789         wctob.o                 \
790         wctomb.o                \
791         wctrans.o               \
792         wctype.o                \
793         wcwidth.o               \
794         wscol.o

796 AIOOBJS=                        \
797         aio.o                   \
798         aio_alloc.o             \
799         posix_aio.o

801 RTOBJS=                         \
802         clock_timer.o           \
803         mqueue.o                \
804         pos4obj.o               \
805         sched.o                 \
806         sem.o                   \
807         shm.o                   \
808         sigev_thread.o

810 TPOOLOBJS=                      \
811         thread_pool.o

813 THREADSOBJS=                    \
814         alloc.o                 \
815         assfail.o               \
816         cancel.o                \
817         door_calls.o            \
818         tmem.o                  \
819         pthr_attr.o             \
820         pthr_barrier.o          \
821         pthr_cond.o             \
822         pthr_mutex.o            \
823         pthr_rwlock.o           \
824         pthread.o               \
825         rwlock.o                \
826         scalls.o                \
827         sema.o                  \
828         sigaction.o             \
829         spawn.o                 \
830         synch.o                 \
831         tdb_agent.o             \
832         thr.o                   \
833         thread_interface.o      \
834         tls.o                   \
835         tsd.o

837 THREADSMACHOBJS=                \
838         machdep.o

840 THREADSASMOBJS=                 \
841         asm_subr.o

843 UNICODEOBJS=                    \
844         u8_textprep.o           \
845         uconv.o

847 UNWINDMACHOBJS=                 \
848         call_frame_inst.o       \
849         eh_frame.o              \
850         thrp_unwind.o           \
```



```
851         unwind.o

853 pics/unwind.o:= COPTFLAG64 =

855 UNWINDASMOBJS=                  \
856         unwind_frame.o

858 # Preserved solely to ease maintenance of 32-bit and 64-bit library builds
859 # This macro should ALWAYS be empty; native APIs are already 'large file'.
860 PORTSYS64=

862 PORTSYS=                        \
863         _autofssys.o            \
864         access.o                \
865         acctctl.o               \
866         bsd_signal.o            \
867         chmod.o                 \
868         chown.o                 \
869         corectl.o               \
870         exacctsys.o             \
871         execl.o                 \
872         execle.o                \
873         execv.o                 \
874         fcntl.o                 \
875         getpagesizes.o          \
876         getpeerucred.o          \
877         inst_sync.o             \
878         issetugid.o             \
879         label.o                 \
880         link.o                  \
881         lockf.o                 \
882         lwp.o                   \
883         lwp_cond.o              \
884         lwp_rwlock.o            \
885         lwp_sigmask.o           \
886         meminfosys.o            \
887         mkdir.o                 \
888         mknod.o                 \
889         msgsys.o                \
890         nfssys.o                \
891         open.o                  \
892         pgrpsys.o               \
893         posix_sigwait.o         \
894         ppriv.o                 \
895         psetsys.o               \
896         rctlsys.o               \
897         readlink.o              \
898         rename.o                \
899         sbrk.o                  \
900         semsys.o                \
901         set_errno.o             \
902         sharefs.o               \
903         shmsys.o                \
904         sidsys.o                \
905         siginterrupt.o          \
906         signal.o                \
907         sigpending.o            \
908         sigstack.o              \
909         stat.o                  \
910         symlink.o               \
911         tasksys.o               \
912         time.o                  \
913         time_util.o             \
914         ucontext.o              \
915         unlink.o                \
916         ustat.o                 \
```

```
917         utimesys.o              \
918         zone.o

920 PORTREGEX=                      \
921         glob.o                  \
922         regcmp.o                \
923         regex.o                 \
924         wordexp.o

926 VALUES=                         \
927         values-Xa.o

929 MOSTOBJS=                       \
930         $(STRETS)               \
931         $(CRTOBJS)              \
932         $(DYNOBJS)              \
933         $(FPOBJS)               \
934         $(I386FPOBJS)           \
935         $(FPASMOBJS)            \
936         $(ATOMICOBJS)           \
937         $(CHACHAOBJS)           \
938         $(XATTROBJS)            \
939         $(COMOBJS)              \
940         $(GENOBJS)              \
941         $(PORTFP)               \
942         $(PORTGEN)              \
943         $(PORTGEN64)            \
944         $(PORTI18N)             \
945         $(PORTI18N_COND)        \
946         $(PORTLOCALE)           \
947         $(PORTPRINT)            \
948         $(PORTPRINT_W)          \
949         $(PORTREGEX)            \
950         $(PORTSTDIO)            \
951         $(PORTSTDIO64)          \
952         $(PORTSTDIO_W)          \
953         $(PORTSYS)              \
954         $(PORTSYS64)            \
955         $(AIOOBJS)              \
956         $(RTOBJS)               \
957         $(TPOOLOBJS)            \
958         $(THREADSOBJS)          \
959         $(THREADSMACHOBJS)      \
960         $(THREADSASMOBJS)       \
961         $(UNICODEOBJS)          \
962         $(UNWINDMACHOBJS)       \
963         $(UNWINDASMOBJS)        \
964         $(COMSYSOBJS)           \
965         $(SYSOBJS)              \
966         $(COMSYSOBJS64)         \
967         $(SYSOBJS64)            \
968         $(VALUES)

970 TRACEOBJS=                      \
971         plockstat.o

973 # NOTE: libc.so.1 must be linked with the minimal crti.o and crtn.o
974 # modules whose source is provided in the $(SRC)/lib/common directory.
975 # This must be done because otherwise the Sun C compiler would insert
976 # its own versions of these modules and those versions contain code
977 # to call out to C++ initialization functions.  Such C++ initialization
978 # functions can call back into libc before thread initialization is
979 # complete and this leads to segmentation violations and other problems.
980 # Since libc contains no C++ code, linking with the minimal crti.o and
981 # crtn.o modules is safe and avoids the problems described above.
982 OBJECTS= $(CRTI) $(MOSTOBJS) $(CRTN)
```



```
983 CRTSRCS= ../../common/amd64

985 # include common library definitions
986 include ../../Makefile.lib
987 include ../../Makefile.lib.64

989 CFLAGS64 += $(CTF_FLAGS)

991 # This is necessary to avoid problems with calling _ex_unwind().
992 # We probably don't want any inlining anyway.
993 CFLAGS64 += -xinline=

995 CERRWARN += -_gcc=-Wno-parentheses
996 CERRWARN += -_gcc=-Wno-switch
997 CERRWARN += -_gcc=-Wno-uninitialized
998 CERRWARN += -_gcc=-Wno-unused-value
999 CERRWARN += -_gcc=-Wno-unused-label
1000 CERRWARN += -_gcc=-Wno-unused-variable
1001 CERRWARN += -_gcc=-Wno-type-limits
1002 CERRWARN += -_gcc=-Wno-char-subscripts
1003 CERRWARN += -_gcc=-Wno-clobbered
1004 CERRWARN += -_gcc=-Wno-unused-function
1005 CERRWARN += -_gcc=-Wno-address

1007 # Setting THREAD_DEBUG = -DTHREAD_DEBUG (make THREAD_DEBUG=-DTHREAD_DEBUG ...)
1008 # enables ASSERT() checking in the threads portion of the library.
1009 # This is automatically enabled for DEBUG builds, not for non-debug builds.
1010 THREAD_DEBUG =
1011 $(NOT_RELEASE_BUILD)THREAD_DEBUG = -DTHREAD_DEBUG

1013 # Make string literals read-only to save memory
1014 CFLAGS64 += $(XSTRCONST)

1016 ALTPICS= $(TRACEOBJS:%=pics/%)

1018 $(DYNLIB) := BUILD.SO = $(LD) -o $@ -G $(DYNFLAGS) $(PICS) $(ALTPICS) $(EXTPICS)

1020 MAPFILES =      $(LIBCDIR)/port/mapfile-vers

1022 CPPFLAGS=       -D_REENTRANT -D$(MACH64) -D__$(MACH64) $(THREAD_DEBUG) \
1023                 -I. -I$(LIBCBASE)/inc -I$(LIBCDIR)/inc $(CPPFLAGS.master)
1024 ASFLAGS=        $(AS_PICFLAGS) -P -D__STDC__ -D_ASM $(CPPFLAGS) \
1025                 $(amd64_AS_XARCH)

1027 # As a favor to the dtrace syscall provider, libc still calls the
1028 # old syscall traps that have been obsoleted by the *at() interfaces.
1029 # Delete this to compile libc using only the new *at() system call traps
1030 CPPFLAGS += -D_RETAIN_OLD_SYSCALLS

1032 # proc64_id.o is built with defines in $(SRC)/uts/intel/sys/x86_archext.h
1033 pics/proc64_id.o        := CFLAGS64 += -I$(SRC)/uts/intel

1035 # Inform the run-time linker about libc specialized initialization
1036 RTLDINFO =      -z rtldinfo=tls_rtldinfo
1037 DYNFLAGS +=     $(RTLDINFO)

1039 # Force libc's internal references to be resolved immediately upon loading
1040 # in order to avoid critical region problems.  Since almost all libc symbols
1041 # are marked 'protected' in the mapfiles, this is a minimal set (15 to 20).
1042 DYNFLAGS +=     -znow

1044 BUILD.s=        $(AS) $(ASFLAGS) $< -o $@

1046 # Override this top level flag so the compiler builds in its native
1047 # C99 mode.  This has been enabled to support the complex arithmetic
1048 # added to libc.
```

```
1049 C99MODE=        $(C99_ENABLE)

1051 # libc method of building an archive
1052 # The "$(GREP) -v ' L '" part is necessary only until
1053 # lorder is fixed to ignore thread-local variables.
1054 BUILD.AR= $(RM) $@ ; \
1055         $(AR) q $@ `$(LORDER) $(MOSTOBJS:%=$(DIR)/%) | $(GREP) -v ' L ' | $(TSOR

1057 # extra files for the clean target
1058 CLEANFILES=                     \
1059         $(LIBCDIR)/port/gen/errlst.c    \
1060         $(LIBCDIR)/port/gen/new_list.c  \
1061         assym.h                 \
1062         genassym                \
1063         crt/_rtld.s             \
1064         pics/crti.o             \
1065         pics/crtn.o             \
1066         $(ALTPICS)

1068 CLOBBERFILES += $(LIB_PIC)

1070 # list of C source for lint
1071 SRCS=                                           \
1072         $(ATOMICOBJS:%.o=$(SRC)/common/atomic/%.c)      \
1073         $(XATTROBJS:%.o=$(SRC)/common/xattr/%.c)        \
1074         $(COMOBJS:%.o=$(SRC)/common/util/%.c)           \
1075         $(PORTFP:%.o=$(LIBCDIR)/port/fp/%.c)                    \
1076         $(PORTGEN:%.o=$(LIBCDIR)/port/gen/%.c)                  \
1077         $(PORTI18N:%.o=$(LIBCDIR)/port/i18n/%.c)                \
1078         $(PORTLOCALE:%.o=$(LIBCDIR)/port/locale/%.c)            \
1079         $(PORTPRINT:%.o=$(LIBCDIR)/port/print/%.c)              \
1080         $(PORTREGEX:%.o=$(LIBCDIR)/port/regex/%.c)              \
1081         $(PORTSTDIO:%.o=$(LIBCDIR)/port/stdio/%.c)              \
1082         $(PORTSYS:%.o=$(LIBCDIR)/port/sys/%.c)                  \
1083         $(AIOOBJS:%.o=$(LIBCDIR)/port/aio/%.c)                  \
1084         $(RTOBJS:%.o=$(LIBCDIR)/port/rt/%.c)                    \
1085         $(TPOOLOBJS:%.o=$(LIBCDIR)/port/tpool/%.c)              \
1086         $(THREADSOBJS:%.o=$(LIBCDIR)/port/threads/%.c)          \
1087         $(THREADSMACHOBJS:%.o=threads/%.c)              \
1088         $(UNICODEOBJS:%.o=$(SRC)/common/unicode/%.c)    \
1089         $(UNWINDMACHOBJS:%.o=unwind/%.c)                \
1090         $(FPOBJS:%.o=fp/%.c)                            \
1091         $(I386FPOBJS:%.o=$(LIBCDIR)/i386/fp/%.c)                \
1092         $(LIBCBASE)/gen/ecvt.c                          \
1093         $(LIBCBASE)/gen/makectxt.c                      \
1094         $(LIBCBASE)/gen/siginfolst.c                    \
1095         $(LIBCBASE)/gen/siglongjmp.c                    \
1096         $(LIBCBASE)/gen/sync_instruction_memory.c       \
1097         $(LIBCBASE)/sys/uadmin.c

1099 # conditional assignments
1100 # $(DYNLIB) $(LIB_PIC) := DYNOBJS = _rtbootld.o
1101 $(DYNLIB) := CRTI = crti.o
1102 $(DYNLIB) := CRTN = crtn.o

1104 # Files which need the threads .il inline template
1105 TIL=                            \
1106         aio.o                   \
1107         alloc.o                 \
1108         assfail.o               \
1109         atexit.o                \
1110         atfork.o                \
1111         cancel.o                \
1112         door_calls.o            \
1113         err.o                   \
1114         errno.o                 \
```



```
1115         lwp.o                   \
1116         ma.o                    \
1117         machdep.o               \
1118         posix_aio.o             \
1119         pthr_attr.o             \
1120         pthr_barrier.o          \
1121         pthr_cond.o             \
1122         pthr_mutex.o            \
1123         pthr_rwlock.o           \
1124         pthread.o               \
1125         rand.o                  \
1126         rwlock.o                \
1127         scalls.o                \
1128         sched.o                 \
1129         sema.o                  \
1130         sigaction.o             \
1131         sigev_thread.o          \
1132         spawn.o                 \
1133         stack.o                 \
1134         synch.o                 \
1135         tdb_agent.o             \
1136         thr.o                   \
1137         thread_interface.o      \
1138         thread_pool.o           \
1139         thrp_unwind.o           \
1140         tls.o                   \
1141         tmem.o                  \
1142         tsd.o

1144 $(TIL:%=pics/%) := CFLAGS64 += $(LIBCBASE)/threads/amd64.il

1146 # pics/mul64.o := CFLAGS64 += crt/mul64.il

1148 # large-file-aware components that should be built large

1150 #$(COMSYSOBJS64:%=pics/%) := \
1151 #       CPPFLAGS += -D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64

1153 #$(SYSOBJS64:%=pics/%) := \
1154 #       CPPFLAGS += -D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64

1156 #$(PORTGEN64:%=pics/%) := \
1157 #       CPPFLAGS += -D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64

1159 #$(PORTSTDIO64:%=pics/%) := \
1160 #       CPPFLAGS += -D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64

1162 #$(PORTSYS64:%=pics/%) := \
1163 #       CPPFLAGS += -D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64

1165 $(PORTSTDIO_W:%=pics/%) := \
1166         CPPFLAGS += -D_WIDE

1168 $(PORTPRINT_W:%=pics/%) := \
1169         CPPFLAGS += -D_WIDE

1171 $(PORTPRINT_C89:%=pics/%) := \
1172         CPPFLAGS += -D_C89_INTMAX32

1174 $(PORTSTDIO_C89:%=pics/%) := \
1175         CPPFLAGS += -D_C89_INTMAX32

1177 $(PORTI18N_COND:%=pics/%) := \
1178         CPPFLAGS += -D_WCS_LONGLONG

1180 pics/arc4random.o :=    CPPFLAGS += -I$(SRC)/common/crypto/chacha
```

```
1182 .KEEP_STATE:

1184 all: $(LIBS) $(LIB_PIC)

1186 lint    :=      CPPFLAGS += -I$(LIBCDIR)/$(MACH)/fp
1187 lint    :=      CPPFLAGS += -D_MSE_INT_H -D_LCONV_C99
1188 lint    :=      LINTFLAGS64 += -mn -erroff=E_SUPPRESSION_DIRECTIVE_UNUSED

1190 lint:
1191         @echo $(LINT.c) ... $(LDLIBS)
1192         @$(LINT.c) $(SRCS) $(LDLIBS)

1194 $(LINTLIB):= SRCS=$(LIBCDIR)/port/llib-lc
1195 $(LINTLIB):= CPPFLAGS += -D_MSE_INT_H
1196 $(LINTLIB):= LINTFLAGS64=-nvx -m64

1198 # object files that depend on inline template
1199 $(TIL:%=pics/%): $(LIBCBASE)/threads/amd64.il
1200 # pics/mul64.o: crt/mul64.il

1202 # include common libc targets
1203 include ../Makefile.targ

1205 # We need to strip out all CTF data from the static library
1206 $(LIB_PIC) := DIR = pics
1207 $(LIB_PIC): pics $$(PICS)
1208         $(BUILD.AR)
1209         $(MCS) -d -n .SUNW_ctf $@ > /dev/null 2>&1
1210         $(AR) -ts $@ > /dev/null
1211         $(POST_PROCESS_A)

1213 ASSYMDEP_OBJS=                  \
1214         _lwp_mutex_unlock.o     \
1215         _stack_grow.o           \
1216         asm_subr.o              \
1217         getcontext.o            \
1218         setjmp.o                \
1219         tls_get_addr.o          \
1220         vforkx.o

1222 $(ASSYMDEP_OBJS:%=pics/%): assym.h

1224 # assym.h build rules

1226 GENASSYM_C = genassym.c

1228 genassym: $(GENASSYM_C)
1229         $(NATIVECC) -Iinc -I$(LIBCDIR)/inc $(CPPFLAGS.native)   \
1230                 -o $@ $(GENASSYM_C)

1232 OFFSETS = $(LIBCDIR)/$(MACH)/offsets.in

1234 assym.h: $(OFFSETS) genassym
1235         $(OFFSETS_CREATE) <$(OFFSETS) >$@
1236         ./genassym >>$@

1238 # derived C source and related explicit dependencies
1239 $(LIBCDIR)/port/gen/errlst.c + \
1240 $(LIBCDIR)/port/gen/new_list.c: $(LIBCDIR)/port/gen/errlist $(LIBCDIR)/port/gen/
1241         cd $(LIBCDIR)/port/gen; pwd; $(AWK) -f errlist.awk < errlist

1243 pics/errlst.o: $(LIBCDIR)/port/gen/errlst.c

1245 pics/new_list.o: $(LIBCDIR)/port/gen/new_list.c
```

****
1327 Wed May 27 19:49:11 2015
new/usr/src/lib/libc/common/sys/brk.s
libc: adjust brk(0) to return the existing break, and use it to initialize sbrk(
****

```
 1 /*
 2  * CDDL HEADER START
 3  *
 4  * The contents of this file are subject to the terms of the
 5  * Common Development and Distribution License (the "License").
 6  * You may not use this file except in compliance with the License.
 7  *
 8  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 9  * or http://www.opensolaris.org/os/licensing.
10  * See the License for the specific language governing permissions
11  * and limitations under the License.
12  *
13  * When distributing Covered Code, include this CDDL HEADER in each
14  * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
15  * If applicable, add the following below this CDDL HEADER, with the
16  * fields enclosed by brackets "[]" replaced with your own identifying
17  * information: Portions Copyright [yyyy] [name of copyright owner]
18  *
19  * CDDL HEADER END
20  */
21 /*
22  * Copyright 2004 Sun Microsystems, Inc.  All rights reserved.
23  * Use is subject to license terms.
24  */

26         .file   "brk.s"

28 #include "SYS.h"

30 /*
31  * _brk_unlocked() simply traps into the kernel to set the brk.  It
32  * returns 0 if the break was successfully set, or -1 otherwise.
33  * It doesn't enforce any alignment and it doesn't perform any locking.
34  * _brk_unlocked() is only called from brk() and _sbrk_unlocked().
35  */

37         ENTRY_NP(_brk_unlocked)
38         SYSTRAP_RVAL1(brk)
39         SYSCERROR
40         RET
40         RETC
41         SET_SIZE(_brk_unlocked)
```

_____unchanged_portion_omitted_

```
**********************************************************
     651 Wed May 27 19:49:12 2015
new/usr/src/lib/libc/common/sys/psecflagsset.s
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
```

```
   1 /*
   2  * This file and its contents are supplied under the terms of the
   3  * Common Development and Distribution License ("CDDL"), version 1.0.
   4  * You may only use this file in accordance with the terms of version
   5  * 1.0 of the CDDL.
   6  *
   7  * A full copy of the text of the CDDL should have accompanied this
   8  * source.  A copy of the CDDL is also available via the Internet at
   9  * http://www.illumos.org/license/CDDL.
  10  */

  12 /*
  13  * Copyright 2014 <contributor>.  All rights reserved.
  14  */

  16         .file   "psecflagsset.s"

  18 #include <sys/asm_linkage.h>
  19 #include "SYS.h"
  20
  21         SYSCALL2_RVAL1(__psecflagsset,psecflags)
  22         RET
  23         SET_SIZE(__psecflagsset)
  24 #endif /* ! codereview */
```

```
**********************************************************
   24337 Wed May 27 19:49:12 2015
new/usr/src/lib/libc/i386/Makefile.com
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
#
# Copyright (c) 2004, 2010, Oracle and/or its affiliates. All rights reserved.
# Copyright (c) 2015, Joyent, Inc.  All rights reserved.
# Copyright (c) 2013, OmniTI Computer Consulting, Inc. All rights reserved.
# Copyright 2013 Garrett D'Amore <garrett@damore.org>
#
# Copyright 2011 Nexenta Systems, Inc.  All rights reserved.
# Use is subject to license terms.
#

LIBCDIR=        $(SRC)/lib/libc
LIB_PIC=        libc_pic.a
VERS=           .1
CPP=            /usr/lib/cpp
TARGET_ARCH=    i386

VALUES=         values-Xa.o

# objects are grouped by source directory

# local objects
STRETS=

CRTOBJS=                        \
        cerror.o                \
        cerror64.o

DYNOBJS=                        \
        _rtbootld.o

FPOBJS=                         \
        _D_cplx_div.o           \
        _D_cplx_div_ix.o        \
        _D_cplx_div_rx.o        \
        _D_cplx_lr_div.o        \
        _D_cplx_lr_div_ix.o     \
        _D_cplx_lr_div_rx.o     \
        _D_cplx_mul.o           \
```



```
        _F_cplx_div.o           \
        _F_cplx_div_ix.o        \
        _F_cplx_div_rx.o        \
        _F_cplx_lr_div.o        \
        _F_cplx_lr_div_ix.o     \
        _F_cplx_lr_div_rx.o     \
        _F_cplx_mul.o           \
        _X_cplx_div.o           \
        _X_cplx_div_ix.o        \
        _X_cplx_div_rx.o        \
        _X_cplx_lr_div.o        \
        _X_cplx_lr_div_ix.o     \
        _X_cplx_lr_div_rx.o     \
        _X_cplx_mul.o           \
        fpgetmask.o             \
        fpgetround.o            \
        fpgetsticky.o           \
        fpsetmask.o             \
        fpsetround.o            \
        fpsetsticky.o           \
        fpstart.o               \
        ieee.o

FPASMOBJS=                      \
        __xgetRD.o              \
        _base_il.o              \
        _xtoll.o                \
        _xtoull.o               \
        fpcw.o

ATOMICOBJS=                     \
        atomic.o

CHACHAOBJS=                     \
        chacha.o

XATTROBJS=                      \
        xattr_common.o

COMOBJS=                        \
        bcmp.o                  \
        bcopy.o                 \
        bsearch.o               \
        bzero.o                 \
        qsort.o                 \
        strtol.o                \
        strtoul.o               \
        strtoll.o               \
        strtoull.o

DTRACEOBJS=                     \
        dtrace_data.o

GENOBJS=                        \
        _div64.o                \
        _divdi3.o               \
        _getsp.o                \
        _mul64.o                \
        abs.o                   \
        alloca.o                \
        arc4random.o            \
        arc4random_uniform.o    \
        byteorder.o             \
        byteorder64.o           \
        cuexit.o                \
        ecvt.o                  \
```

```
125         errlst.o                \
126         i386_data.o             \
127         ladd.o                  \
128         ldivide.o               \
129         lmul.o                  \
130         lock.o                  \
131         lshiftl.o               \
132         lsign.o                 \
133         lsub.o                  \
134         makectxt.o              \
135         memccpy.o               \
136         memchr.o                \
137         memcmp.o                \
138         memcpy.o                \
139         memset.o                \
140         new_list.o              \
141         setjmp.o                \
142         siginfolst.o            \
143         siglongjmp.o            \
144         strcat.o                \
145         strchr.o                \
146         strcmp.o                \
147         strcpy.o                \
148         strlen.o                \
149         strncat.o               \
150         strncmp.o               \
151         strncpy.o               \
152         strnlen.o               \
153         strrchr.o               \
154         sync_instruction_memory.o

156 # sysobjs that contain large-file interfaces
157 COMSYSOBJS64=                   \
158         fstatvfs64.o            \
159         getdents64.o            \
160         getrlimit64.o           \
161         lseek64.o               \
162         mmap64.o                \
163         pread64.o               \
164         preadv64.o              \
165         pwrite64.o              \
166         pwritev64.o             \
167         setrlimit64.o           \
168         statvfs64.o

170 SYSOBJS64=

172 COMSYSOBJS=                    \
173         __clock_timer.o         \
174         __getloadavg.o          \
175         __rusagesys.o           \
176         __signotify.o           \
177         __sigrt.o               \
178         __time.o                \
179         _lgrp_home_fast.o       \
180         _lgrpsys.o              \
181         _nfssys.o               \
182         _portfs.o               \
183         _pset.o                 \
184         _rpcsys.o               \
185         _sigaction.o            \
186         _so_accept.o            \
187         _so_bind.o              \
188         _so_connect.o           \
189         _so_getpeername.o       \
190         _so_getsockname.o       \
```



```
191         _so_getsockopt.o        \
192         _so_listen.o            \
193         _so_recv.o              \
194         _so_recvfrom.o          \
195         _so_recvmsg.o           \
196         _so_send.o              \
197         _so_sendmsg.o           \
198         _so_sendto.o            \
199         _so_setsockopt.o        \
200         _so_shutdown.o          \
201         _so_socket.o            \
202         _so_socketpair.o        \
203         _sockconfig.o           \
204         acct.o                  \
205         acl.o                   \
206         adjtime.o               \
207         alarm.o                 \
208         brk.o                   \
209         chdir.o                 \
210         chroot.o                \
211         cladm.o                 \
212         close.o                 \
213         execve.o                \
214         exit.o                  \
215         facl.o                  \
216         fchdir.o                \
217         fchroot.o               \
218         fdsync.o                \
219         fpathconf.o             \
220         fstatfs.o               \
221         fstatvfs.o              \
222         getcpuid.o              \
223         getdents.o              \
224         getegid.o               \
225         geteuid.o               \
226         getgid.o                \
227         getgroups.o             \
228         gethrtime.o             \
229         getitimer.o             \
230         getmsg.o                \
231         getpid.o                \
232         getpmsg.o               \
233         getppid.o               \
234         getrandom.o             \
235         getrlimit.o             \
236         getuid.o                \
237         gtty.o                  \
238         install_utrap.o         \
239         ioctl.o                 \
240         kaio.o                  \
241         kill.o                  \
242         llseek.o                \
243         lseek.o                 \
244         mmapobjsys.o            \
245         memcntl.o               \
246         mincore.o               \
247         mmap.o                  \
248         modctl.o                \
249         mount.o                 \
250         mprotect.o              \
251         munmap.o                \
252         nice.o                  \
253         ntp_adjtime.o           \
254         ntp_gettime.o           \
255         p_online.o              \
256         pathconf.o              \
```

```
257         pause.o                 \
258         pcsample.o              \
259         pipe2.o                 \
260         pollsys.o               \
261         pread.o                 \
262         preadv.o                \
263         priocntlset.o           \
264         processor_bind.o        \
265         processor_info.o        \
266         profil.o                \
267         psecflagsset.o          \
268 #endif /* ! codereview */
269         putmsg.o                \
270         putpmsg.o               \
271         pwrite.o                \
272         pwritev.o               \
273         read.o                  \
274         readv.o                 \
275         resolvepath.o           \
276         seteguid.o              \
277         setgid.o                \
278         setgroups.o             \
279         setitimer.o             \
280         setreid.o               \
281         setrlimit.o             \
282         setuid.o                \
283         sigaltstk.o             \
284         sigprocmsk.o            \
285         sigsendset.o            \
286         sigsuspend.o            \
287         statfs.o                \
288         statvfs.o               \
289         stty.o                  \
290         sync.o                  \
291         sysconfig.o             \
292         sysfs.o                 \
293         sysinfo.o               \
294         syslwp.o                \
295         times.o                 \
296         ulimit.o                \
297         umask.o                 \
298         umount2.o               \
299         utssys.o                \
300         uucopy.o                \
301         vhangup.o               \
302         waitid.o                \
303         write.o                 \
304         writev.o                \
305         yield.o

307 SYSOBJS=                        \
308         __clock_gettime.o       \
309         __getcontext.o          \
310         __uadmin.o              \
311         _lwp_mutex_unlock.o     \
312         _stack_grow.o           \
313         door.o                  \
314         forkx.o                 \
315         forkallx.o              \
316         getcontext.o            \
317         gettimeofday.o          \
318         lwp_private.o           \
319         nuname.o                \
320         ptrace.o                \
321         syscall.o               \
322         sysi86.o                \
```



```
323         tls_get_addr.o          \
324         uadmin.o                \
325         umount.o                \
326         uname.o                 \
327         vforkx.o                \
328         xstat.o

330 # objects under $(LIBCDIR)/port which contain transitional large file interfaces
331 PORTGEN64=                      \
332         _xftw64.o               \
333         attropen64.o            \
334         ftw64.o                 \
335         mkstemp64.o             \
336         nftw64.o                \
337         tell64.o                \
338         truncate64.o

340 # objects from source under $(LIBCDIR)/port
341 PORTFP=                         \
342         __flt_decim.o           \
343         __flt_rounds.o          \
344         __tbl_10_b.o            \
345         __tbl_10_h.o            \
346         __tbl_10_s.o            \
347         __tbl_2_b.o             \
348         __tbl_2_h.o             \
349         __tbl_2_s.o             \
350         __tbl_fdq.o             \
351         __tbl_tens.o            \
352         __x_power.o             \
353         _base_sup.o             \
354         aconvert.o              \
355         decimal_bin.o           \
356         double_decim.o          \
357         econvert.o              \
358         fconvert.o              \
359         file_decim.o            \
360         finite.o                \
361         fp_data.o               \
362         func_decim.o            \
363         gconvert.o              \
364         hex_bin.o               \
365         ieee_globals.o          \
366         pack_float.o            \
367         sigfpe.o                \
368         string_decim.o

370 PORTGEN=                        \
371         _env_data.o             \
372         _xftw.o                 \
373         a64l.o                  \
374         abort.o                 \
375         addsev.o                \
376         ascii_strcasecmp.o      \
377         ascii_strncasecmp.o     \
378         assert.o                \
379         atof.o                  \
380         atoi.o                  \
381         atol.o                  \
382         atoll.o                 \
383         attrat.o                \
384         attropen.o              \
385         atexit.o                \
386         atfork.o                \
387         basename.o              \
388         calloc.o                \
```

```
389         catgets.o               \
390         catopen.o               \
391         cfgetispeed.o           \
392         cfgetospeed.o           \
393         cfree.o                 \
394         cfsetispeed.o           \
395         cfsetospeed.o           \
396         cftime.o                \
397         clock.o                 \
398         closedir.o              \
399         closefrom.o             \
400         confstr.o               \
401         crypt.o                 \
402         csetlen.o               \
403         ctime.o                 \
404         ctime_r.o               \
405         daemon.o                \
406         deflt.o                 \
407         directio.o              \
408         dirname.o               \
409         div.o                   \
410         drand48.o               \
411         dup.o                   \
412         env_data.o              \
413         err.o                   \
414         errno.o                 \
415         euclen.o                \
416         event_port.o            \
417         execvp.o                \
418         explicit_bzero.o        \
419         fattach.o               \
420         fdetach.o               \
421         fdopendir.o             \
422         ffs.o                   \
423         fls.o                   \
424         fmtmsg.o                \
425         ftime.o                 \
426         ftok.o                  \
427         ftw.o                   \
428         gcvt.o                  \
429         getauxv.o               \
430         getcwd.o                \
431         getdate_err.o           \
432         getdtblsize.o           \
433         getentropy.o            \
434         getenv.o                \
435         getexecname.o           \
436         getgrnam.o              \
437         getgrnam_r.o            \
438         gethostid.o             \
439         gethostname.o           \
440         gethz.o                 \
441         getisax.o               \
442         getloadavg.o            \
443         getlogin.o              \
444         getmntent.o             \
445         getnetgrent.o           \
446         get_nprocs.o            \
447         getopt.o                \
448         getopt_long.o           \
449         getpagesize.o           \
450         getpw.o                 \
451         getpwnam.o              \
452         getpwnam_r.o            \
453         getrusage.o             \
454         getspent.o              \
```



```
455         getspent_r.o            \
456         getsubopt.o             \
457         gettxt.o                \
458         getusershell.o          \
459         getut.o                 \
460         getutx.o                \
461         getvfsent.o             \
462         getwd.o                 \
463         getwidth.o              \
464         getxby_door.o           \
465         gtxt.o                  \
466         hsearch.o               \
467         iconv.o                 \
468         imaxabs.o               \
469         imaxdiv.o               \
470         index.o                 \
471         initgroups.o            \
472         insque.o                \
473         isaexec.o               \
474         isastream.o             \
475         isatty.o                \
476         killpg.o                \
477         klpdlib.o               \
478         l64a.o                  \
479         lckpwdf.o               \
480         lconstants.o            \
481         lexp10.o                \
482         lfind.o                 \
483         lfmt.o                  \
484         lfmt_log.o              \
485         llabs.o                 \
486         lldiv.o                 \
487         llog10.o                \
488         lltostr.o               \
489         localtime.o             \
490         lsearch.o               \
491         madvise.o               \
492         malloc.o                \
493         memalign.o              \
494         memmem.o                \
495         mkdev.o                 \
496         mkdtemp.o               \
497         mkfifo.o                \
498         mkstemp.o               \
499         mktemp.o                \
500         mlock.o                 \
501         mlockall.o              \
502         mon.o                   \
503         msync.o                 \
504         munlock.o               \
505         munlockall.o            \
506         ndbm.o                  \
507         nftw.o                  \
508         nlspath_checks.o        \
509         nsparse.o               \
510         nss_common.o            \
511         nss_dbdefs.o            \
512         nss_deffinder.o         \
513         opendir.o               \
514         opt_data.o              \
515         perror.o                \
516         pfmt.o                  \
517         pfmt_data.o             \
518         pfmt_print.o            \
519         pipe.o                  \
520         plock.o                 \
```

```
521         poll.o                  \
522         posix_fadvise.o         \
523         posix_fallocate.o       \
524         posix_madvise.o         \
525         posix_memalign.o        \
526         priocntl.o              \
527         privlib.o               \
528         priv_str_xlate.o        \
529         psecflags.o             \
530 #endif /* ! codereview */
531         psiginfo.o              \
532         psignal.o               \
533         pt.o                    \
534         putpwent.o              \
535         putspent.o              \
536         raise.o                 \
537         rand.o                  \
538         random.o                \
539         rctlops.o               \
540         readdir.o               \
541         readdir_r.o             \
542         realpath.o              \
543         reboot.o                \
544         regexpr.o               \
545         remove.o                \
546         rewinddir.o             \
547         rindex.o                \
548         scandir.o               \
549         seekdir.o               \
550         select.o                \
551         select_large_fdset.o    \
552         setlabel.o              \
553         setpriority.o           \
554         settimeofday.o          \
555         sh_locks.o              \
556         sigflag.o               \
557         siglist.o               \
558         sigsend.o               \
559         sigsetops.o             \
560         ssignal.o               \
561         stack.o                 \
562         stpcpy.o                \
563         stpncpy.o               \
564         str2sig.o               \
565         strcase_charmap.o       \
566         strchrnul.o             \
567         strcspn.o               \
568         strdup.o                \
569         strerror.o              \
570         strlcat.o               \
571         strlcpy.o               \
572         strndup.o               \
573         strpbrk.o               \
574         strsep.o                \
575         strsignal.o             \
576         strspn.o                \
577         strstr.o                \
578         strtod.o                \
579         strtoimax.o             \
580         strtok.o                \
581         strtok_r.o              \
582         strtoumax.o             \
583         swab.o                  \
584         swapctl.o               \
585         sysconf.o               \
586         syslog.o                \
```



```
587         tcdrain.o               \
588         tcflow.o                \
589         tcflush.o               \
590         tcgetattr.o             \
591         tcgetpgrp.o             \
592         tcgetsid.o              \
593         tcsendbreak.o           \
594         tcsetattr.o             \
595         tcsetpgrp.o             \
596         tell.o                  \
597         telldir.o               \
598         tfind.o                 \
599         time_data.o             \
600         time_gdata.o            \
601         tls_data.o              \
602         truncate.o              \
603         tsdalloc.o              \
604         tsearch.o               \
605         ttyname.o               \
606         ttyslot.o               \
607         ualarm.o                \
608         ucred.o                 \
609         valloc.o                \
610         vlfmt.o                 \
611         vpfmt.o                 \
612         waitpid.o               \
613         walkstack.o             \
614         wdata.o                 \
615         xgetwidth.o             \
616         xpg4.o                  \
617         xpg6.o

619 PORTPRINT_W=                    \
620         doprnt_w.o

622 PORTPRINT=                      \
623         asprintf.o              \
624         doprnt.o                \
625         fprintf.o               \
626         printf.o                \
627         snprintf.o              \
628         sprintf.o               \
629         vfprintf.o              \
630         vprintf.o               \
631         vsnprintf.o             \
632         vsprintf.o              \
633         vwprintf.o              \
634         wprintf.o

636 # c89 variants to support 32-bit size of c89 u/intmax_t (32-bit libc only)
637 PORTPRINT_C89=                  \
638         vfprintf_c89.o          \
639         vprintf_c89.o           \
640         vsnprintf_c89.o         \
641         vsprintf_c89.o          \
642         vwprintf_c89.o

644 PORTSTDIO_C89=                 \
645         vscanf_c89.o            \
646         vwscanf_c89.o           \

648 # portable stdio objects that contain large file interfaces.
649 # Note: fopen64 is a special case, as we build it small.
650 PORTSTDIO64=                    \
651         fopen64.o               \
652         fpos64.o
```

```
654 PORTSTDIO_W=                    \
655         doscan_w.o

657 PORTSTDIO=                      \
658         __extensions.o          \
659         _endopen.o              \
660         _filbuf.o               \
661         _findbuf.o              \
662         _flsbuf.o               \
663         _wrtchk.o               \
664         clearerr.o              \
665         ctermid.o               \
666         ctermid_r.o             \
667         cuserid.o               \
668         data.o                  \
669         doscan.o                \
670         fdopen.o                \
671         feof.o                  \
672         ferror.o                \
673         fgetc.o                 \
674         fgets.o                 \
675         fileno.o                \
676         flockf.o                \
677         flush.o                 \
678         fopen.o                 \
679         fpos.o                  \
680         fputc.o                 \
681         fputs.o                 \
682         fread.o                 \
683         fseek.o                 \
684         fseeko.o                \
685         ftell.o                 \
686         ftello.o                \
687         fwrite.o                \
688         getc.o                  \
689         getchar.o               \
690         getline.o               \
691         getpass.o               \
692         gets.o                  \
693         getw.o                  \
694         mse.o                   \
695         popen.o                 \
696         putc.o                  \
697         putchar.o               \
698         puts.o                  \
699         putw.o                  \
700         rewind.o                \
701         scanf.o                 \
702         setbuf.o                \
703         setbuffer.o             \
704         setvbuf.o               \
705         system.o                \
706         tempnam.o               \
707         tmpfile.o               \
708         tmpnam_r.o              \
709         ungetc.o                \
710         vscanf.o                \
711         vwscanf.o               \
712         wscanf.o

714 PORTI18N=                       \
715         getwchar.o              \
716         putwchar.o              \
717         putws.o                 \
718         strtows.o               \
```



```
719         wcsnlen.o               \
720         wcsstr.o                \
721         wcstoimax.o             \
722         wcstol.o                \
723         wcstoul.o               \
724         wcswcs.o                \
725         wmemchr.o               \
726         wmemcmp.o               \
727         wmemcpy.o               \
728         wmemmove.o              \
729         wmemset.o               \
730         wscat.o                 \
731         wschr.o                 \
732         wscmp.o                 \
733         wscpy.o                 \
734         wscspn.o                \
735         wsdup.o                 \
736         wslen.o                 \
737         wsncat.o                \
738         wsncmp.o                \
739         wsncpy.o                \
740         wspbrk.o                \
741         wsprintf.o              \
742         wsrchr.o                \
743         wsscanf.o               \
744         wsspn.o                 \
745         wstod.o                 \
746         wstok.o                 \
747         wstol.o                 \
748         wstoll.o                \
749         wsxfrm.o                \
750         gettext.o               \
751         gettext_gnu.o           \
752         gettext_real.o          \
753         gettext_util.o          \
754         plural_parser.o         \
755         wdresolve.o             \
756         _ctype.o                \
757         isascii.o               \
758         toascii.o

760 PORTI18N_COND=                  \
761         wcstol_longlong.o       \
762         wcstoul_longlong.o

764 PORTLOCALE=                     \
765         big5.o                  \
766         btowc.o                 \
767         collate.o               \
768         collcmp.o               \
769         euc.o                   \
770         fnmatch.o               \
771         fgetwc.o                \
772         fgetws.o                \
773         fix_grouping.o          \
774         fputwc.o                \
775         fputws.o                \
776         fwide.o                 \
777         gb18030.o               \
778         gb2312.o                \
779         gbk.o                   \
780         getdate.o               \
781         isdigit.o               \
782         iswctype.o              \
783         ldpart.o                \
784         lmessages.o             \
```

```
785         lnumeric.o              \
786         lmonetary.o             \
787         localeconv.o            \
788         localeimpl.o            \
789         mbftowc.o               \
790         mblen.o                 \
791         mbrlen.o                \
792         mbrtowc.o               \
793         mbsinit.o               \
794         mbsnrtowcs.o            \
795         mbsrtowcs.o             \
796         mbstowcs.o              \
797         mbtowc.o                \
798         mskanji.o               \
799         nextwctype.o            \
800         nl_langinfo.o           \
801         none.o                  \
802         regcomp.o               \
803         regfree.o               \
804         regerror.o              \
805         regexec.o               \
806         rune.o                  \
807         runetype.o              \
808         setlocale.o             \
809         setrunelocale.o         \
810         strcasecmp.o            \
811         strcasestr.o            \
812         strcoll.o               \
813         strfmon.o               \
814         strftime.o              \
815         strncasecmp.o           \
816         strptime.o              \
817         strxfrm.o               \
818         table.o                 \
819         timelocal.o             \
820         tolower.o               \
821         towlower.o              \
822         ungetwc.o               \
823         utf8.o                  \
824         wcrtomb.o               \
825         wcscasecmp.o            \
826         wcscoll.o               \
827         wcsftime.o              \
828         wcsnrtombs.o            \
829         wcsrtombs.o             \
830         wcswidth.o              \
831         wcstombs.o              \
832         wcsxfrm.o               \
833         wctob.o                 \
834         wctomb.o                \
835         wctrans.o               \
836         wctype.o                \
837         wcwidth.o               \
838         wscol.o

840 AIOOBJS=                       \
841         aio.o                   \
842         aio_alloc.o             \
843         posix_aio.o

845 RTOBJS=                        \
846         clock_timer.o           \
847         mqueue.o                \
848         pos4obj.o               \
849         sched.o                 \
850         sem.o                   \
```



```
851         shm.o                   \
852         sigev_thread.o

854 TPOOLOBJS=                     \
855         thread_pool.o

857 THREADSOBJS=                   \
858         alloc.o                 \
859         assfail.o               \
860         cancel.o                \
861         door_calls.o            \
862         tmem.o                  \
863         pthr_attr.o             \
864         pthr_barrier.o          \
865         pthr_cond.o             \
866         pthr_mutex.o            \
867         pthr_rwlock.o           \
868         pthread.o               \
869         rwlock.o                \
870         scalls.o                \
871         sema.o                  \
872         sigaction.o             \
873         spawn.o                 \
874         synch.o                 \
875         tdb_agent.o             \
876         thr.o                   \
877         thread_interface.o      \
878         tls.o                   \
879         tsd.o

881 THREADSMACHOBJS=               \
882         machdep.o

884 THREADSASMOBJS=                \
885         asm_subr.o

887 UNICODEOBJS=                   \
888         u8_textprep.o           \
889         uconv.o

891 UNWINDMACHOBJS=                \
892         unwind.o

894 UNWINDASMOBJS=                 \
895         unwind_frame.o

897 # objects that implement the transitional large file API
898 PORTSYS64=                     \
899         lockf64.o               \
900         stat64.o

902 PORTSYS=                       \
903         _autofssys.o            \
904         access.o                \
905         acctctl.o               \
906         bsd_signal.o            \
907         chmod.o                 \
908         chown.o                 \
909         corectl.o               \
910         exacctsys.o             \
911         execl.o                 \
912         execle.o                \
913         execv.o                 \
914         fcntl.o                 \
915         getpagesizes.o          \
916         getpeerucred.o          \
```

```
917         inst_sync.o             \
918         issetugid.o             \
919         label.o                 \
920         link.o                  \
921         lockf.o                 \
922         lwp.o                   \
923         lwp_cond.o              \
924         lwp_rwlock.o            \
925         lwp_sigmask.o           \
926         meminfosys.o            \
927         mkdir.o                 \
928         mknod.o                 \
929         msgsys.o                \
930         nfssys.o                \
931         open.o                  \
932         pgrpsys.o               \
933         posix_sigwait.o         \
934         ppriv.o                 \
935         psetsys.o               \
936         rctlsys.o               \
937         readlink.o              \
938         rename.o                \
939         sbrk.o                  \
940         semsys.o                \
941         set_errno.o             \
942         sharefs.o               \
943         shmsys.o                \
944         sidsys.o                \
945         siginterrupt.o          \
946         signal.o                \
947         sigpending.o            \
948         sigstack.o              \
949         stat.o                  \
950         symlink.o               \
951         tasksys.o               \
952         time.o                  \
953         time_util.o             \
954         ucontext.o              \
955         unlink.o                \
956         ustat.o                 \
957         utimesys.o              \
958         zone.o

960 PORTREGEX=                       \
961         glob.o                  \
962         regcmp.o                \
963         regex.o                 \
964         wordexp.o

966 MOSTOBJS=                        \
967         $(STRETS)               \
968         $(CRTOBJS)              \
969         $(DYNOBJS)              \
970         $(FPOBJS)               \
971         $(FPASMOBJS)            \
972         $(ATOMICOBJS)           \
973         $(CHACHAOBJS)           \
974         $(XATTROBJS)            \
975         $(COMOBJS)              \
976         $(DTRACEOBJS)           \
977         $(GENOBJS)              \
978         $(PORTFP)               \
979         $(PORTGEN)              \
980         $(PORTGEN64)            \
981         $(PORTI18N)             \
982         $(PORTI18N_COND)        \
```



```
983         $(PORTLOCALE)           \
984         $(PORTPRINT)            \
985         $(PORTPRINT_C89)        \
986         $(PORTPRINT_W)          \
987         $(PORTREGEX)            \
988         $(PORTSTDIO)            \
989         $(PORTSTDIO64)          \
990         $(PORTSTDIO_C89)        \
991         $(PORTSTDIO_W)          \
992         $(PORTSYS)              \
993         $(PORTSYS64)            \
994         $(AIOOBJS)              \
995         $(RTOBJS)               \
996         $(TPOOLOBJS)            \
997         $(THREADSOBJS)          \
998         $(THREADSMACHOBJS)      \
999         $(THREADSASMOBJS)       \
1000        $(UNICODEOBJS)          \
1001        $(UNWINDMACHOBJS)       \
1002        $(UNWINDASMOBJS)        \
1003        $(COMSYSOBJS)           \
1004        $(SYSOBJS)              \
1005        $(COMSYSOBJS64)         \
1006        $(SYSOBJS64)            \
1007        $(VALUES)

1009 TRACEOBJS=                     \
1010        plockstat.o

1012 # NOTE: libc.so.1 must be linked with the minimal crti.o and crtn.o
1013 # modules whose source is provided in the $(SRC)/lib/common directory.
1014 # This must be done because otherwise the Sun C compiler would insert
1015 # its own versions of these modules and those versions contain code
1016 # to call out to C++ initialization functions.  Such C++ initialization
1017 # functions can call back into libc before thread initialization is
1018 # complete and this leads to segmentation violations and other problems.
1019 # Since libc contains no C++ code, linking with the minimal crti.o and
1020 # crtn.o modules is safe and avoids the problems described above.
1021 OBJECTS= $(CRTI) $(MOSTOBJS) $(CRTN)
1022 CRTSRCS= ../../common/i386

1024 LDPASS_OFF=     $(POUND_SIGN)

1026 # include common library definitions
1027 include ../../Makefile.lib

1029 # we need to override the default SONAME here because we might
1030 # be building a variant object (still libc.so.1, but different filename)
1031 SONAME = libc.so.1

1033 CFLAGS += $(CCVERBOSE) $(CTF_FLAGS)

1035 # This is necessary to avoid problems with calling _ex_unwind().
1036 # We probably don't want any inlining anyway.
1037 XINLINE = -xinline=
1038 CFLAGS += $(XINLINE)

1040 CERRWARN += -_gcc=-Wno-parentheses
1041 CERRWARN += -_gcc=-Wno-switch
1042 CERRWARN += -_gcc=-Wno-uninitialized
1043 CERRWARN += -_gcc=-Wno-unused-value
1044 CERRWARN += -_gcc=-Wno-unused-label
1045 CERRWARN += -_gcc=-Wno-unused-variable
1046 CERRWARN += -_gcc=-Wno-type-limits
1047 CERRWARN += -_gcc=-Wno-char-subscripts
1048 CERRWARN += -_gcc=-Wno-clobbered
```

```
1049 CERRWARN += -_gcc=-Wno-unused-function
1050 CERRWARN += -_gcc=-Wno-address

1052 # Setting THREAD_DEBUG = -DTHREAD_DEBUG (make THREAD_DEBUG=-DTHREAD_DEBUG ...)
1053 # enables ASSERT() checking in the threads portion of the library.
1054 # This is automatically enabled for DEBUG builds, not for non-debug builds.
1055 THREAD_DEBUG =
1056 $(NOT_RELEASE_BUILD)THREAD_DEBUG = -DTHREAD_DEBUG

1058 # Make string literals read-only to save memory.
1059 CFLAGS += $(XSTRCONST)

1061 ALTPICS= $(TRACEOBJS:%=pics/%)

1063 $(DYNLIB) := BUILD.SO = $(LD) -o $@ -G $(DYNFLAGS) $(PICS) $(ALTPICS) \
1064                 $(EXTPICS) $(LDLIBS)

1066 MAPFILES =      $(LIBCDIR)/port/mapfile-vers

1068 #
1069 # EXTN_CPPFLAGS and EXTN_CFLAGS set in enclosing Makefile
1070 #
1071 CFLAGS +=       $(EXTN_CFLAGS)
1072 CPPFLAGS=       -D_REENTRANT -Di386 $(EXTN_CPPFLAGS) $(THREAD_DEBUG) \
1073                 -I$(LIBCBASE)/inc -I$(LIBCDIR)/inc $(CPPFLAGS.master)
1074 ASFLAGS=        $(AS_PICFLAGS) -P -D__STDC__ -D_ASM $(CPPFLAGS) $(i386_AS_XARCH)

1076 # As a favor to the dtrace syscall provider, libc still calls the
1077 # old syscall traps that have been obsoleted by the *at() interfaces.
1078 # Delete this to compile libc using only the new *at() system call traps
1079 CPPFLAGS += -D_RETAIN_OLD_SYSCALLS

1081 # Inform the run-time linker about libc specialized initialization
1082 RTLDINFO =      -z rtldinfo=tls_rtldinfo
1083 DYNFLAGS +=     $(RTLDINFO)

1085 # Force libc's internal references to be resolved immediately upon loading
1086 # in order to avoid critical region problems.  Since almost all libc symbols
1087 # are marked 'protected' in the mapfiles, this is a minimal set (15 to 20).
1088 DYNFLAGS +=     -znow

1090 DYNFLAGS +=     -e __rtboot
1091 DYNFLAGS +=     $(EXTN_DYNFLAGS)

1093 # Inform the kernel about the initial DTrace area (in case
1094 # libc is being used as the interpreter / runtime linker).
1095 DTRACE_DATA =   -zdtrace=dtrace_data
1096 DYNFLAGS +=     $(DTRACE_DATA)

1098 # DTrace needs an executable data segment.
1099 MAPFILE.NED=

1101 BUILD.s=        $(AS) $(ASFLAGS) $< -o $@

1103 # Override this top level flag so the compiler builds in its native
1104 # C99 mode.  This has been enabled to support the complex arithmetic
1105 # added to libc.
1106 C99MODE=        $(C99_ENABLE)

1108 # libc method of building an archive
1109 # The "$(GREP) -v ' L '" part is necessary only until
1110 # lorder is fixed to ignore thread-local variables.
1111 BUILD.AR= $(RM) $@ ; \
1112         $(AR) q $@ '$(LORDER) $(MOSTOBJS:%=$(DIR)/%) | $(GREP) -v ' L ' | $(TSOR

1114 # extra files for the clean target
```



```
1115 CLEANFILES=                    \
1116         $(LIBCDIR)/port/gen/errlst.c    \
1117         $(LIBCDIR)/port/gen/new_list.c  \
1118         assym.h                 \
1119         genassym                \
1120         crt/_rtld.s             \
1121         crt/_rtbootld.s         \
1122         pics/_rtbootld.o        \
1123         pics/crti.o             \
1124         pics/crtn.o             \
1125         $(ALTPICS)

1127 CLOBBERFILES += $(LIB_PIC)

1129 # list of C source for lint
1130 SRCS=                                                   \
1131         $(ATOMICOBJS:%.o=$(SRC)/common/atomic/%.c)      \
1132         $(XATTROBJS:%.o=$(SRC)/common/xattr/%.c)        \
1133         $(COMOBJS:%.o=$(SRC)/common/util/%.c)           \
1134         $(DTRACEOBJS:%.o=$(SRC)/common/dtrace/%.c)      \
1135         $(PORTFP:%.o=$(LIBCDIR)/port/fp/%.c)                    \
1136         $(PORTGEN:%.o=$(LIBCDIR)/port/gen/%.c)                  \
1137         $(PORTI18N:%.o=$(LIBCDIR)/port/i18n/%.c)                \
1138         $(PORTLOCALE:%.o=$(LIBCDIR)/port/locale/%.c)            \
1139         $(PORTPRINT:%.o=$(LIBCDIR)/port/print/%.c)              \
1140         $(PORTREGEX:%.o=$(LIBCDIR)/port/regex/%.c)              \
1141         $(PORTSTDIO:%.o=$(LIBCDIR)/port/stdio/%.c)              \
1142         $(PORTSYS:%.o=$(LIBCDIR)/port/sys/%.c)                  \
1143         $(AIOOBJS:%.o=$(LIBCDIR)/port/aio/%.c)                  \
1144         $(RTOBJS:%.o=$(LIBCDIR)/port/rt/%.c)                    \
1145         $(TPOOLOBJS:%.o=$(LIBCDIR)/port/tpool/%.c)              \
1146         $(THREADSOBJS:%.o=$(LIBCDIR)/port/threads/%.c)          \
1147         $(THREADSMACHOBJS:%.o=$(LIBCDIR)/$(MACH)/threads/%.c)   \
1148         $(UNICODEOBJS:%.o=$(SRC)/common/unicode/%.c)    \
1149         $(UNWINDMACHOBJS:%.o=$(LIBCDIR)/port/unwind/%.c)        \
1150         $(FPOBJS:%.o=$(LIBCDIR)/$(MACH)/fp/%.c)                 \
1151         $(LIBCBASE)/gen/ecvt.c                          \
1152         $(LIBCBASE)/gen/makectxt.c                      \
1153         $(LIBCBASE)/gen/siginfolst.c                    \
1154         $(LIBCBASE)/gen/siglongjmp.c                    \
1155         $(LIBCBASE)/gen/strcmp.c                        \
1156         $(LIBCBASE)/gen/sync_instruction_memory.c       \
1157         $(LIBCBASE)/sys/ptrace.c                        \
1158         $(LIBCBASE)/sys/uadmin.c

1160 # conditional assignments
1161 $(DYNLIB) := CRTI = crti.o
1162 $(DYNLIB) := CRTN = crtn.o

1164 # Files which need the threads .il inline template
1165 TIL=                            \
1166         aio.o                   \
1167         alloc.o                 \
1168         assfail.o               \
1169         atexit.o                \
1170         atfork.o                \
1171         cancel.o                \
1172         door_calls.o            \
1173         err.o                   \
1174         errno.o                 \
1175         lwp.o                   \
1176         ma.o                    \
1177         machdep.o               \
1178         posix_aio.o             \
1179         pthr_attr.o             \
1180         pthr_barrier.o          \
```

```
1181         pthr_cond.o             \
1182         pthr_mutex.o            \
1183         pthr_rwlock.o           \
1184         pthread.o               \
1185         rand.o                  \
1186         rwlock.o                \
1187         scalls.o                \
1188         sched.o                 \
1189         sema.o                  \
1190         sigaction.o             \
1191         sigev_thread.o          \
1192         spawn.o                 \
1193         stack.o                 \
1194         synch.o                 \
1195         tdb_agent.o             \
1196         thr.o                   \
1197         thread_interface.o      \
1198         thread_pool.o           \
1199         tls.o                   \
1200         tsd.o                   \
1201         tmem.o                  \
1202         unwind.o

1204 THREADS_INLINES = $(LIBCBASE)/threads/i386.il
1205 $(TIL:%=pics/%) := CFLAGS += $(THREADS_INLINES)

1207 # pics/mul64.o := CFLAGS += $(LIBCBASE)/crt/mul64.il

1209 # large-file-aware components that should be built large

1211 $(COMSYSOBJS64:%=pics/%) := \
1212         CPPFLAGS += -D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64

1214 $(SYSOBJS64:%=pics/%) := \
1215         CPPFLAGS += -D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64

1217 $(PORTGEN64:%=pics/%) := \
1218         CPPFLAGS += -D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64

1220 $(PORTSTDIO64:%=pics/%) := \
1221         CPPFLAGS += -D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64

1223 $(PORTSYS64:%=pics/%) := \
1224         CPPFLAGS += -D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64

1226 $(PORTSTDIO_W:%=pics/%) := \
1227         CPPFLAGS += -D_WIDE

1229 $(PORTPRINT_W:%=pics/%) := \
1230         CPPFLAGS += -D_WIDE

1232 $(PORTPRINT_C89:%=pics/%) := \
1233         CPPFLAGS += -D_C89_INTMAX32

1235 $(PORTSTDIO_C89:%=pics/%) := \
1236         CPPFLAGS += -D_C89_INTMAX32

1238 $(PORTI18N_COND:%=pics/%) := \
1239         CPPFLAGS += -D_WCS_LONGLONG

1241 pics/arc4random.o :=    CPPFLAGS += -I$(SRC)/common/crypto/chacha

1243 .KEEP_STATE:

1245 all: $(LIBS) $(LIB_PIC)
```



```
1247 lint    :=      CPPFLAGS += -I$(LIBCDIR)/$(MACH)/fp
1248 lint    :=      CPPFLAGS += -D_MSE_INT_H -D_LCONV_C99
1249 lint    :=      LINTFLAGS += -mn -erroff=E_SUPPRESSION_DIRECTIVE_UNUSED

1251 lint:
1252         @echo $(LINT.c) ...
1253         @$(LINT.c) $(SRCS) $(LDLIBS)

1255 $(LINTLIB):= SRCS=$(LIBCDIR)/port/llib-lc
1256 $(LINTLIB):= CPPFLAGS += -D_MSE_INT_H
1257 $(LINTLIB):= LINTFLAGS=-nvx

1259 # object files that depend on inline template
1260 $(TIL:%=pics/%): $(LIBCBASE)/threads/i386.il
1261 # pics/mul64.o: $(LIBCBASE)/crt/mul64.il

1263 # include common libc targets
1264 include $(LIBCDIR)/Makefile.targ

1266 # We need to strip out all CTF and DOF data from the static library
1267 $(LIB_PIC) := DIR = pics
1268 $(LIB_PIC): pics $$(PICS)
1269         $(BUILD.AR)
1270         $(MCS) -d -n .SUNW_ctf $@ > /dev/null 2>&1
1271         $(MCS) -d -n .SUNW_dof $@ > /dev/null 2>&1
1272         $(AR) -ts $@ > /dev/null
1273         $(POST_PROCESS_A)

1275 $(LIBCBASE)/crt/_rtbootld.s: $(LIBCBASE)/crt/_rtboot.s $(LIBCBASE)/crt/_rtld.c
1276         $(CC) $(CPPFLAGS) $(CTF_FLAGS) -O -S $(C_PICFLAGS) \
1277             $(LIBCBASE)/crt/_rtld.c -o $(LIBCBASE)/crt/_rtld.s
1278         $(CAT) $(LIBCBASE)/crt/_rtboot.s $(LIBCBASE)/crt/_rtld.s > $@
1279         $(RM) $(LIBCBASE)/crt/_rtld.s

1281 # partially built from C source
1282 pics/_rtbootld.o: $(LIBCBASE)/crt/_rtbootld.s
1283         $(AS) $(ASFLAGS) $(LIBCBASE)/crt/_rtbootld.s -o $@
1284         $(CTFCONVERT_O)

1286 ASSYMDEP_OBJS=                  \
1287         _lwp_mutex_unlock.o     \
1288         _stack_grow.o           \
1289         getcontext.o            \
1290         setjmp.o                \
1291         tls_get_addr.o          \
1292         vforkx.o

1294 $(ASSYMDEP_OBJS:%=pics/%)       :=      CPPFLAGS += -I.

1296 $(ASSYMDEP_OBJS:%=pics/%): assym.h

1298 # assym.h build rules

1300 GENASSYM_C = $(LIBCDIR)/$(MACH)/genassym.c

1302 genassym: $(GENASSYM_C)
1303         $(NATIVECC) -I$(LIBCBASE)/inc -I$(LIBCDIR)/inc  \
1304                 -D__EXTENSIONS__ $(CPPFLAGS.native) -o $@ $(GENASSYM_C)

1306 OFFSETS = $(LIBCDIR)/$(MACH)/offsets.in

1308 assym.h: $(OFFSETS) genassym
1309         $(OFFSETS_CREATE) <$(OFFSETS) >$@
1310         ./genassym >>$@

1312 # derived C source and related explicit dependencies
```

```
1313 $(LIBCDIR)/port/gen/errlst.c + \
1314 $(LIBCDIR)/port/gen/new_list.c: $(LIBCDIR)/port/gen/errlist $(LIBCDIR)/port/gen/
1315         cd $(LIBCDIR)/port/gen; pwd; $(AWK) -f errlist.awk < errlist

1317 pics/errlst.o: $(LIBCDIR)/port/gen/errlst.c

1319 pics/new_list.o: $(LIBCDIR)/port/gen/new_list.c
```

```
**********************************************************
     803 Wed May 27 19:49:13 2015
new/usr/src/lib/libc/port/gen/psecflags.c
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
```

```
   1 /*
   2  * This file and its contents are supplied under the terms of the
   3  * Common Development and Distribution License ("CDDL"), version 1.0.
   4  * You may only use this file in accordance with the terms of version
   5  * 1.0 of the CDDL.
   6  *
   7  * A full copy of the text of the CDDL should have accompanied this
   8  * source.  A copy of the CDDL is also available via the Internet at
   9  * http://www.illumos.org/license/CDDL.
  10  */

  12 /*
  13  * Copyright 2014 <contributor>.  All rights reserved.
  14  */

  16 #include <sys/proc.h>
  17 #include <sys/procset.h>
  18 #include <sys/syscall.h>

  20 extern int __psecflagsset(procset_t *, int, uint_t);

  22 int
  23 psecflags(idtype_t idtype, id_t id, psecflags_cmd_t cmd, uint_t arg)
  24 {
  25         procset_t procset;

  27         setprocset(&procset, POP_AND, idtype, id, P_ALL, 0);

  29         return (__psecflagsset(&procset, cmd, arg));
  30 }
  31 #endif /* ! codereview */
```

```
**********************************************************
   56819 Wed May 27 19:49:13 2015
new/usr/src/lib/libc/port/mapfile-vers
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
______unchanged_portion_omitted_



# There should never be more than one SUNWprivate version.
# Don't add any more.  Add new private symbols to SUNWprivate_1.1

SYMBOL_VERSION SUNWprivate_1.1 {
    global:
        ___Argv                 { FLAGS = NODIRECT };
        cfree                   { FLAGS = NODIRECT };
        _cswidth;
        __ctype_mask;
        __environ_lock          { FLAGS = NODIRECT };
        __inf_read;
        __inf_written;
        __i_size;
        _isnanf                 { TYPE = FUNCTION; FILTER = libm.so.2 };
        __iswrune;
        __libc_threaded;
        _lib_version            { FLAGS = NODIRECT };
        _logb                   { TYPE = FUNCTION; FILTER = libm.so.2 };
        _lone                   { FLAGS = NODYNSORT };
        _lten                   { FLAGS = NODYNSORT };
        _lzero                  { FLAGS = NODYNSORT };
        __malloc_lock;
        _memcmp;
        _memcpy                 { FLAGS = NODYNSORT };
        _memmove;
        _memset;
        _modff                  { TYPE = FUNCTION; FILTER = libm.so.2 };
        __nan_read;
        __nan_written;
        __nextwctype;
        __nis_debug_bind;
        __nis_debug_calls;
        __nis_debug_file;
        __nis_debug_rpc;
        __nis_prefsrv;
        __nis_preftype;
        __nis_server;
        _nss_default_finders;
        __progname              { FLAGS = NODIRECT };
        _smbuf;
        _sp;
        __strdupa_str           { FLAGS = NODIRECT };
        __strdupa_len           { FLAGS = NODIRECT };
        _tdb_bootstrap;
        __threaded;
        thr_probe_getfunc_addr;
        __trans_lower;
        __trans_upper;
        _uberdata;
        __xpg6                  { FLAGS = NODIRECT };

$if _ELF32
        _dladdr                 { TYPE = FUNCTION; FILTER = /usr/lib/ld.so.1 };
        _dladdr1                { TYPE = FUNCTION; FILTER = /usr/lib/ld.so.1 };
```



```
        _dlclose                { TYPE = FUNCTION; FILTER = /usr/lib/ld.so.1 };
        _dldump                 { TYPE = FUNCTION; FILTER = /usr/lib/ld.so.1 };
        _dlerror                { TYPE = FUNCTION; FILTER = /usr/lib/ld.so.1 };
        _dlinfo                 { TYPE = FUNCTION; FILTER = /usr/lib/ld.so.1 };
        _dlmopen                { TYPE = FUNCTION; FILTER = /usr/lib/ld.so.1 };
        _dlopen                 { TYPE = FUNCTION; FILTER = /usr/lib/ld.so.1 };
        _dlsym                  { TYPE = FUNCTION; FILTER = /usr/lib/ld.so.1 };
        _ld_libc                { TYPE = FUNCTION; FILTER = /usr/lib/ld.so.1 };
        _sys_errlist;
        _sys_errs;
        _sys_index;
        _sys_nerr               { FLAGS = NODYNSORT };
        _sys_num_err;
$elif sparcv9
        _dladdr         { TYPE = FUNCTION; FILTER = /usr/lib/sparcv9/ld.so.1 };
        _dladdr1        { TYPE = FUNCTION; FILTER = /usr/lib/sparcv9/ld.so.1 };
        _dlclose        { TYPE = FUNCTION; FILTER = /usr/lib/sparcv9/ld.so.1 };
        _dldump         { TYPE = FUNCTION; FILTER = /usr/lib/sparcv9/ld.so.1 };
        _dlerror        { TYPE = FUNCTION; FILTER = /usr/lib/sparcv9/ld.so.1 };
        _dlinfo         { TYPE = FUNCTION; FILTER = /usr/lib/sparcv9/ld.so.1 };
        _dlmopen        { TYPE = FUNCTION; FILTER = /usr/lib/sparcv9/ld.so.1 };
        _dlopen         { TYPE = FUNCTION; FILTER = /usr/lib/sparcv9/ld.so.1 };
        _dlsym          { TYPE = FUNCTION; FILTER = /usr/lib/sparcv9/ld.so.1 };
        _ld_libc        { TYPE = FUNCTION; FILTER = /usr/lib/sparcv9/ld.so.1 };
$elif amd64
        _dladdr         { TYPE = FUNCTION; FILTER = /usr/lib/amd64/ld.so.1 };
        _dladdr1        { TYPE = FUNCTION; FILTER = /usr/lib/amd64/ld.so.1 };
        _dlamd64getunwind { TYPE = FUNCTION; FILTER = /usr/lib/amd64/ld.so.1 };
        _dlclose        { TYPE = FUNCTION; FILTER = /usr/lib/amd64/ld.so.1 };
        _dldump         { TYPE = FUNCTION; FILTER = /usr/lib/amd64/ld.so.1 };
        _dlerror        { TYPE = FUNCTION; FILTER = /usr/lib/amd64/ld.so.1 };
        _dlinfo         { TYPE = FUNCTION; FILTER = /usr/lib/amd64/ld.so.1 };
        _dlmopen        { TYPE = FUNCTION; FILTER = /usr/lib/amd64/ld.so.1 };
        _dlopen         { TYPE = FUNCTION; FILTER = /usr/lib/amd64/ld.so.1 };
        _dlsym          { TYPE = FUNCTION; FILTER = /usr/lib/amd64/ld.so.1 };
        _ld_libc        { TYPE = FUNCTION; FILTER = /usr/lib/amd64/ld.so.1 };
$else
$error unknown platform
$endif

$if _sparc
        __lyday_to_month;
        __mon_lengths;
        __yday_to_month;
$endif
$if i386
        _sse_hw;
$endif

    protected:
        acctctl;
        allocids;
        _assert_c99;
        __assert_c99;
        _assfail;
        attr_count;
        attr_to_data_type;
        attr_to_name;
        attr_to_option;
        attr_to_xattr_view;
        _autofssys;
        _bufsync;
        _cladm;
        __class_quadruple;
        core_get_default_content;
        core_get_default_path;
```

```
2807        core_get_global_content;
2808        core_get_global_path;
2809        core_get_options;
2810        core_get_process_content;
2811        core_get_process_path;
2812        core_set_default_content;
2813        core_set_default_path;
2814        core_set_global_content;
2815        core_set_global_path;
2816        core_set_options;
2817        core_set_process_content;
2818        core_set_process_path;
2819        dbm_close_status;
2820        dbm_do_nextkey;
2821        dbm_setdefwrite;
2822        _D_cplx_div;
2823        _D_cplx_div_ix;
2824        _D_cplx_div_rx;
2825        _D_cplx_mul;
2826        defclose_r;
2827        defcntl;
2828        defcntl_r;
2829        defopen;
2830        defopen_r;
2831        defread;
2832        defread_r;
2833        _delete;
2834        _dgettext;
2835        _doprnt;
2836        _doscan;
2837        _errfp;
2838        _errxfp;
2839        exportfs;
2840        _F_cplx_div;
2841        _F_cplx_div_ix;
2842        _F_cplx_div_rx;
2843        _F_cplx_mul;
2844        __fgetwc_xpg5;
2845        __fgetws_xpg5;
2846        _findbuf;
2847        _findiop;
2848        __fini_daemon_priv;
2849        _finite;
2850        _fork1                  { FLAGS = NODYNSORT };
2851        _forkall                { FLAGS = NODYNSORT };
2852        _fpclass;
2853        _fpgetmask;
2854        _fpgetround;
2855        _fpgetsticky;
2856        _fprintf;
2857        _fpsetmask;
2858        _fpsetround;
2859        _fpsetsticky;
2860        __fputwc_xpg5;
2861        __fputws_xpg5;
2862        _ftw;
2863        _gcvt;
2864        _getarg;
2865        __getcontext;
2866        _getdents;
2867        _get_exit_frame_monitor;
2868        _getfp;
2869        _getgroupsbymember;
2870        _getlogin_r;
2871        getrandom;
2872        _getsp;
```



```
2873        __gettsp;
2874        getvmusage;
2875        __getwchar_xpg5;
2876        __getwc_xpg5;
2877        gtty;
2878        __idmap_flush_kcache;
2879        __idmap_reg;
2880        __idmap_unreg;
2881        __init_daemon_priv;
2882        __init_suid_priv;
2883        _insert;
2884        inst_sync;
2885        _iswctype;
2886        klpd_create;
2887        klpd_getpath;
2888        klpd_getport;
2889        klpd_getucred;
2890        klpd_register;
2891        klpd_register_id;
2892        klpd_unregister;
2893        klpd_unregister_id;
2894        _lgrp_home_fast         { FLAGS = NODYNSORT };
2895        _lgrpsys;
2896        _lltostr;
2897        _lock_clear;
2898        _lock_try;
2899        _ltzset;
2900        lwp_self;
2901        makeut;
2902        makeutx;
2903        _mbftowc;
2904        mcfiller;
2905        mntopt;
2906        modctl;
2907        modutx;
2908        msgctl64;
2909        __multi_innetgr;
2910        _mutex_destroy          { FLAGS = NODYNSORT };
2911        mutex_held;
2912        _mutex_init             { FLAGS = NODYNSORT };
2913        _mutex_unlock           { FLAGS = NODYNSORT };
2914        name_to_attr;
2915        nfs_getfh;
2916        nfssvc;
2917        _nfssys;
2918        __nis_get_environment;
2919        _nss_db_state_destr;
2920        nss_default_key2str;
2921        nss_delete;
2922        nss_endent;
2923        nss_getent;
2924        _nss_initf_group;
2925        _nss_initf_netgroup;
2926        _nss_initf_passwd;
2927        _nss_initf_shadow;
2928        nss_packed_arg_init;
2929        nss_packed_context_init;
2930        nss_packed_getkey;
2931        nss_packed_set_status;
2932        nss_search;
2933        nss_setent;
2934        _nss_XbyY_fgets;
2935        __nsw_extended_action_v1;
2936        __nsw_freeconfig_v1;
2937        __nsw_getconfig_v1;
2938        __nthreads;
```

```
2939         __openattrdirat;
2940         option_to_attr;
2941         __priv_bracket;
2942         __priv_relinquish;
2943         psecflags;
2944 #endif /* ! codereview */
2945         pset_assign_forced;
2946         pset_bind_lwp;
2947         _psignal;
2948         _pthread_setcleanupinit;
2949         __putwchar_xpg5;
2950         __putwc_xpg5;
2951         rctlctl;
2952         rctllist;
2953         _realbufend;
2954         _resume;
2955         _resume_ret;
2956         _rpcsys;
2957         _sbrk_grow_aligned;
2958         scrwidth;
2959         semctl64;
2960         _semctl64;
2961         set_setcontext_enforcement;
2962         _setbufend;
2963         __set_errno;
2964         setprojrctl;
2965         _setregid;
2966         _setreuid;
2967         setsigacthandler;
2968         shmctl64;
2969         _shmctl64;
2970         sigflag;
2971         _signal;
2972         _sigoff;
2973         _sigon;
2974         _so_accept;
2975         _so_bind;
2976         _sockconfig;
2977         _so_connect;
2978         _so_getpeername;
2979         _so_getsockname;
2980         _so_getsockopt;
2981         _so_listen;
2982         _so_recv;
2983         _so_recvfrom;
2984         _so_recvmsg;
2985         _so_send;
2986         _so_sendmsg;
2987         _so_sendto;
2988         _so_setsockopt;
2989         _so_shutdown;
2990         _so_socket;
2991         _so_socketpair;
2992         str2group;
2993         str2passwd;
2994         str2spwd;
2995         __strptime_dontzero;
2996         stty;
2997         syscall;
2998         _sysconfig;
2999         __systemcall;
3000         thr_continue_allmutators;
3001         _thr_continue_allmutators;
3002         thr_continue_mutator;
3003         _thr_continue_mutator;
3004         thr_getstate;
```



```
3005         _thr_getstate;
3006         thr_mutators_barrier;
3007         _thr_mutators_barrier;
3008         thr_probe_setup;
3009         _thr_schedctl;
3010         thr_setmutator;
3011         _thr_setmutator;
3012         thr_setstate;
3013         _thr_setstate;
3014         thr_sighndlrinfo;
3015         _thr_sighndlrinfo;
3016         _thr_slot_offset;
3017         thr_suspend_allmutators;
3018         _thr_suspend_allmutators;
3019         thr_suspend_mutator;
3020         _thr_suspend_mutator;
3021         thr_wait_mutator;
3022         _thr_wait_mutator;
3023         __tls_get_addr;
3024         _tmem_get_base;
3025         _tmem_get_nentries;
3026         _tmem_set_cleanup;
3027         tpool_create;
3028         tpool_dispatch;
3029         tpool_destroy;
3030         tpool_wait;
3031         tpool_suspend;
3032         tpool_suspended;
3033         tpool_resume;
3034         tpool_member;
3035         _ttyname_dev;
3036         _ucred_alloc;
3037         ucred_getamask;
3038         _ucred_getamask;
3039         ucred_getasid;
3040         _ucred_getasid;
3041         ucred_getatid;
3042         _ucred_getatid;
3043         ucred_getauid;
3044         _ucred_getauid;
3045         _ulltostr;
3046         _uncached_getgrgid_r;
3047         _uncached_getgrnam_r;
3048         _uncached_getpwnam_r;
3049         _uncached_getpwuid_r;
3050         __ungetwc_xpg5;
3051         _unordered;
3052         utssys;
3053         _verrfp;
3054         _verrxfp;
3055         _vwarnfp;
3056         _vwarnxfp;
3057         _warnfp;
3058         _warnxfp;
3059         __wcsftime_xpg5;
3060         __wcstok_xpg5;
3061         wdbindf;
3062         wdchkind;
3063         wddelim;
3064         _wrtchk;
3065         _xflsbuf;
3066         _xgetwidth;
3067         zone_add_datalink;
3068         zone_boot;
3069         zone_check_datalink;
3070         zone_create;
```

```
3071         zone_destroy;
3072         zone_enter;
3073         zone_getattr;
3074         zone_get_id;
3075         zone_list;
3076         zone_list_datalink;
3077         zonept;
3078         zone_remove_datalink;
3079         zone_setattr;
3080         zone_shutdown;
3081         zone_version;

3083 $if _ELF32
3084         __divdi3;
3085         _file_set;
3086         _fprintf_c89;
3087         _fscanf_c89;
3088         _fwprintf_c89;
3089         _fwscanf_c89;
3090         _imaxabs_c89;
3091         _imaxdiv_c89;
3092         __moddi3;
3093         _printf_c89;
3094         _scanf_c89;
3095         _snprintf_c89;
3096         _sprintf_c89;
3097         _sscanf_c89;
3098         _strtoimax_c89;
3099         _strtoumax_c89;
3100         _swprintf_c89;
3101         _swscanf_c89;
3102         __udivdi3;
3103         __umoddi3;
3104         _vfprintf_c89;
3105         _vfscanf_c89;
3106         _vfwprintf_c89;
3107         _vfwscanf_c89;
3108         _vprintf_c89;
3109         _vscanf_c89;
3110         _vsnprintf_c89;
3111         _vsprintf_c89;
3112         _vsscanf_c89;
3113         _vswprintf_c89;
3114         _vswscanf_c89;
3115         _vwprintf_c89;
3116         _vwscanf_c89;
3117         _wcstoimax_c89;
3118         _wcstoumax_c89;
3119         _wprintf_c89;
3120         _wscanf_c89;
3121 $endif

3123 $if _sparc
3124         _cerror;
3125         install_utrap;
3126         _install_utrap;
3127         nop;
3128         _Q_cplx_div;
3129         _Q_cplx_div_ix;
3130         _Q_cplx_div_rx;
3131         _Q_cplx_lr_div;
3132         _Q_cplx_lr_div_ix;
3133         _Q_cplx_lr_div_rx;
3134         _Q_cplx_lr_mul;
3135         _Q_cplx_mul;
3136         _QgetRD;
```



```
3137         _xregs_clrptr;
3138 $endif

3140 $if sparc32
3141         __ashldi3;
3142         __ashrdi3;
3143         _cerror64;
3144         __cmpdi2;
3145         __floatdidf;
3146         __floatdisf;
3147         __floatundidf;
3148         __floatundisf;
3149         __lshrdi3;
3150         __muldi3;
3151         __ucmpdi2;
3152 $endif

3154 $if _x86
3155         _D_cplx_lr_div;
3156         _D_cplx_lr_div_ix;
3157         _D_cplx_lr_div_rx;
3158         _F_cplx_lr_div;
3159         _F_cplx_lr_div_ix;
3160         _F_cplx_lr_div_rx;
3161         __fltrounds;
3162         sysi86;
3163         _sysi86;
3164         _X_cplx_div;
3165         _X_cplx_div_ix;
3166         _X_cplx_div_rx;
3167         _X_cplx_lr_div;
3168         _X_cplx_lr_div_ix;
3169         _X_cplx_lr_div_rx;
3170         _X_cplx_mul;
3171         __xgetRD;
3172         __xtol;
3173         __xtoll;
3174         __xtoul;
3175         __xtoull;
3176 $endif

3178 $if i386
3179         __divrem64;
3180         ___tls_get_addr;
3181         __udivrem64;
3182 $endif

3184 # The following functions should not be exported from libc,
3185 # but /lib/libm.so.2, some older versions of the Studio
3186 # compiler/debugger components, and some ancient programs
3187 # found in /usr/dist reference them.  When we no longer
3188 # care about these old and broken binary objects, these
3189 # symbols should be deleted.
3190         _brk                                    { FLAGS = NODYNSORT };
3191         _cond_broadcast                         { FLAGS = NODYNSORT };
3192         _cond_init                              { FLAGS = NODYNSORT };
3193         _cond_signal                            { FLAGS = NODYNSORT };
3194         _cond_wait                              { FLAGS = NODYNSORT };
3195         _ecvt                                   { FLAGS = NODYNSORT };
3196         _fcvt                                   { FLAGS = NODYNSORT };
3197         _getc_unlocked                          { FLAGS = NODYNSORT };
3198         _llseek                                 { FLAGS = NODYNSORT };
3199         _pthread_attr_getdetachstate            { FLAGS = NODYNSORT };
3200         _pthread_attr_getinheritsched           { FLAGS = NODYNSORT };
3201         _pthread_attr_getschedparam             { FLAGS = NODYNSORT };
3202         _pthread_attr_getschedpolicy            { FLAGS = NODYNSORT };
```

```
3203         _pthread_attr_getscope                  { FLAGS = NODYNSORT };
3204         _pthread_attr_getstackaddr              { FLAGS = NODYNSORT };
3205         _pthread_attr_getstacksize              { FLAGS = NODYNSORT };
3206         _pthread_attr_init                      { FLAGS = NODYNSORT };
3207         _pthread_condattr_getpshared            { FLAGS = NODYNSORT };
3208         _pthread_condattr_init                  { FLAGS = NODYNSORT };
3209         _pthread_cond_init                      { FLAGS = NODYNSORT };
3210         _pthread_create                         { FLAGS = NODYNSORT };
3211         _pthread_getschedparam                  { FLAGS = NODYNSORT };
3212         _pthread_join                           { FLAGS = NODYNSORT };
3213         _pthread_key_create                     { FLAGS = NODYNSORT };
3214         _pthread_mutexattr_getprioceiling       { FLAGS = NODYNSORT };
3215         _pthread_mutexattr_getprotocol          { FLAGS = NODYNSORT };
3216         _pthread_mutexattr_getpshared           { FLAGS = NODYNSORT };
3217         _pthread_mutexattr_init                 { FLAGS = NODYNSORT };
3218         _pthread_mutex_getprioceiling           { FLAGS = NODYNSORT };
3219         _pthread_mutex_init                     { FLAGS = NODYNSORT };
3220         _pthread_sigmask                        { FLAGS = NODYNSORT };
3221         _rwlock_init                            { FLAGS = NODYNSORT };
3222         _rw_rdlock                              { FLAGS = NODYNSORT };
3223         _rw_unlock                              { FLAGS = NODYNSORT };
3224         _rw_wrlock                              { FLAGS = NODYNSORT };
3225         _sbrk_unlocked                          { FLAGS = NODYNSORT };
3226         _select                                 { FLAGS = NODYNSORT };
3227         _sema_init                              { FLAGS = NODYNSORT };
3228         _sema_post                              { FLAGS = NODYNSORT };
3229         _sema_trywait                           { FLAGS = NODYNSORT };
3230         _sema_wait                              { FLAGS = NODYNSORT };
3231         _sysfs                                  { FLAGS = NODYNSORT };
3232         _thr_create                             { FLAGS = NODYNSORT };
3233         _thr_exit                               { FLAGS = NODYNSORT };
3234         _thr_getprio                            { FLAGS = NODYNSORT };
3235         _thr_getspecific                        { FLAGS = NODYNSORT };
3236         _thr_join                               { FLAGS = NODYNSORT };
3237         _thr_keycreate                          { FLAGS = NODYNSORT };
3238         _thr_kill                               { FLAGS = NODYNSORT };
3239         _thr_main                               { FLAGS = NODYNSORT };
3240         _thr_self                               { FLAGS = NODYNSORT };
3241         _thr_setspecific                        { FLAGS = NODYNSORT };
3242         _thr_sigsetmask                         { FLAGS = NODYNSORT };
3243         _thr_stksegment                         { FLAGS = NODYNSORT };
3244         _ungetc_unlocked                        { FLAGS = NODYNSORT };

3246     local:
3247         __imax_lldiv                            { FLAGS = NODYNSORT };
3248         _ti_thr_self                            { FLAGS = NODYNSORT };
3249         *;

3251 $if lf64
3252         _seekdir64              { FLAGS = NODYNSORT };
3253         _telldir64              { FLAGS = NODYNSORT };
3254 $endif

3256 $if _sparc
3257         __cerror                { FLAGS = NODYNSORT };
3258 $endif

3260 $if sparc32
3261         __cerror64              { FLAGS = NODYNSORT };
3262 $endif

3264 $if sparcv9
3265         __cleanup               { FLAGS = NODYNSORT };
3266 $endif

3268 $if i386
```



```
3269         _syscall6               { FLAGS = NODYNSORT };
3270         __systemcall6           { FLAGS = NODYNSORT };
3271 $endif

3273 $if amd64
3274         ___tls_get_addr         { FLAGS = NODYNSORT };
3275 $endif
3276 };
```

```
**********************************************************
    4702 Wed May 27 19:49:13 2015
new/usr/src/lib/libc/port/sys/sbrk.c
libc: adjust brk(0) to return the existing break, and use it to initialize sbrk(
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright 2008 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

#pragma ident   "%Z%%M% %I%     %E% SMI"

#pragma weak _sbrk = sbrk
#pragma weak _brk = brk

#include "lint.h"
#include <synch.h>
#include <errno.h>
#include <sys/isa_defs.h>
#include <sys/types.h>
#include <sys/sysmacros.h>
#include <inttypes.h>
#include <unistd.h>
#include "mtlib.h"
#include "libc.h"

void *_nd = NULL;
extern int _end;
void *_nd = &_end;
mutex_t __sbrk_lock = DEFAULTMUTEX;

extern intptr_t _brk_unlocked(void *);
void *_sbrk_unlocked(intptr_t);
extern int _brk_unlocked(void *);
extern void *_sbrk_unlocked(intptr_t);

/*
 * The break must always be at least 8-byte aligned
 */
#if (_MAX_ALIGNMENT < 8)
#define ALIGNSZ         8
#else
#define ALIGNSZ         _MAX_ALIGNMENT
#endif
```



```
#define BRKALIGN(x)     (caddr_t)P2ROUNDUP((uintptr_t)(x), ALIGNSZ)

void *
sbrk(intptr_t addend)
{
        void *result;

        if (!primary_link_map) {
                errno = ENOTSUP;
                return ((void *)-1);
        }
        lmutex_lock(&__sbrk_lock);
        result = _sbrk_unlocked(addend);
        lmutex_unlock(&__sbrk_lock);

        return (result);
}

/*
 * _sbrk_unlocked() aligns the old break, adds the addend, aligns
 * the new break, and calls _brk_unlocked() to set the new break.
 * We must align the old break because _nd may begin life misaligned.
 * The addend can be either positive or negative, so there are two
 * overflow/underflow edge conditions to reject:
 *
 *   - the addend is negative and brk + addend < 0.
 *   - the addend is positive and brk + addend > ULONG_MAX
 */
void *
_sbrk_unlocked(intptr_t addend)
{
        char *old_brk;
        char *new_brk;

        if (_nd == NULL) {
                _nd = (void *)_brk_unlocked(0);
        }

        old_brk = BRKALIGN(_nd);
        new_brk = BRKALIGN(old_brk + addend);
        char *old_brk = BRKALIGN(_nd);
        char *new_brk = BRKALIGN(old_brk + addend);

        if ((addend > 0 && new_brk < old_brk) ||
            (addend < 0 && new_brk > old_brk)) {
                errno = ENOMEM;
                return ((void *)-1);
        }
        if (_brk_unlocked(new_brk) != 0)
                return ((void *)-1);
        _nd = new_brk;
        return (old_brk);
}

/*
 * _sbrk_grow_aligned() aligns the old break to a low_align boundry,
 * adds min_size, aligns to a high_align boundry, and calls _brk_unlocked()
 * to set the new break.  The low_aligned-aligned value is returned, and
 * the actual space allocated is returned through actual_size.
 *
 * Unlike sbrk(2), _sbrk_grow_aligned takes an unsigned size, and does
 * not allow shrinking the heap.
 */
void *
_sbrk_grow_aligned(size_t min_size, size_t low_align, size_t high_align,
    size_t *actual_size)
```

```
120 {
121         uintptr_t old_brk;
122         uintptr_t ret_brk;
123         uintptr_t high_brk;
124         uintptr_t new_brk;
125         int brk_result;

127         if (!primary_link_map) {
128                 errno = ENOTSUP;
129                 return ((void *)-1);
130         }
131         if ((low_align & (low_align - 1)) != 0 ||
132             (high_align & (high_align - 1)) != 0) {
133                 errno = EINVAL;
134                 return ((void *)-1);
135         }
136         low_align = MAX(low_align, ALIGNSZ);
137         high_align = MAX(high_align, ALIGNSZ);

139         lmutex_lock(&__sbrk_lock);

141         if (_nd == NULL) {
142                 _nd = (void *)_brk_unlocked(0);
143         }

145 #endif /* ! codereview */
146         old_brk = (uintptr_t)BRKALIGN(_nd);
147         ret_brk = P2ROUNDUP(old_brk, low_align);
148         high_brk = ret_brk + min_size;
149         new_brk = P2ROUNDUP(high_brk, high_align);

151         /*
152          * Check for overflow
153          */
154         if (ret_brk < old_brk || high_brk < ret_brk || new_brk < high_brk) {
155                 lmutex_unlock(&__sbrk_lock);
156                 errno = ENOMEM;
157                 return ((void *)-1);
158         }

160         if ((brk_result = (int)_brk_unlocked((void *)new_brk)) == 0)
137         if ((brk_result = _brk_unlocked((void *)new_brk)) == 0)
161                 _nd = (void *)new_brk;
162         lmutex_unlock(&__sbrk_lock);

164         if (brk_result != 0)
165                 return ((void *)-1);

167         if (actual_size != NULL)
168                 *actual_size = (new_brk - ret_brk);
169         return ((void *)ret_brk);
170 }

172 int
173 brk(void *new_brk)
174 {
175         int result;

177         /*
178          * brk(2) will return the current brk if given an argument of 0, so we
179          * need to fail it here
180          */
181         if (new_brk == 0) {
182                 errno = ENOMEM;
183                 return (-1);
184         }
```



```
186 #endif /* ! codereview */
187         if (!primary_link_map) {
188                 errno = ENOTSUP;
189                 return (-1);
190         }
191         /*
192          * Need to align this here;  _brk_unlocked won't do it for us.
193          */
194         new_brk = BRKALIGN(new_brk);

196         lmutex_lock(&__sbrk_lock);
197         if ((result = (int)_brk_unlocked(new_brk)) == 0)
154         if ((result = _brk_unlocked(new_brk)) == 0)
198                 _nd = new_brk;
199         lmutex_unlock(&__sbrk_lock);

201         return (result);
202 }
```

_____*unchanged_portion_omitted_*

```
**********************************************************
   26123 Wed May 27 19:49:14 2015
new/usr/src/lib/libc/sparc/Makefile.com
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
#
# Copyright (c) 1989, 2010, Oracle and/or its affiliates. All rights reserved.
# Copyright (c) 2015, Joyent, Inc.  All rights reserved.
# Copyright (c) 2013, OmniTI Computer Consulting, Inc. All rights reserved.
# Copyright 2013 Garrett D'Amore <garrett@damore.org>
#
# Copyright 2011 Nexenta Systems, Inc.  All rights reserved.
# Use is subject to license terms.
#

LIBCDIR=        $(SRC)/lib/libc
LIB_PIC=        libc_pic.a
VERS=           .1
CPP=            /usr/lib/cpp
TARGET_ARCH=    sparc

# objects are grouped by source directory

# Symbol capabilities objects.
EXTPICS=                                \
        $(LIBCDIR)/capabilities/sun4u/sparc/pics/symcap.o \
        $(LIBCDIR)/capabilities/sun4u-opl/sparc/pics/symcap.o \
        $(LIBCDIR)/capabilities/sun4u-us3-hwcap1/sparc/pics/symcap.o \
        $(LIBCDIR)/capabilities/sun4u-us3-hwcap2/sparc/pics/symcap.o \
        $(LIBCDIR)/capabilities/sun4v-hwcap1/sparc/pics/symcap.o \
        $(LIBCDIR)/capabilities/sun4v-hwcap2/sparc/pics/symcap.o

# local objects
STRETS=                         \
        stret1.o                \
        stret2.o                \
        stret4.o

CRTOBJS=                        \
        _ftou.o                 \
        cerror.o                \
        cerror64.o              \
        hwmuldiv.o
```



```
DYNOBJS=                        \
        _rtbootld.o

FPOBJS=                         \
        _D_cplx_div.o           \
        _D_cplx_div_ix.o        \
        _D_cplx_div_rx.o        \
        _D_cplx_mul.o           \
        _F_cplx_div.o           \
        _F_cplx_div_ix.o        \
        _F_cplx_div_rx.o        \
        _F_cplx_mul.o           \
        _Q_add.o                \
        _Q_cmp.o                \
        _Q_cmpe.o               \
        _Q_cplx_div.o           \
        _Q_cplx_div_ix.o        \
        _Q_cplx_div_rx.o        \
        _Q_cplx_lr_div.o        \
        _Q_cplx_lr_div_ix.o     \
        _Q_cplx_lr_div_rx.o     \
        _Q_cplx_lr_mul.o        \
        _Q_cplx_mul.o           \
        _Q_div.o                \
        _Q_dtoq.o               \
        _Q_fcc.o                \
        _Q_itoq.o               \
        _Q_lltoq.o              \
        _Q_mul.o                \
        _Q_neg.o                \
        _Q_qtod.o               \
        _Q_qtoi.o               \
        _Q_qtos.o               \
        _Q_qtou.o               \
        _Q_scl.o                \
        _Q_set_except.o         \
        _Q_sqrt.o               \
        _Q_stoq.o               \
        _Q_sub.o                \
        _Q_ulltoq.o             \
        _Q_utoq.o               \
        __quad_mag.o

FPASMOBJS=                      \
        _Q_get_rp_rd.o          \
        fpgetmask.o             \
        fpgetrnd.o              \
        fpgetsticky.o           \
        fpsetmask.o             \
        fpsetrnd.o              \
        fpsetsticky.o

$(__GNUC)FPASMOBJS +=           \
        __quad.o

ATOMICOBJS=                     \
        atomic.o

CHACHAOBJS=                     \
        chacha.o

XATTROBJS=                      \
        xattr_common.o

COMOBJS=                        \
```

```
125         bcmp.o                  \
126         bcopy.o                 \
127         bzero.o                 \
128         bsearch.o               \
129         memccpy.o               \
130         qsort.o                 \
131         strtol.o                \
132         strtoul.o               \
133         strtoll.o               \
134         strtoull.o

136 DTRACEOBJS=                      \
137         dtrace_data.o

139 GENOBJS=                         \
140         _getsp.o                \
141         _xregs_clrptr.o         \
142         abs.o                   \
143         alloca.o                \
144         arc4random.o            \
145         arc4random_uniform.o    \
146         ascii_strcasecmp.o      \
147         byteorder.o             \
148         cuexit.o                \
149         ecvt.o                  \
150         errlst.o                \
151         getctxt.o               \
152         ladd.o                  \
153         lmul.o                  \
154         lock.o                  \
155         lshiftl.o               \
156         lsign.o                 \
157         lsub.o                  \
158         makectxt.o              \
159         memchr.o                \
160         memcmp.o                \
161         new_list.o              \
162         setjmp.o                \
163         siginfolst.o            \
164         siglongjmp.o            \
165         smt_pause.o             \
166         sparc_data.o            \
167         strchr.o                \
168         strcmp.o                \
169         strlcpy.o               \
170         strncmp.o               \
171         strncpy.o               \
172         strnlen.o               \
173         swapctxt.o              \
174         sync_instruction_memory.o

176 # sysobjs that contain large-file interfaces
177 COMSYSOBJS64=                   \
178         fstatvfs64.o            \
179         getdents64.o            \
180         getrlimit64.o           \
181         lseek64.o               \
182         mmap64.o                \
183         pread64.o               \
184         preadv64.o              \
185         pwrite64.o              \
186         pwritev64.o             \
187         setrlimit64.o           \
188         statvfs64.o

190 SYSOBJS64=
```



```
192 COMSYSOBJS=                     \
193         __clock_timer.o         \
194         __getloadavg.o          \
195         __rusagesys.o           \
196         __signotify.o           \
197         __sigrt.o               \
198         __time.o                \
199         _lgrp_home_fast.o       \
200         _lgrpsys.o              \
201         _nfssys.o               \
202         _portfs.o               \
203         _pset.o                 \
204         _rpcsys.o               \
205         _sigaction.o            \
206         _so_accept.o            \
207         _so_bind.o              \
208         _so_connect.o           \
209         _so_getpeername.o       \
210         _so_getsockname.o       \
211         _so_getsockopt.o        \
212         _so_listen.o            \
213         _so_recv.o              \
214         _so_recvfrom.o          \
215         _so_recvmsg.o           \
216         _so_send.o              \
217         _so_sendmsg.o           \
218         _so_sendto.o            \
219         _so_setsockopt.o        \
220         _so_shutdown.o          \
221         _so_socket.o            \
222         _so_socketpair.o        \
223         _sockconfig.o           \
224         acct.o                  \
225         acl.o                   \
226         adjtime.o               \
227         alarm.o                 \
228         brk.o                   \
229         chdir.o                 \
230         chroot.o                \
231         cladm.o                 \
232         close.o                 \
233         execve.o                \
234         exit.o                  \
235         facl.o                  \
236         fchdir.o                \
237         fchroot.o               \
238         fdsync.o                \
239         fpathconf.o             \
240         fstatfs.o               \
241         fstatvfs.o              \
242         getcpuid.o              \
243         getdents.o              \
244         getegid.o               \
245         geteuid.o               \
246         getgid.o                \
247         getgroups.o             \
248         gethrtime.o             \
249         getitimer.o             \
250         getmsg.o                \
251         getpid.o                \
252         getpmsg.o               \
253         getppid.o               \
254         getrandom.o             \
255         getrlimit.o             \
256         getuid.o                \
```

```
257         gtty.o                  \
258         install_utrap.o         \
259         ioctl.o                 \
260         kaio.o                  \
261         kill.o                  \
262         llseek.o                \
263         lseek.o                 \
264         memcntl.o               \
265         mincore.o               \
266         mmap.o                  \
267         mmapobjsys.o            \
268         modctl.o                \
269         mount.o                 \
270         mprotect.o              \
271         munmap.o                \
272         nice.o                  \
273         ntp_adjtime.o           \
274         ntp_gettime.o           \
275         p_online.o              \
276         pathconf.o              \
277         pause.o                 \
278         pcsample.o              \
279         pipe2.o                 \
280         pollsys.o               \
281         pread.o                 \
282         preadv.o                \
283         priocntlset.o           \
284         processor_bind.o        \
285         processor_info.o        \
286         profil.o                \
287         psecflagsset.o          \
288 #endif /* ! codereview */
289         putmsg.o                \
290         putpmsg.o               \
291         pwrite.o                \
292         pwritev.o               \
293         read.o                  \
294         readv.o                 \
295         resolvepath.o           \
296         seteguid.o              \
297         setgid.o                \
298         setgroups.o             \
299         setitimer.o             \
300         setreid.o               \
301         setrlimit.o             \
302         setuid.o                \
303         sigaltstk.o             \
304         sigprocmsk.o            \
305         sigsendset.o            \
306         sigsuspend.o            \
307         statfs.o                \
308         statvfs.o               \
309         stty.o                  \
310         sync.o                  \
311         sysconfig.o             \
312         sysfs.o                 \
313         sysinfo.o               \
314         syslwp.o                \
315         times.o                 \
316         ulimit.o                \
317         umask.o                 \
318         umount2.o               \
319         utssys.o                \
320         uucopy.o                \
321         vhangup.o               \
322         waitid.o                \
```



```
323         write.o                 \
324         writev.o                \
325         yield.o

327 SYSOBJS=                        \
328         __clock_gettime.o       \
329         __getcontext.o          \
330         _lwp_mutex_unlock.o     \
331         _stack_grow.o           \
332         __uadmin.o              \
333         door.o                  \
334         forkx.o                 \
335         forkallx.o              \
336         gettimeofday.o          \
337         ptrace.o                \
338         syscall.o               \
339         tls_get_addr.o          \
340         uadmin.o                \
341         umount.o                \
342         uname.o                 \
343         vforkx.o

345 # objects under $(LIBCDIR)/port which contain transitional large file interfaces
346 PORTGEN64=                      \
347         _xftw64.o               \
348         attropen64.o            \
349         ftw64.o                 \
350         mkstemp64.o             \
351         nftw64.o                \
352         tell64.o                \
353         truncate64.o

355 # objects from source under $(LIBCDIR)/port
356 PORTFP=                         \
357         __flt_decim.o           \
358         __flt_rounds.o          \
359         __tbl_10_b.o            \
360         __tbl_10_h.o            \
361         __tbl_10_s.o            \
362         __tbl_2_b.o             \
363         __tbl_2_h.o             \
364         __tbl_2_s.o             \
365         __tbl_fdq.o             \
366         __tbl_tens.o            \
367         __x_power.o             \
368         _base_sup.o             \
369         aconvert.o              \
370         decimal_bin.o           \
371         double_decim.o          \
372         econvert.o              \
373         fconvert.o              \
374         file_decim.o            \
375         finite.o                \
376         fp_data.o               \
377         func_decim.o            \
378         gconvert.o              \
379         hex_bin.o               \
380         ieee_globals.o          \
381         pack_float.o            \
382         sigfpe.o                \
383         string_decim.o          \
384         ashldi3.o               \
385         ashrdi3.o               \
386         cmpdi2.o                \
387         divdi3.o                \
388         floatdidf.o             \
```

```
389         floatdisf.o             \
390         floatundidf.o           \
391         floatundisf.o           \
392         lshrdi3.o               \
393         moddi3.o                \
394         muldi3.o                \
395         qdivrem.o               \
396         ucmpdi2.o               \
397         udivdi3.o               \
398         umoddi3.o

400 PORTGEN=                        \
401         _env_data.o             \
402         _ftoll.o                \
403         _ftoull.o               \
404         _xftw.o                 \
405         a64l.o                  \
406         abort.o                 \
407         addsev.o                \
408         ascii_strncasecmp.o     \
409         assert.o                \
410         atof.o                  \
411         atoi.o                  \
412         atol.o                  \
413         atoll.o                 \
414         attrat.o                \
415         attropen.o              \
416         atexit.o                \
417         atfork.o                \
418         basename.o              \
419         calloc.o                \
420         catgets.o               \
421         catopen.o               \
422         cfgetispeed.o           \
423         cfgetospeed.o           \
424         cfree.o                 \
425         cfsetispeed.o           \
426         cfsetospeed.o           \
427         cftime.o                \
428         clock.o                 \
429         closedir.o              \
430         closefrom.o             \
431         confstr.o               \
432         crypt.o                 \
433         csetlen.o               \
434         ctime.o                 \
435         ctime_r.o               \
436         daemon.o                \
437         deflt.o                 \
438         directio.o              \
439         dirname.o               \
440         div.o                   \
441         drand48.o               \
442         dup.o                   \
443         env_data.o              \
444         err.o                   \
445         errno.o                 \
446         euclen.o                \
447         event_port.o            \
448         execvp.o                \
449         explicit_bzero.o        \
450         fattach.o               \
451         fdetach.o               \
452         fdopendir.o             \
453         ffs.o                   \
454         fls.o                   \
```



```
455         fmtmsg.o                \
456         ftime.o                 \
457         ftok.o                  \
458         ftw.o                   \
459         gcvt.o                  \
460         getauxv.o               \
461         getcwd.o                \
462         getdate_err.o           \
463         getdtblsize.o           \
464         getentropy.o            \
465         getenv.o                \
466         getexecname.o           \
467         getgrnam.o              \
468         getgrnam_r.o            \
469         gethostid.o             \
470         gethostname.o           \
471         gethz.o                 \
472         getisax.o               \
473         getloadavg.o            \
474         getlogin.o              \
475         getmntent.o             \
476         getnetgrent.o           \
477         get_nprocs.o            \
478         getopt.o                \
479         getopt_long.o           \
480         getpagesize.o           \
481         getpw.o                 \
482         getpwnam.o              \
483         getpwnam_r.o            \
484         getrusage.o             \
485         getspent.o              \
486         getspent_r.o            \
487         getsubopt.o             \
488         gettxt.o                \
489         getusershell.o          \
490         getut.o                 \
491         getutx.o                \
492         getvfsent.o             \
493         getwd.o                 \
494         getwidth.o              \
495         getxby_door.o           \
496         gtxt.o                  \
497         hsearch.o               \
498         iconv.o                 \
499         imaxabs.o               \
500         imaxdiv.o               \
501         index.o                 \
502         initgroups.o            \
503         insque.o                \
504         isaexec.o               \
505         isastream.o             \
506         isatty.o                \
507         killpg.o                \
508         klpdlib.o               \
509         l64a.o                  \
510         lckpwdf.o               \
511         lconstants.o            \
512         ldivide.o               \
513         lexp10.o                \
514         lfind.o                 \
515         lfmt.o                  \
516         lfmt_log.o              \
517         llabs.o                 \
518         lldiv.o                 \
519         llog10.o                \
520         lltostr.o               \
```

```
521         localtime.o             \
522         lsearch.o               \
523         madvise.o               \
524         malloc.o                \
525         memalign.o              \
526         memmem.o                \
527         mkdev.o                 \
528         mkdtemp.o               \
529         mkfifo.o                \
530         mkstemp.o               \
531         mktemp.o                \
532         mlock.o                 \
533         mlockall.o              \
534         mon.o                   \
535         msync.o                 \
536         munlock.o               \
537         munlockall.o            \
538         ndbm.o                  \
539         nftw.o                  \
540         nlspath_checks.o        \
541         nsparse.o               \
542         nss_common.o            \
543         nss_dbdefs.o            \
544         nss_deffinder.o         \
545         opendir.o               \
546         opt_data.o              \
547         perror.o                \
548         pfmt.o                  \
549         pfmt_data.o             \
550         pfmt_print.o            \
551         pipe.o                  \
552         plock.o                 \
553         poll.o                  \
554         posix_fadvise.o         \
555         posix_fallocate.o       \
556         posix_madvise.o         \
557         posix_memalign.o        \
558         priocntl.o              \
559         privlib.o               \
560         priv_str_xlate.o        \
561         psecflags.o             \
562 #endif /* ! codereview */
563         psiginfo.o              \
564         psignal.o               \
565         pt.o                    \
566         putpwent.o              \
567         putspent.o              \
568         raise.o                 \
569         rand.o                  \
570         random.o                \
571         rctlops.o               \
572         readdir.o               \
573         readdir_r.o             \
574         realpath.o              \
575         reboot.o                \
576         regexpr.o               \
577         remove.o                \
578         rewinddir.o             \
579         rindex.o                \
580         scandir.o               \
581         seekdir.o               \
582         select.o                \
583         select_large_fdset.o    \
584         setlabel.o              \
585         setpriority.o           \
586         settimeofday.o          \
```



```
587         sh_locks.o              \
588         sigflag.o               \
589         siglist.o               \
590         sigsend.o               \
591         sigsetops.o             \
592         ssignal.o               \
593         stack.o                 \
594         stpcpy.o                \
595         stpncpy.o               \
596         str2sig.o               \
597         strcase_charmap.o       \
598         strcat.o                \
599         strchrnul.o             \
600         strcspn.o               \
601         strdup.o                \
602         strerror.o              \
603         strlcat.o               \
604         strncat.o               \
605         strndup.o               \
606         strpbrk.o               \
607         strrchr.o               \
608         strsep.o                \
609         strsignal.o             \
610         strspn.o                \
611         strstr.o                \
612         strtod.o                \
613         strtoimax.o             \
614         strtok.o                \
615         strtok_r.o              \
616         strtoumax.o             \
617         swab.o                  \
618         swapctl.o               \
619         sysconf.o               \
620         syslog.o                \
621         tcdrain.o               \
622         tcflow.o                \
623         tcflush.o               \
624         tcgetattr.o             \
625         tcgetpgrp.o             \
626         tcgetsid.o              \
627         tcsendbreak.o           \
628         tcsetattr.o             \
629         tcsetpgrp.o             \
630         tell.o                  \
631         telldir.o               \
632         tfind.o                 \
633         time_data.o             \
634         time_gdata.o            \
635         tls_data.o              \
636         truncate.o              \
637         tsdalloc.o              \
638         tsearch.o               \
639         ttyname.o               \
640         ttyslot.o               \
641         ualarm.o                \
642         ucred.o                 \
643         valloc.o                \
644         vlfmt.o                 \
645         vpfmt.o                 \
646         waitpid.o               \
647         walkstack.o             \
648         wdata.o                 \
649         xgetwidth.o             \
650         xpg4.o                  \
651         xpg6.o
```

```
653 PORTPRINT_W=                    \
654         doprnt_w.o

656 PORTPRINT=                      \
657         asprintf.o              \
658         doprnt.o                \
659         fprintf.o               \
660         printf.o                \
661         snprintf.o              \
662         sprintf.o               \
663         vfprintf.o              \
664         vprintf.o               \
665         vsnprintf.o             \
666         vsprintf.o              \
667         vwprintf.o              \
668         wprintf.o

670 # c89 variants to support 32-bit size of c89 u/intmax_t (32-bit libc only)
671 PORTPRINT_C89=                  \
672         vfprintf_c89.o          \
673         vprintf_c89.o           \
674         vsnprintf_c89.o         \
675         vsprintf_c89.o          \
676         vwprintf_c89.o

678 PORTSTDIO_C89=                  \
679         vscanf_c89.o            \
680         vwscanf_c89.o

682 # portable stdio objects that contain large file interfaces.
683 # Note: fopen64 is a special case, as we build it small.
684 PORTSTDIO64=                    \
685         fopen64.o               \
686         fpos64.o

688 PORTSTDIO_W=                    \
689         doscan_w.o

691 PORTSTDIO=                      \
692         __extensions.o          \
693         _endopen.o              \
694         _filbuf.o               \
695         _findbuf.o              \
696         _flsbuf.o               \
697         _wrtchk.o               \
698         clearerr.o              \
699         ctermid.o               \
700         ctermid_r.o             \
701         cuserid.o               \
702         data.o                  \
703         doscan.o                \
704         fdopen.o                \
705         feof.o                  \
706         ferror.o                \
707         fgetc.o                 \
708         fgets.o                 \
709         fileno.o                \
710         flockf.o                \
711         flush.o                 \
712         fopen.o                 \
713         fpos.o                  \
714         fputc.o                 \
715         fputs.o                 \
716         fread.o                 \
717         fseek.o                 \
718         fseeko.o                \
```



```
719         ftell.o                 \
720         ftello.o                \
721         fwrite.o                \
722         getc.o                  \
723         getchar.o               \
724         getline.o               \
725         getpass.o               \
726         gets.o                  \
727         getw.o                  \
728         popen.o                 \
729         putc.o                  \
730         putchar.o               \
731         puts.o                  \
732         putw.o                  \
733         rewind.o                \
734         scanf.o                 \
735         setbuf.o                \
736         setbuffer.o             \
737         setvbuf.o               \
738         system.o                \
739         tempnam.o               \
740         tmpfile.o               \
741         tmpnam_r.o              \
742         ungetc.o                \
743         mse.o                   \
744         vscanf.o                \
745         vwscanf.o               \
746         wscanf.o

748 PORTI18N=                       \
749         getwchar.o              \
750         putwchar.o              \
751         putws.o                 \
752         strtows.o               \
753         wcsnlen.o               \
754         wcstoimax.o             \
755         wcstol.o                \
756         wcstoul.o               \
757         wcswcs.o                \
758         wscat.o                 \
759         wschr.o                 \
760         wscmp.o                 \
761         wscpy.o                 \
762         wscspn.o                \
763         wsdup.o                 \
764         wslen.o                 \
765         wsncat.o                \
766         wsncmp.o                \
767         wsncpy.o                \
768         wspbrk.o                \
769         wsprintf.o              \
770         wsrchr.o                \
771         wsscanf.o               \
772         wsspn.o                 \
773         wstod.o                 \
774         wstok.o                 \
775         wstol.o                 \
776         wstoll.o                \
777         wsxfrm.o                \
778         wmemchr.o               \
779         wmemcmp.o               \
780         wmemcpy.o               \
781         wmemmove.o              \
782         wmemset.o               \
783         wcsstr.o                \
784         gettext.o               \
```

```
785         gettext_real.o          \
786         gettext_util.o          \
787         gettext_gnu.o           \
788         plural_parser.o         \
789         wdresolve.o             \
790         _ctype.o                \
791         isascii.o               \
792         toascii.o

794 PORTI18N_COND=                  \
795         wcstol_longlong.o       \
796         wcstoul_longlong.o

798 PORTLOCALE=                    \
799         big5.o                  \
800         btowc.o                 \
801         collate.o               \
802         collcmp.o               \
803         euc.o                   \
804         fnmatch.o               \
805         fgetwc.o                \
806         fgetws.o                \
807         fix_grouping.o          \
808         fputwc.o                \
809         fputws.o                \
810         fwide.o                 \
811         gb18030.o               \
812         gb2312.o                \
813         gbk.o                   \
814         getdate.o               \
815         isdigit.o               \
816         iswctype.o              \
817         ldpart.o                \
818         lmessages.o             \
819         lnumeric.o              \
820         lmonetary.o             \
821         localeimpl.o            \
822         localeconv.o            \
823         mbftowc.o               \
824         mblen.o                 \
825         mbrlen.o                \
826         mbrtowc.o               \
827         mbsinit.o               \
828         mbsnrtowcs.o            \
829         mbsrtowcs.o             \
830         mbstowcs.o              \
831         mbtowc.o                \
832         mskanji.o               \
833         nextwctype.o            \
834         nl_langinfo.o           \
835         none.o                  \
836         regcomp.o               \
837         regfree.o               \
838         regerror.o              \
839         regexec.o               \
840         rune.o                  \
841         runetype.o              \
842         setlocale.o             \
843         setrunelocale.o         \
844         strcasecmp.o            \
845         strcasestr.o            \
846         strcoll.o               \
847         strfmon.o               \
848         strftime.o              \
849         strncasecmp.o           \
850         strptime.o              \
```



```
851         strxfrm.o               \
852         table.o                 \
853         timelocal.o             \
854         tolower.o               \
855         towlower.o              \
856         ungetwc.o               \
857         utf8.o                  \
858         wcrtomb.o               \
859         wcscasecmp.o            \
860         wcscoll.o               \
861         wcsftime.o              \
862         wcsnrtombs.o            \
863         wcsrtombs.o             \
864         wcstombs.o              \
865         wcswidth.o              \
866         wcsxfrm.o               \
867         wctob.o                 \
868         wctomb.o                \
869         wctrans.o               \
870         wctype.o                \
871         wcwidth.o               \
872         wscol.o

874 AIOOBJS=                       \
875         aio.o                   \
876         aio_alloc.o             \
877         posix_aio.o

879 RTOBJS=                        \
880         clock_timer.o           \
881         mqueue.o                \
882         pos4obj.o               \
883         sched.o                 \
884         sem.o                   \
885         shm.o                   \
886         sigev_thread.o

888 TPOOLOBJS=                     \
889         thread_pool.o

891 THREADSOBJS=                   \
892         alloc.o                 \
893         assfail.o               \
894         cancel.o                \
895         door_calls.o            \
896         tmem.o                  \
897         pthr_attr.o             \
898         pthr_barrier.o          \
899         pthr_cond.o             \
900         pthr_mutex.o            \
901         pthr_rwlock.o           \
902         pthread.o               \
903         rwlock.o                \
904         scalls.o                \
905         sema.o                  \
906         sigaction.o             \
907         spawn.o                 \
908         synch.o                 \
909         tdb_agent.o             \
910         thr.o                   \
911         thread_interface.o      \
912         tls.o                   \
913         tsd.o

915 THREADSMACHOBJS=               \
916         machdep.o
```

```
THREADSASMOBJS=                 \
        asm_subr.o

UNICODEOBJS=                    \
        u8_textprep.o           \
        uconv.o

UNWINDMACHOBJS=                 \
        unwind.o

UNWINDASMOBJS=                  \
        unwind_frame.o

# objects that implement the transitional large file API
PORTSYS64=                      \
        lockf64.o               \
        stat64.o

PORTSYS=                        \
        _autofssys.o            \
        access.o                \
        acctctl.o               \
        bsd_signal.o            \
        chmod.o                 \
        chown.o                 \
        corectl.o               \
        exacctsys.o             \
        execl.o                 \
        execle.o                \
        execv.o                 \
        fcntl.o                 \
        getpagesizes.o          \
        getpeerucred.o          \
        inst_sync.o             \
        issetugid.o             \
        label.o                 \
        link.o                  \
        lockf.o                 \
        lwp.o                   \
        lwp_cond.o              \
        lwp_rwlock.o            \
        lwp_sigmask.o           \
        meminfosys.o            \
        mkdir.o                 \
        mknod.o                 \
        msgsys.o                \
        nfssys.o                \
        open.o                  \
        pgrpsys.o               \
        posix_sigwait.o         \
        ppriv.o                 \
        psetsys.o               \
        rctlsys.o               \
        readlink.o              \
        rename.o                \
        sbrk.o                  \
        semsys.o                \
        set_errno.o             \
        sharefs.o               \
        shmsys.o                \
        sidsys.o                \
        siginterrupt.o          \
        signal.o                \
        sigpending.o            \
        sigstack.o              \
```



```
        stat.o                  \
        symlink.o               \
        tasksys.o               \
        time.o                  \
        time_util.o             \
        ucontext.o              \
        unlink.o                \
        ustat.o                 \
        utimesys.o              \
        zone.o

PORTREGEX=                      \
        glob.o                  \
        regcmp.o                \
        regex.o                 \
        wordexp.o

VALUES= values-Xa.o

MOSTOBJS=                       \
        $(STRETS)               \
        $(CRTOBJS)              \
        $(DYNOBJS)              \
        $(FPOBJS)               \
        $(FPASMOBJS)            \
        $(ATOMICOBJS)           \
        $(CHACHAOBJS)           \
        $(XATTROBJS)            \
        $(COMOBJS)              \
        $(DTRACEOBJS)           \
        $(GENOBJS)              \
        $(PRFOBJS)              \
        $(PORTFP)               \
        $(PORTGEN)              \
        $(PORTGEN64)            \
        $(PORTI18N)             \
        $(PORTI18N_COND)        \
        $(PORTLOCALE)           \
        $(PORTPRINT)            \
        $(PORTPRINT_C89)        \
        $(PORTPRINT_W)          \
        $(PORTREGEX)            \
        $(PORTSTDIO)            \
        $(PORTSTDIO64)          \
        $(PORTSTDIO_C89)        \
        $(PORTSTDIO_W)          \
        $(PORTSYS)              \
        $(PORTSYS64)            \
        $(AIOOBJS)              \
        $(RTOBJS)               \
        $(TPOOLOBJS)            \
        $(THREADSOBJS)          \
        $(THREADSMACHOBJS)      \
        $(THREADSASMOBJS)       \
        $(UNICODEOBJS)          \
        $(UNWINDMACHOBJS)       \
        $(UNWINDASMOBJS)        \
        $(COMSYSOBJS)           \
        $(SYSOBJS)              \
        $(COMSYSOBJS64)         \
        $(SYSOBJS64)            \
        $(VALUES)

TRACEOBJS=                      \
        plockstat.o
```

```
# NOTE: libc.so.1 must be linked with the minimal crti.o and crtn.o
# modules whose source is provided in the $(SRC)/lib/common directory.
# This must be done because otherwise the Sun C compiler would insert
# its own versions of these modules and those versions contain code
# to call out to C++ initialization functions.  Such C++ initialization
# functions can call back into libc before thread initialization is
# complete and this leads to segmentation violations and other problems.
# Since libc contains no C++ code, linking with the minimal crti.o and
# crtn.o modules is safe and avoids the problems described above.
OBJECTS= $(CRTI) $(MOSTOBJS) $(CRTN)
CRTSRCS= ../../common/sparc

# include common library definitions
include $(SRC)/lib/Makefile.lib

# we need to override the default SONAME here because we might
# be building a variant object (still libc.so.1, but different filename)
SONAME = libc.so.1

CFLAGS += $(CCVERBOSE)

# This is necessary to avoid problems with calling _ex_unwind().
# We probably don't want any inlining anyway.
CFLAGS += -xinline=

CERRWARN += -_gcc=-Wno-parentheses
CERRWARN += -_gcc=-Wno-switch
CERRWARN += -_gcc=-Wno-uninitialized
CERRWARN += -_gcc=-Wno-unused-value
CERRWARN += -_gcc=-Wno-unused-label
CERRWARN += -_gcc=-Wno-unused-variable
CERRWARN += -_gcc=-Wno-type-limits
CERRWARN += -_gcc=-Wno-char-subscripts
CERRWARN += -_gcc=-Wno-clobbered
CERRWARN += -_gcc=-Wno-unused-function
CERRWARN += -_gcc=-Wno-address

# Setting THREAD_DEBUG = -DTHREAD_DEBUG (make THREAD_DEBUG=-DTHREAD_DEBUG ...)
# enables ASSERT() checking in the threads portion of the library.
# This is automatically enabled for DEBUG builds, not for non-debug builds.
THREAD_DEBUG =
$(NOT_RELEASE_BUILD)THREAD_DEBUG = -DTHREAD_DEBUG

# Make string literals read-only to save memory.
CFLAGS += $(XSTRCONST)

ALTPICS= $(TRACEOBJS:%=pics/%)

$(DYNLIB) := BUILD.SO = $(LD) -o $@ -G $(DYNFLAGS) $(PICS) $(ALTPICS) $(EXTPICS)

MAPFILES =      $(LIBCDIR)/port/mapfile-vers

CFLAGS +=       $(EXTN_CFLAGS)
CPPFLAGS=       -D_REENTRANT -Dsparc $(EXTN_CPPFLAGS) $(THREAD_DEBUG) \
                -I$(LIBCBASE)/inc -I$(LIBCDIR)/inc $(CPPFLAGS.master)
ASFLAGS=        $(EXTN_ASFLAGS) -K pic -P -D__STDC__ -D_ASM $(CPPFLAGS) $(sparc_

# As a favor to the dtrace syscall provider, libc still calls the
# old syscall traps that have been obsoleted by the *at() interfaces.
# Delete this to compile libc using only the new *at() system call traps
CPPFLAGS += -D_RETAIN_OLD_SYSCALLS

# Inform the run-time linker about libc specialized initialization
RTLDINFO =      -z rtldinfo=tls_rtldinfo
DYNFLAGS +=     $(RTLDINFO)
```



```
# Force libc's internal references to be resolved immediately upon loading
# in order to avoid critical region problems.  Since almost all libc symbols
# are marked 'protected' in the mapfiles, this is a minimal set (15 to 20).
DYNFLAGS +=     -znow

DYNFLAGS +=     -e __rtboot
DYNFLAGS +=     $(EXTN_DYNFLAGS)

# Inform the kernel about the initial DTrace area (in case
# libc is being used as the interpreter / runtime linker).
DTRACE_DATA =   -zdtrace=dtrace_data
DYNFLAGS +=     $(DTRACE_DATA)

# DTrace needs an executable data segment.
MAPFILE.NED=

BUILD.s=        $(AS) $(ASFLAGS) $< -o $@

# Override this top level flag so the compiler builds in its native
# C99 mode.  This has been enabled to support the complex arithmetic
# added to libc.
C99MODE=        $(C99_ENABLE)

# libc method of building an archive
# The "$(GREP) -v ' L '" part is necessary only until
# lorder is fixed to ignore thread-local variables.
BUILD.AR= $(RM) $@ ; \
        $(AR) q $@ `$(LORDER) $(MOSTOBJS:%=$(DIR)/%) | $(GREP) -v ' L ' | $(TSOR

# extra files for the clean target
CLEANFILES=                     \
        $(LIBCDIR)/port/gen/errlst.c    \
        $(LIBCDIR)/port/gen/new_list.c  \
        assym.h                 \
        genassym                \
        $(LIBCBASE)/crt/_rtld.s         \
        $(LIBCBASE)/crt/_rtbootld.s             \
        pics/_rtbootld.o        \
        pics/crti.o             \
        pics/crtn.o             \
        $(ALTPICS)

CLOBBERFILES += $(LIB_PIC)

# list of C source for lint
SRCS=                                                   \
        $(ATOMICOBJS:%.o=$(SRC)/common/atomic/%.c)      \
        $(XATTROBJS:%.o=$(SRC)/common/xattr/%.c)        \
        $(COMOBJS:%.o=$(SRC)/common/util/%.c)           \
        $(DTRACEOBJS:%.o=$(SRC)/common/dtrace/%.c)      \
        $(PORTFP:%.o=$(LIBCDIR)/port/fp/%.c)                    \
        $(PORTGEN:%.o=$(LIBCDIR)/port/gen/%.c)                  \
        $(PORTI18N:%.o=$(LIBCDIR)/port/i18n/%.c)                \
        $(PORTLOCALE:%.o=$(LIBCDIR)/port/locale/%.c)            \
        $(PORTPRINT:%.o=$(LIBCDIR)/port/print/%.c)              \
        $(PORTREGEX:%.o=$(LIBCDIR)/port/regex/%.c)              \
        $(PORTSTDIO:%.o=$(LIBCDIR)/port/stdio/%.c)              \
        $(PORTSYS:%.o=$(LIBCDIR)/port/sys/%.c)                  \
        $(AIOOBJS:%.o=$(LIBCDIR)/port/aio/%.c)                  \
        $(RTOBJS:%.o=$(LIBCDIR)/port/rt/%.c)                    \
        $(TPOOLOBJS:%.o=$(LIBCDIR)/port/tpool/%.c)              \
        $(THREADSOBJS:%.o=$(LIBCDIR)/port/threads/%.c)          \
        $(THREADSMACHOBJS:%.o=$(LIBCDIR)/$(MACH)/threads/%.c)   \
        $(UNICODEOBJS:%.o=$(SRC)/common/unicode/%.c)    \
        $(UNWINDMACHOBJS:%.o=$(LIBCDIR)/port/unwind/%.c)        \
        $(FPOBJS:%.o=$(LIBCDIR)/$(MACH)/fp/%.c)                 \
```

```
1181         $(LIBCBASE)/crt/_ftou.c                        \
1182         $(LIBCBASE)/gen/_xregs_clrptr.c                \
1183         $(LIBCBASE)/gen/byteorder.c                    \
1184         $(LIBCBASE)/gen/ecvt.c                         \
1185         $(LIBCBASE)/gen/getctxt.c                      \
1186         $(LIBCBASE)/gen/lmul.c                         \
1187         $(LIBCBASE)/gen/makectxt.c                     \
1188         $(LIBCBASE)/gen/siginfolst.c                   \
1189         $(LIBCBASE)/gen/siglongjmp.c                   \
1190         $(LIBCBASE)/gen/swapctxt.c                     \
1191         $(LIBCBASE)/sys/ptrace.c                       \
1192         $(LIBCBASE)/sys/uadmin.c

1194 # conditional assignments
1195 $(DYNLIB) := CRTI = crti.o
1196 $(DYNLIB) := CRTN = crtn.o

1198 # Files which need the threads .il inline template
1199 TIL=                            \
1200         aio.o                  \
1201         alloc.o                \
1202         assfail.o              \
1203         atexit.o               \
1204         atfork.o               \
1205         cancel.o               \
1206         door_calls.o           \
1207         err.o                  \
1208         errno.o                \
1209         getctxt.o              \
1210         lwp.o                  \
1211         ma.o                   \
1212         machdep.o              \
1213         posix_aio.o            \
1214         pthr_attr.o            \
1215         pthr_barrier.o         \
1216         pthr_cond.o            \
1217         pthr_mutex.o           \
1218         pthr_rwlock.o          \
1219         pthread.o              \
1220         rand.o                 \
1221         rwlock.o               \
1222         scalls.o               \
1223         sched.o                \
1224         sema.o                 \
1225         sigaction.o            \
1226         sigev_thread.o         \
1227         spawn.o                \
1228         stack.o                \
1229         swapctxt.o             \
1230         synch.o                \
1231         tdb_agent.o            \
1232         thr.o                  \
1233         thread_interface.o     \
1234         thread_pool.o          \
1235         tls.o                  \
1236         tsd.o                  \
1237         unwind.o

1239 $(TIL:%=pics/%) := CFLAGS += $(LIBCBASE)/threads/sparc.il

1241 # special kludge for inlines with 'cas':
1242 pics/rwlock.o pics/synch.o pics/lwp.o pics/door_calls.o := \
1243         sparc_CFLAGS += -_gcc=-Wa,-xarch=v8plus

1245 # Files in port/fp subdirectory that need base.il inline template
1246 IL=                             \
```



```
1247         __flt_decim.o          \
1248         decimal_bin.o

1250 $(IL:%=pics/%) := CFLAGS += $(LIBCBASE)/fp/base.il

1252 # Files in fp subdirectory which need __quad.il inline template
1253 QIL=                            \
1254         _Q_add.o               \
1255         _Q_cmp.o               \
1256         _Q_cmpe.o              \
1257         _Q_div.o               \
1258         _Q_dtoq.o              \
1259         _Q_fcc.o               \
1260         _Q_mul.o               \
1261         _Q_qtod.o              \
1262         _Q_qtoi.o              \
1263         _Q_qtos.o              \
1264         _Q_qtou.o              \
1265         _Q_sqrt.o              \
1266         _Q_stoq.o              \
1267         _Q_sub.o

1269 $(QIL:%=pics/%) := CFLAGS += $(LIBCDIR)/$(MACH)/fp/__quad.il
1270 pics/_Q%.o := sparc_COPTFLAG = -xO4 -dalign
1271 pics/__quad%.o := sparc_COPTFLAG = -xO4 -dalign

1273 # large-file-aware components that should be built large

1275 $(COMSYSOBJS64:%=pics/%) := \
1276         CPPFLAGS += -D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64

1278 $(SYSOBJS64:%=pics/%) := \
1279         CPPFLAGS += -D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64

1281 $(PORTGEN64:%=pics/%) := \
1282         CPPFLAGS += -D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64

1284 $(PORTSTDIO64:%=pics/%) := \
1285         CPPFLAGS += -D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64

1287 $(PORTSYS64:%=pics/%) := \
1288         CPPFLAGS += -D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64

1290 $(PORTSTDIO_W:%=pics/%) := \
1291         CPPFLAGS += -D_WIDE

1293 $(PORTPRINT_W:%=pics/%) := \
1294         CPPFLAGS += -D_WIDE

1296 # printf/scanf functions to support c89-sized intmax_t variables
1297 $(PORTPRINT_C89:%=pics/%) := \
1298         CPPFLAGS += -D_C89_INTMAX32

1300 $(PORTSTDIO_C89:%=pics/%) := \
1301         CPPFLAGS += -D_C89_INTMAX32

1303 $(PORTI18N_COND:%=pics/%) := \
1304         CPPFLAGS += -D_WCS_LONGLONG

1306 pics/arc4random.o :=    CPPFLAGS += -I$(SRC)/common/crypto/chacha

1308 # Files which need extra optimization
1309 pics/getenv.o := sparc_COPTFLAG = -xO4

1311 .KEEP_STATE:
```

```
1313 all: $(LIBS) $(LIB_PIC)

1315 lint    :=      CPPFLAGS += -I$(LIBCDIR)/$(MACH)/fp
1316 lint    :=      CPPFLAGS += -D_MSE_INT_H -D_LCONV_C99
1317 lint    :=      LINTFLAGS += -mn

1319 lint:
1320         @echo $(LINT.c) ... $(LDLIBS)
1321         @$(LINT.c) $(SRCS) $(LDLIBS)

1323 $(LINTLIB):= SRCS=$(LIBCDIR)/port/llib-lc
1324 $(LINTLIB):= CPPFLAGS += -D_MSE_INT_H
1325 $(LINTLIB):= LINTFLAGS=-nvx

1327 # object files that depend on inline template
1328 $(TIL:%=pics/%): $(LIBCBASE)/threads/sparc.il
1329 $(IL:%=pics/%): $(LIBCBASE)/fp/base.il
1330 $(QIL:%=pics/%): $(LIBCDIR)/$(MACH)/fp/__quad.il

1332 # include common libc targets
1333 include $(LIBCDIR)/Makefile.targ

1335 # We need to strip out all CTF and DOF data from the static library
1336 $(LIB_PIC) := DIR = pics
1337 $(LIB_PIC): pics $$(PICS)
1338         $(BUILD.AR)
1339         $(MCS) -d -n .SUNW_ctf $@ > /dev/null 2>&1
1340         $(MCS) -d -n .SUNW_dof $@ > /dev/null 2>&1
1341         $(AR) -ts $@ > /dev/null
1342         $(POST_PROCESS_A)

1344 # special cases
1345 $(STRETS:%=pics/%): $(LIBCBASE)/crt/stret.s
1346         $(AS) $(ASFLAGS) -DSTRET$(@F:stret%.o=%) $(LIBCBASE)/crt/stret.s -o $@
1347         $(POST_PROCESS_O)

1349 $(LIBCBASE)/crt/_rtbootld.s:    $(LIBCBASE)/crt/_rtboot.s $(LIBCBASE)/crt/_rtld.
1350         $(CC) $(CPPFLAGS) $(CTF_FLAGS) -O -S -K pic \
1351             $(LIBCBASE)/crt/_rtld.c -o $(LIBCBASE)/crt/_rtld.s
1352         $(CAT) $(LIBCBASE)/crt/_rtboot.s $(LIBCBASE)/crt/_rtld.s > $@
1353         $(RM) $(LIBCBASE)/crt/_rtld.s

1355 # partially built from C source
1356 pics/_rtbootld.o: $(LIBCBASE)/crt/_rtbootld.s
1357         $(AS) $(ASFLAGS) $(LIBCBASE)/crt/_rtbootld.s -o $@
1358         $(CTFCONVERT_O)

1360 ASSYMDEP_OBJS=                  \
1361         _lwp_mutex_unlock.o     \
1362         _stack_grow.o           \
1363         asm_subr.o              \
1364         setjmp.o                \
1365         smt_pause.o             \
1366         tls_get_addr.o          \
1367         unwind_frame.o          \
1368         vforkx.o

1370 $(ASSYMDEP_OBJS:%=pics/%)       :=      CPPFLAGS += -I.

1372 $(ASSYMDEP_OBJS:%=pics/%): assym.h

1374 # assym.h build rules

1376 assym.h := CFLAGS += -g

1378 GENASSYM_C = $(LIBCDIR)/$(MACH)/genassym.c
```



```
1380 genassym: $(GENASSYM_C)
1381         $(NATIVECC) -I$(LIBCBASE)/inc -I$(LIBCDIR)/inc \
1382                 $(CPPFLAGS.native) -o $@ $(GENASSYM_C)

1384 OFFSETS = $(LIBCDIR)/$(MACH)/offsets.in

1386 assym.h: $(OFFSETS) genassym
1387         $(OFFSETS_CREATE) <$(OFFSETS) >$@
1388         ./genassym >>$@

1390 # derived C source and related explicit dependencies
1391 $(LIBCDIR)/port/gen/errlst.c + \
1392 $(LIBCDIR)/port/gen/new_list.c: $(LIBCDIR)/port/gen/errlist $(LIBCDIR)/port/gen/
1393         cd $(LIBCDIR)/port/gen; pwd; $(AWK) -f errlist.awk < errlist

1395 pics/errlst.o: $(LIBCDIR)/port/gen/errlst.c

1397 pics/new_list.o: $(LIBCDIR)/port/gen/new_list.c
```

```
**********************************************************
   24755 Wed May 27 19:49:14 2015
new/usr/src/lib/libc/sparcv9/Makefile.com
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
#
# Copyright (c) 1989, 2010, Oracle and/or its affiliates. All rights reserved.
# Copyright (c) 2015, Joyent, Inc.  All rights reserved.
# Copyright (c) 2013, OmniTI Computer Consulting, Inc. All rights reserved.
# Copyright 2013 Garrett D'Amore <garrett@damore.org>
#
# Copyright 2011 Nexenta Systems, Inc.  All rights reserved.
# Use is subject to license terms.
#

LIBCDIR=        $(SRC)/lib/libc
LIB_PIC=        libc_pic.a
VERS=           .1
CPP=            /usr/lib/cpp
TARGET_ARCH=    sparc

# objects are grouped by source directory

# Symbol capabilities objects.
EXTPICS=                        \
        $(LIBCDIR)/capabilities/sun4u/sparcv9/pics/symcap.o \
        $(LIBCDIR)/capabilities/sun4u-opl/sparcv9/pics/symcap.o \
        $(LIBCDIR)/capabilities/sun4u-us3-hwcap1/sparcv9/pics/symcap.o \
        $(LIBCDIR)/capabilities/sun4u-us3-hwcap2/sparcv9/pics/symcap.o \
        $(LIBCDIR)/capabilities/sun4v-hwcap1/sparcv9/pics/symcap.o \
        $(LIBCDIR)/capabilities/sun4v-hwcap2/sparcv9/pics/symcap.o

# local objects
STRETS=

CRTOBJS=                        \
        __align_cpy_2.o         \
        __align_cpy_4.o         \
        __align_cpy_8.o         \
        _ftou.o                 \
        cerror.o

DYNOBJS=
```



```
FPOBJS=                         \
        _D_cplx_div.o           \
        _D_cplx_div_ix.o        \
        _D_cplx_div_rx.o        \
        _D_cplx_mul.o           \
        _F_cplx_div.o           \
        _F_cplx_div_ix.o        \
        _F_cplx_div_rx.o        \
        _F_cplx_mul.o           \
        _Q_add.o                \
        _Q_cmp.o                \
        _Q_cmpe.o               \
        _Q_cplx_div.o           \
        _Q_cplx_div_ix.o        \
        _Q_cplx_div_rx.o        \
        _Q_cplx_lr_div.o        \
        _Q_cplx_lr_div_ix.o     \
        _Q_cplx_lr_div_rx.o     \
        _Q_cplx_lr_mul.o        \
        _Q_cplx_mul.o           \
        _Q_div.o                \
        _Q_dtoq.o               \
        _Q_fcc.o                \
        _Q_itoq.o               \
        _Q_mul.o                \
        _Q_neg.o                \
        _Q_qtod.o               \
        _Q_qtoi.o               \
        _Q_qtos.o               \
        _Q_qtou.o               \
        _Q_scl.o                \
        _Q_sqrt.o               \
        _Q_stoq.o               \
        _Q_sub.o                \
        _Q_utoq.o

FPOBJS64=                       \
        _Qp_qtox.o              \
        _Qp_qtoux.o             \
        _Qp_xtoq.o              \
        _Qp_uxtoq.o             \
        __dtoul.o               \
        __ftoul.o

FPASMOBJS=                      \
        _Q_get_rp_rd.o          \
        __quad_mag64.o          \
        fpgetmask.o             \
        fpgetrnd.o              \
        fpgetsticky.o           \
        fpsetmask.o             \
        fpsetrnd.o              \
        fpsetsticky.o

$(__GNUC)FPASMOBJS +=           \
        __quad.o

ATOMICOBJS=                     \
        atomic.o

CHACHAOBJS=                     \
        chacha.o

XATTROBJS=                      \
        xattr_common.o
```

```
126 COMOBJS=                        \
127         bcmp.o                  \
128         bcopy.o                 \
129         bsearch.o               \
130         bzero.o                 \
131         memccpy.o               \
132         qsort.o                 \
133         strtol.o                \
134         strtoul.o               \
135         strtoll.o               \
136         strtoull.o

138 GENOBJS=                        \
139         _getsp.o                \
140         _xregs_clrptr.o         \
141         abs.o                   \
142         alloca.o                \
143         arc4random.o            \
144         arc4random_uniform.o    \
145         ascii_strcasecmp.o      \
146         byteorder.o             \
147         cuexit.o                \
148         ecvt.o                  \
149         getctxt.o               \
150         lock.o                  \
151         makectxt.o              \
152         memchr.o                \
153         memcmp.o                \
154         new_list.o              \
155         setjmp.o                \
156         siginfolst.o            \
157         siglongjmp.o            \
158         smt_pause.o             \
159         sparc_data.o            \
160         strchr.o                \
161         strcmp.o                \
162         strlcpy.o               \
163         strncmp.o               \
164         strncpy.o               \
165         strnlen.o               \
166         swapctxt.o              \
167         sync_instruction_memory.o

169 # Preserved solely to ease maintenance of 32-bit and 64-bit library builds
170 # This macro should ALWAYS be empty; native APIs are already 'large file'.
171 COMSYSOBJS64=

173 SYSOBJS64=

175 COMSYSOBJS=                     \
176         __clock_timer.o         \
177         __getloadavg.o          \
178         __rusagesys.o           \
179         __signotify.o           \
180         __sigrt.o               \
181         __time.o                \
182         _lgrp_home_fast.o       \
183         _lgrpsys.o              \
184         _nfssys.o               \
185         _portfs.o               \
186         _pset.o                 \
187         _rpcsys.o               \
188         _sigaction.o            \
189         _so_accept.o            \
190         _so_bind.o              \
```



```
191         _so_connect.o           \
192         _so_getpeername.o       \
193         _so_getsockname.o       \
194         _so_getsockopt.o        \
195         _so_listen.o            \
196         _so_recv.o              \
197         _so_recvfrom.o          \
198         _so_recvmsg.o           \
199         _so_send.o              \
200         _so_sendmsg.o           \
201         _so_sendto.o            \
202         _so_setsockopt.o        \
203         _so_shutdown.o          \
204         _so_socket.o            \
205         _so_socketpair.o        \
206         _sockconfig.o           \
207         acct.o                  \
208         acl.o                   \
209         adjtime.o               \
210         alarm.o                 \
211         brk.o                   \
212         chdir.o                 \
213         chroot.o                \
214         cladm.o                 \
215         close.o                 \
216         execve.o                \
217         exit.o                  \
218         facl.o                  \
219         fchdir.o                \
220         fchroot.o               \
221         fdsync.o                \
222         fpathconf.o             \
223         fstatfs.o               \
224         fstatvfs.o              \
225         getcpuid.o              \
226         getdents.o              \
227         getegid.o               \
228         geteuid.o               \
229         getgid.o                \
230         getgroups.o             \
231         gethrtime.o             \
232         getitimer.o             \
233         getmsg.o                \
234         getpid.o                \
235         getpmsg.o               \
236         getppid.o               \
237         getrandom.o             \
238         getrlimit.o             \
239         getuid.o                \
240         gtty.o                  \
241         install_utrap.o         \
242         ioctl.o                 \
243         kaio.o                  \
244         kill.o                  \
245         llseek.o                \
246         lseek.o                 \
247         memcntl.o               \
248         mincore.o               \
249         mmap.o                  \
250         mmapobjsys.o            \
251         modctl.o                \
252         mount.o                 \
253         mprotect.o              \
254         munmap.o                \
255         nice.o                  \
256         ntp_adjtime.o           \
```

```
257         ntp_gettime.o           \
258         p_online.o              \
259         pathconf.o              \
260         pause.o                 \
261         pcsample.o              \
262         pipe2.o                 \
263         pollsys.o               \
264         pread.o                 \
265         preadv.o                \
266         priocntlset.o           \
267         processor_bind.o        \
268         processor_info.o        \
269         profil.o                \
270         psecflagsset.o          \
271 #endif /* ! codereview */
272         putmsg.o                \
273         putpmsg.o               \
274         pwrite.o                \
275         pwritev.o               \
276         read.o                  \
277         readv.o                 \
278         resolvepath.o           \
279         seteguid.o              \
280         setgid.o                \
281         setgroups.o             \
282         setitimer.o             \
283         setreid.o               \
284         setrlimit.o             \
285         setuid.o                \
286         sigaltstk.o             \
287         sigprocmsk.o            \
288         sigsendset.o            \
289         sigsuspend.o            \
290         statfs.o                \
291         statvfs.o               \
292         stty.o                  \
293         sync.o                  \
294         sysconfig.o             \
295         sysfs.o                 \
296         sysinfo.o               \
297         syslwp.o                \
298         times.o                 \
299         ulimit.o                \
300         umask.o                 \
301         umount2.o               \
302         utssys.o                \
303         uucopy.o                \
304         vhangup.o               \
305         waitid.o                \
306         write.o                 \
307         writev.o                \
308         yield.o

310 SYSOBJS=                       \
311         __clock_gettime.o       \
312         __getcontext.o          \
313         __uadmin.o              \
314         _lwp_mutex_unlock.o     \
315         _stack_grow.o           \
316         door.o                  \
317         forkx.o                 \
318         forkallx.o              \
319         gettimeofday.o          \
320         sparc_utrap_install.o   \
321         syscall.o               \
322         tls_get_addr.o          \
```



```
323         uadmin.o                \
324         umount.o                \
325         uname.o                 \
326         vforkx.o

328 # Preserved solely to ease maintenance of 32-bit and 64-bit library builds
329 # This macro should ALWAYS be empty; native APIs are already 'large file'.
330 PORTGEN64=

332 # objects from source under $(LIBCDIR)/port
333 PORTFP=                         \
334         __flt_decim.o           \
335         __flt_rounds.o          \
336         __tbl_10_b.o            \
337         __tbl_10_h.o            \
338         __tbl_10_s.o            \
339         __tbl_2_b.o             \
340         __tbl_2_h.o             \
341         __tbl_2_s.o             \
342         __tbl_fdq.o             \
343         __tbl_tens.o            \
344         __x_power.o             \
345         _base_sup.o             \
346         aconvert.o              \
347         decimal_bin.o           \
348         double_decim.o          \
349         econvert.o              \
350         fconvert.o              \
351         file_decim.o            \
352         finite.o                \
353         fp_data.o               \
354         func_decim.o            \
355         gconvert.o              \
356         hex_bin.o               \
357         ieee_globals.o          \
358         pack_float.o            \
359         sigfpe.o                \
360         string_decim.o

362 PORTGEN=                        \
363         _env_data.o             \
364         _xftw.o                 \
365         a64l.o                  \
366         abort.o                 \
367         addsev.o                \
368         ascii_strncasecmp.o     \
369         assert.o                \
370         attrat.o                \
371         atof.o                  \
372         atoi.o                  \
373         atol.o                  \
374         atoll.o                 \
375         attropen.o              \
376         atexit.o                \
377         atfork.o                \
378         basename.o              \
379         calloc.o                \
380         catgets.o               \
381         catopen.o               \
382         cfgetispeed.o           \
383         cfgetospeed.o           \
384         cfree.o                 \
385         cfsetispeed.o           \
386         cfsetospeed.o           \
387         cftime.o                \
388         clock.o                 \
```

```
389         closedir.o              \
390         closefrom.o             \
391         confstr.o               \
392         crypt.o                 \
393         csetlen.o               \
394         ctime.o                 \
395         ctime_r.o               \
396         daemon.o                \
397         deflt.o                 \
398         directio.o              \
399         dirname.o               \
400         div.o                   \
401         drand48.o               \
402         dup.o                   \
403         env_data.o              \
404         err.o                   \
405         errno.o                 \
406         euclen.o                \
407         event_port.o            \
408         execvp.o                \
409         explicit_bzero.o        \
410         fattach.o               \
411         fdetach.o               \
412         fdopendir.o             \
413         ffs.o                   \
414         fls.o                   \
415         fmtmsg.o                \
416         ftime.o                 \
417         ftok.o                  \
418         ftw.o                   \
419         gcvt.o                  \
420         getauxv.o               \
421         getcwd.o                \
422         getdate_err.o           \
423         getdtblsize.o           \
424         getentropy.o            \
425         getenv.o                \
426         getexecname.o           \
427         getgrnam.o              \
428         getgrnam_r.o            \
429         gethostid.o             \
430         gethostname.o           \
431         gethz.o                 \
432         getisax.o               \
433         getloadavg.o            \
434         getlogin.o              \
435         getmntent.o             \
436         getnetgrent.o           \
437         get_nprocs.o            \
438         getopt.o                \
439         getopt_long.o           \
440         getpagesize.o           \
441         getpw.o                 \
442         getpwnam.o              \
443         getpwnam_r.o            \
444         getrusage.o             \
445         getspent.o              \
446         getspent_r.o            \
447         getsubopt.o             \
448         gettxt.o                \
449         getusershell.o          \
450         getut.o                 \
451         getutx.o                \
452         getvfsent.o             \
453         getwd.o                 \
454         getwidth.o              \
```



```
455         getxby_door.o           \
456         gtxt.o                  \
457         hsearch.o               \
458         iconv.o                 \
459         imaxabs.o               \
460         imaxdiv.o               \
461         index.o                 \
462         initgroups.o            \
463         insque.o                \
464         isaexec.o               \
465         isastream.o             \
466         isatty.o                \
467         killpg.o                \
468         klpdlib.o               \
469         l64a.o                  \
470         lckpwdf.o               \
471         lconstants.o            \
472         ldivide.o               \
473         lexp10.o                \
474         lfind.o                 \
475         lfmt.o                  \
476         lfmt_log.o              \
477         lldiv.o                 \
478         llog10.o                \
479         lltostr.o               \
480         lmath.o                 \
481         localtime.o             \
482         lsearch.o               \
483         madvise.o               \
484         malloc.o                \
485         memalign.o              \
486         memmem.o                \
487         mkdev.o                 \
488         mkdtemp.o               \
489         mkfifo.o                \
490         mkstemp.o               \
491         mktemp.o                \
492         mlock.o                 \
493         mlockall.o              \
494         mon.o                   \
495         msync.o                 \
496         munlock.o               \
497         munlockall.o            \
498         ndbm.o                  \
499         nftw.o                  \
500         nlspath_checks.o        \
501         nsparse.o               \
502         nss_common.o            \
503         nss_dbdefs.o            \
504         nss_deffinder.o         \
505         opendir.o               \
506         opt_data.o              \
507         perror.o                \
508         pfmt.o                  \
509         pfmt_data.o             \
510         pfmt_print.o            \
511         pipe.o                  \
512         plock.o                 \
513         poll.o                  \
514         posix_fadvise.o         \
515         posix_fallocate.o       \
516         posix_madvise.o         \
517         posix_memalign.o        \
518         priocntl.o              \
519         privlib.o               \
520         priv_str_xlate.o        \
```

```
521         psecflags.o             \
522 #endif /* ! codereview */
523         psiginfo.o              \
524         psignal.o               \
525         pt.o                    \
526         putpwent.o              \
527         putspent.o              \
528         raise.o                 \
529         rand.o                  \
530         random.o                \
531         rctlops.o               \
532         readdir.o               \
533         readdir_r.o             \
534         realpath.o              \
535         reboot.o                \
536         regexpr.o               \
537         remove.o                \
538         rewinddir.o             \
539         rindex.o                \
540         scandir.o               \
541         seekdir.o               \
542         select.o                \
543         setlabel.o              \
544         setpriority.o           \
545         settimeofday.o          \
546         sh_locks.o              \
547         sigflag.o               \
548         siglist.o               \
549         sigsend.o               \
550         sigsetops.o             \
551         ssignal.o               \
552         stack.o                 \
553         stpcpy.o                \
554         stpncpy.o               \
555         str2sig.o               \
556         strcase_charmap.o       \
557         strcat.o                \
558         strchrnul.o             \
559         strcspn.o               \
560         strdup.o                \
561         strerror.o              \
562         strlcat.o               \
563         strncat.o               \
564         strndup.o               \
565         strpbrk.o               \
566         strrchr.o               \
567         strsep.o                \
568         strsignal.o             \
569         strspn.o                \
570         strstr.o                \
571         strtod.o                \
572         strtoimax.o             \
573         strtok.o                \
574         strtok_r.o              \
575         strtoumax.o             \
576         swab.o                  \
577         swapctl.o               \
578         sysconf.o               \
579         syslog.o                \
580         tcdrain.o               \
581         tcflow.o                \
582         tcflush.o               \
583         tcgetattr.o             \
584         tcgetpgrp.o             \
585         tcgetsid.o              \
586         tcsendbreak.o           \
```



```
587         tcsetattr.o             \
588         tcsetpgrp.o             \
589         tell.o                  \
590         telldir.o               \
591         tfind.o                 \
592         time_data.o             \
593         time_gdata.o            \
594         tls_data.o              \
595         truncate.o              \
596         tsdalloc.o              \
597         tsearch.o               \
598         ttyname.o               \
599         ttyslot.o               \
600         ualarm.o                \
601         ucred.o                 \
602         valloc.o                \
603         vlfmt.o                 \
604         vpfmt.o                 \
605         waitpid.o               \
606         walkstack.o             \
607         wdata.o                 \
608         xgetwidth.o             \
609         xpg4.o                  \
610         xpg6.o

612 PORTPRINT_W=                    \
613         doprnt_w.o

615 PORTPRINT=                      \
616         asprintf.o              \
617         doprnt.o                \
618         fprintf.o               \
619         printf.o                \
620         snprintf.o              \
621         sprintf.o               \
622         vfprintf.o              \
623         vprintf.o               \
624         vsnprintf.o             \
625         vsprintf.o              \
626         vwprintf.o              \
627         wprintf.o

629 # Preserved solely to ease maintenance of 32-bit and 64-bit library builds
630 # This macro should ALWAYS be empty; native APIs are already 'large file'.
631 PORTSTDIO64=

633 PORTSTDIO_W=                    \
634         doscan_w.o

636 PORTSTDIO=                      \
637         __extensions.o          \
638         _endopen.o              \
639         _filbuf.o               \
640         _findbuf.o              \
641         _flsbuf.o               \
642         _wrtchk.o               \
643         clearerr.o              \
644         ctermid.o               \
645         ctermid_r.o             \
646         cuserid.o               \
647         data.o                  \
648         doscan.o                \
649         fdopen.o                \
650         feof.o                  \
651         ferror.o                \
652         fgetc.o                 \
```

```
653         fgets.o                 \
654         fileno.o                \
655         flockf.o                \
656         flush.o                 \
657         fopen.o                 \
658         fpos.o                  \
659         fputc.o                 \
660         fputs.o                 \
661         fread.o                 \
662         fseek.o                 \
663         fseeko.o                \
664         ftell.o                 \
665         ftello.o                \
666         fwrite.o                \
667         getc.o                  \
668         getchar.o               \
669         getline.o               \
670         getpass.o               \
671         gets.o                  \
672         getw.o                  \
673         popen.o                 \
674         putc.o                  \
675         putchar.o               \
676         puts.o                  \
677         putw.o                  \
678         rewind.o                \
679         scanf.o                 \
680         setbuf.o                \
681         setbuffer.o             \
682         setvbuf.o               \
683         system.o                \
684         tempnam.o               \
685         tmpfile.o               \
686         tmpnam_r.o              \
687         ungetc.o                \
688         mse.o                   \
689         vscanf.o                \
690         vwscanf.o               \
691         wscanf.o

693 PORTI18N=                       \
694         getwchar.o              \
695         putwchar.o              \
696         putws.o                 \
697         strtows.o               \
698         wcsnlen.o               \
699         wcstoimax.o             \
700         wcstol.o                \
701         wcstoul.o               \
702         wcswcs.o                \
703         wscat.o                 \
704         wschr.o                 \
705         wscmp.o                 \
706         wscpy.o                 \
707         wscspn.o                \
708         wsdup.o                 \
709         wslen.o                 \
710         wsncat.o                \
711         wsncmp.o                \
712         wsncpy.o                \
713         wspbrk.o                \
714         wsprintf.o              \
715         wsrchr.o                \
716         wsscanf.o               \
717         wsspn.o                 \
718         wstod.o                 \
```



```
719         wstok.o                 \
720         wstol.o                 \
721         wstoll.o                \
722         wsxfrm.o                \
723         wmemchr.o               \
724         wmemcmp.o               \
725         wmemcpy.o               \
726         wmemmove.o              \
727         wmemset.o               \
728         wcsstr.o                \
729         gettext.o               \
730         gettext_real.o          \
731         gettext_util.o          \
732         gettext_gnu.o           \
733         plural_parser.o         \
734         wdresolve.o             \
735         _ctype.o                \
736         isascii.o               \
737         toascii.o

739 PORTI18N_COND=                  \
740         wcstol_longlong.o       \
741         wcstoul_longlong.o

743 PORTLOCALE=                    \
744         big5.o                  \
745         btowc.o                 \
746         collate.o               \
747         collcmp.o               \
748         euc.o                   \
749         fnmatch.o               \
750         fgetwc.o                \
751         fgetws.o                \
752         fix_grouping.o          \
753         fputwc.o                \
754         fputws.o                \
755         fwide.o                 \
756         gb18030.o               \
757         gb2312.o                \
758         gbk.o                   \
759         getdate.o               \
760         isdigit.o               \
761         iswctype.o              \
762         ldpart.o                \
763         lmessages.o             \
764         lnumeric.o              \
765         lmonetary.o             \
766         localeconv.o            \
767         localeimpl.o            \
768         mbftowc.o               \
769         mblen.o                 \
770         mbrlen.o                \
771         mbrtowc.o               \
772         mbsinit.o               \
773         mbsnrtowcs.o            \
774         mbsrtowcs.o             \
775         mbstowcs.o              \
776         mbtowc.o                \
777         mskanji.o               \
778         nextwctype.o            \
779         nl_langinfo.o           \
780         none.o                  \
781         regcomp.o               \
782         regfree.o               \
783         regerror.o              \
784         regexec.o               \
```

```
785         rune.o                  \
786         runetype.o              \
787         setlocale.o             \
788         setrunelocale.o         \
789         strcasecmp.o            \
790         strcasestr.o            \
791         strcoll.o               \
792         strfmon.o               \
793         strftime.o              \
794         strncasecmp.o           \
795         strptime.o              \
796         strxfrm.o               \
797         table.o                 \
798         timelocal.o             \
799         tolower.o               \
800         towlower.o              \
801         ungetwc.o               \
802         utf8.o                  \
803         wcrtomb.o               \
804         wcscasecmp.o            \
805         wcscoll.o               \
806         wcsftime.o              \
807         wcsnrtombs.o            \
808         wcsrtombs.o             \
809         wcstombs.o              \
810         wcswidth.o              \
811         wcsxfrm.o               \
812         wctob.o                 \
813         wctomb.o                \
814         wctrans.o               \
815         wctype.o                \
816         wcwidth.o               \
817         wscol.o

819 AIOOBJS=                       \
820         aio.o                   \
821         aio_alloc.o             \
822         posix_aio.o

824 RTOBJS=                        \
825         clock_timer.o           \
826         mqueue.o                \
827         pos4obj.o               \
828         sched.o                 \
829         sem.o                   \
830         shm.o                   \
831         sigev_thread.o

833 TPOOLOBJS=                     \
834         thread_pool.o

836 THREADSOBJS=                   \
837         alloc.o                 \
838         assfail.o               \
839         cancel.o                \
840         door_calls.o            \
841         tmem.o                  \
842         pthr_attr.o             \
843         pthr_barrier.o          \
844         pthr_cond.o             \
845         pthr_mutex.o            \
846         pthr_rwlock.o           \
847         pthread.o               \
848         rwlock.o                \
849         scalls.o                \
850         sema.o                  \
```



```
851         sigaction.o             \
852         spawn.o                 \
853         synch.o                 \
854         tdb_agent.o             \
855         thr.o                   \
856         thread_interface.o      \
857         tls.o                   \
858         tsd.o

860 THREADSMACHOBJS=               \
861         machdep.o

863 THREADSASMOBJS=                \
864         asm_subr.o

866 UNICODEOBJS=                   \
867         u8_textprep.o           \
868         uconv.o

870 UNWINDMACHOBJS=                \
871         unwind.o

873 UNWINDASMOBJS=                 \
874         unwind_frame.o

876 # Preserved solely to ease maintenance of 32-bit and 64-bit library builds
877 # This macro should ALWAYS be empty; native APIs are already 'large file'.
878 PORTSYS64=

880 PORTSYS=                        \
881         _autofssys.o            \
882         access.o                \
883         acctctl.o               \
884         bsd_signal.o            \
885         chmod.o                 \
886         chown.o                 \
887         corectl.o               \
888         exacctsys.o             \
889         execl.o                 \
890         execle.o                \
891         execv.o                 \
892         fcntl.o                 \
893         getpagesizes.o          \
894         getpeerucred.o          \
895         inst_sync.o             \
896         issetugid.o             \
897         label.o                 \
898         link.o                  \
899         lockf.o                 \
900         lwp.o                   \
901         lwp_cond.o              \
902         lwp_rwlock.o            \
903         lwp_sigmask.o           \
904         meminfosys.o            \
905         mkdir.o                 \
906         mknod.o                 \
907         msgsys.o                \
908         nfssys.o                \
909         open.o                  \
910         pgrpsys.o               \
911         posix_sigwait.o         \
912         ppriv.o                 \
913         psetsys.o               \
914         rctlsys.o               \
915         readlink.o              \
916         rename.o                \
```

```
917         sbrk.o                  \
918         semsys.o                \
919         set_errno.o             \
920         sharefs.o               \
921         shmsys.o                \
922         sidsys.o                \
923         siginterrupt.o          \
924         signal.o                \
925         sigpending.o            \
926         sigstack.o              \
927         stat.o                  \
928         symlink.o               \
929         tasksys.o               \
930         time.o                  \
931         time_util.o             \
932         ucontext.o              \
933         unlink.o                \
934         ustat.o                 \
935         utimesys.o              \
936         zone.o

938 PORTREGEX=                      \
939         glob.o                  \
940         regcmp.o                \
941         regex.o                 \
942         wordexp.o

944 VALUES= values-Xa.o

946 MOSTOBJS=                       \
947         $(STRETS)               \
948         $(CRTOBJS)              \
949         $(DYNOBJS)              \
950         $(FPOBJS)               \
951         $(FPOBJS64)             \
952         $(FPASMOBJS)            \
953         $(ATOMICOBJS)           \
954         $(CHACHAOBJS)           \
955         $(XATTROBJS)            \
956         $(COMOBJS)              \
957         $(GENOBJS)              \
958         $(PRFOBJS)              \
959         $(PORTFP)               \
960         $(PORTGEN)              \
961         $(PORTGEN64)            \
962         $(PORTI18N)             \
963         $(PORTI18N_COND)        \
964         $(PORTLOCALE)           \
965         $(PORTPRINT)            \
966         $(PORTPRINT_W)          \
967         $(PORTREGEX)            \
968         $(PORTSTDIO)            \
969         $(PORTSTDIO64)          \
970         $(PORTSTDIO_W)          \
971         $(PORTSYS)              \
972         $(PORTSYS64)            \
973         $(AIOOBJS)              \
974         $(RTOBJS)               \
975         $(TPOOLOBJS)            \
976         $(THREADSOBJS)          \
977         $(THREADSMACHOBJS)      \
978         $(THREADSASMOBJS)       \
979         $(UNICODEOBJS)          \
980         $(UNWINDMACHOBJS)       \
981         $(UNWINDASMOBJS)        \
982         $(COMSYSOBJS)           \
```



```
983         $(SYSOBJS)              \
984         $(COMSYSOBJS64)         \
985         $(SYSOBJS64)            \
986         $(VALUES)

988 TRACEOBJS=                      \
989         plockstat.o

991 # NOTE: libc.so.1 must be linked with the minimal crti.o and crtn.o
992 # modules whose source is provided in the $(SRC)/lib/common directory.
993 # This must be done because otherwise the Sun C compiler would insert
994 # its own versions of these modules and those versions contain code
995 # to call out to C++ initialization functions.  Such C++ initialization
996 # functions can call back into libc before thread initialization is
997 # complete and this leads to segmentation violations and other problems.
998 # Since libc contains no C++ code, linking with the minimal crti.o and
999 # crtn.o modules is safe and avoids the problems described above.
1000 OBJECTS= $(CRTI) $(MOSTOBJS) $(CRTN)
1001 CRTSRCS= ../../common/sparcv9

1003 # include common library definitions
1004 include $(SRC)/lib/Makefile.lib
1005 include $(SRC)/lib/Makefile.lib.64

1007 # we need to override the default SONAME here because we might
1008 # be building a variant object (still libc.so.1, but different filename)
1009 SONAME = libc.so.1

1011 CFLAGS64 += $(CCVERBOSE)

1013 # This is necessary to avoid problems with calling _ex_unwind().
1014 # We probably don't want any inlining anyway.
1015 CFLAGS64 += -xinline=

1017 CERRWARN += -_gcc=-Wno-parentheses
1018 CERRWARN += -_gcc=-Wno-switch
1019 CERRWARN += -_gcc=-Wno-uninitialized
1020 CERRWARN += -_gcc=-Wno-unused-value
1021 CERRWARN += -_gcc=-Wno-unused-label
1022 CERRWARN += -_gcc=-Wno-unused-variable
1023 CERRWARN += -_gcc=-Wno-type-limits
1024 CERRWARN += -_gcc=-Wno-char-subscripts
1025 CERRWARN += -_gcc=-Wno-clobbered
1026 CERRWARN += -_gcc=-Wno-unused-function
1027 CERRWARN += -_gcc=-Wno-address

1029 # Setting THREAD_DEBUG = -DTHREAD_DEBUG (make THREAD_DEBUG=-DTHREAD_DEBUG ...)
1030 # enables ASSERT() checking in the threads portion of the library.
1031 # This is automatically enabled for DEBUG builds, not for non-debug builds.
1032 THREAD_DEBUG =
1033 $(NOT_RELEASE_BUILD)THREAD_DEBUG = -DTHREAD_DEBUG

1035 # Make string literals read-only to save memory.
1036 CFLAGS64 += $(XSTRCONST)

1038 ALTPICS= $(TRACEOBJS:%=pics/%)

1040 $(DYNLIB) := BUILD.SO = $(LD) -o $@ -G $(DYNFLAGS) $(PICS) $(ALTPICS) $(EXTPICS)

1042 MAPFILES =      $(LIBCDIR)/port/mapfile-vers

1044 sparcv9_C_PICFLAGS= -K PIC
1045 CFLAGS64 +=     $(EXTN_CFLAGS)
1046 CPPFLAGS=       -D_REENTRANT -Dsparc $(EXTN_CPPFLAGS) $(THREAD_DEBUG) \
1047                 -I$(LIBCBASE)/inc -I$(LIBCDIR)/inc $(CPPFLAGS.master)
1048 ASFLAGS=        $(EXTN_ASFLAGS) -K PIC -P -D__STDC__ -D_ASM -D__sparcv9 $(CPPFLA
```

```
1049                $(sparcv9_AS_XARCH)

1051 # As a favor to the dtrace syscall provider, libc still calls the
1052 # old syscall traps that have been obsoleted by the *at() interfaces.
1053 # Delete this to compile libc using only the new *at() system call traps
1054 CPPFLAGS += -D_RETAIN_OLD_SYSCALLS

1056 # Inform the run-time linker about libc specialized initialization
1057 RTLDINFO =      -z rtldinfo=tls_rtldinfo
1058 DYNFLAGS +=     $(RTLDINFO)

1060 # Force libc's internal references to be resolved immediately upon loading
1061 # in order to avoid critical region problems.  Since almost all libc symbols
1062 # are marked 'protected' in the mapfiles, this is a minimal set (15 to 20).
1063 DYNFLAGS +=     -znow

1065 DYNFLAGS +=     $(EXTN_DYNFLAGS)

1067 BUILD.s=        $(AS) $(ASFLAGS) $< -o $@

1069 # Override this top level flag so the compiler builds in its native
1070 # C99 mode.  This has been enabled to support the complex arithmetic
1071 # added to libc.
1072 C99MODE=        $(C99_ENABLE)

1074 # libc method of building an archive
1075 # The "$(GREP) -v ' L '" part is necessary only until
1076 # lorder is fixed to ignore thread-local variables.
1077 BUILD.AR= $(RM) $@ ; \
1078         $(AR) q $@ `$(LORDER) $(MOSTOBJS:%=$(DIR)/%) | $(GREP) -v ' L ' | $(TSOR

1080 # extra files for the clean target
1081 CLEANFILES=                     \
1082         $(LIBCDIR)/port/gen/errlst.c    \
1083         $(LIBCDIR)/port/gen/new_list.c  \
1084         assym.h                 \
1085         genassym                \
1086         pics/crti.o             \
1087         pics/crtn.o             \
1088         $(ALTPICS)

1090 CLOBBERFILES += $(LIB_PIC)

1092 # list of C source for lint
1093 SRCS=                                           \
1094         $(ATOMICOBJS:%.o=$(SRC)/common/atomic/%.c)      \
1095         $(XATTROBJS:%.o=$(SRC)/common/xattr/%.c)        \
1096         $(COMOBJS:%.o=$(SRC)/common/util/%.c)           \
1097         $(PORTFP:%.o=$(LIBCDIR)/port/fp/%.c)                    \
1098         $(PORTGEN:%.o=$(LIBCDIR)/port/gen/%.c)                  \
1099         $(PORTI18N:%.o=$(LIBCDIR)/port/i18n/%.c)                \
1100         $(PORTLOCALE:%.o=$(LIBCDIR)/port/locale/%.c)            \
1101         $(PORTPRINT:%.o=$(LIBCDIR)/port/print/%.c)              \
1102         $(PORTREGEX:%.o=$(LIBCDIR)/port/regex/%.c)              \
1103         $(PORTSTDIO:%.o=$(LIBCDIR)/port/stdio/%.c)              \
1104         $(PORTSYS:%.o=$(LIBCDIR)/port/sys/%.c)                  \
1105         $(AIOOBJS:%.o=$(LIBCDIR)/port/aio/%.c)                  \
1106         $(RTOBJS:%.o=$(LIBCDIR)/port/rt/%.c)                    \
1107         $(TPOOLOBJS:%.o=$(LIBCDIR)/port/tpool/%.c)              \
1108         $(THREADSOBJS:%.o=$(LIBCDIR)/port/threads/%.c)          \
1109         $(THREADSMACHOBJS:%.o=$(LIBCDIR)/$(MACH)/threads/%.c)   \
1110         $(UNICODEOBJS:%.o=$(SRC)/common/unicode/%.c)    \
1111         $(UNWINDMACHOBJS:%.o=$(LIBCDIR)/port/unwind/%.c)        \
1112         $(FPOBJS:%.o=$(LIBCDIR)/$(MACH)/fp/%.c)                 \
1113         $(FPOBJS64:%.o=$(LIBCBASE)/fp/%.c)              \
1114         $(LIBCBASE)/crt/_ftou.c                         \
```



```
1115         $(LIBCBASE)/gen/_xregs_clrptr.c                 \
1116         $(LIBCBASE)/gen/byteorder.c                     \
1117         $(LIBCBASE)/gen/ecvt.c                          \
1118         $(LIBCBASE)/gen/getctxt.c                       \
1119         $(LIBCBASE)/gen/makectxt.c                      \
1120         $(LIBCBASE)/gen/siginfolst.c                    \
1121         $(LIBCBASE)/gen/siglongjmp.c                    \
1122         $(LIBCBASE)/gen/swapctxt.c

1124 # conditional assignments
1125 $(DYNLIB) := CRTI = crti.o
1126 $(DYNLIB) := CRTN = crtn.o

1128 # Files which need the threads .il inline template
1129 TIL=                            \
1130         aio.o                   \
1131         alloc.o                 \
1132         assfail.o               \
1133         atexit.o                \
1134         atfork.o                \
1135         cancel.o                \
1136         door_calls.o            \
1137         err.o                   \
1138         errno.o                 \
1139         getctxt.o               \
1140         lwp.o                   \
1141         ma.o                    \
1142         machdep.o               \
1143         posix_aio.o             \
1144         pthr_attr.o             \
1145         pthr_barrier.o          \
1146         pthr_cond.o             \
1147         pthr_mutex.o            \
1148         pthr_rwlock.o           \
1149         pthread.o               \
1150         rand.o                  \
1151         rwlock.o                \
1152         scalls.o                \
1153         sched.o                 \
1154         sema.o                  \
1155         sigaction.o             \
1156         sigev_thread.o          \
1157         spawn.o                 \
1158         stack.o                 \
1159         swapctxt.o              \
1160         synch.o                 \
1161         tdb_agent.o             \
1162         thr.o                   \
1163         thread_interface.o      \
1164         thread_pool.o           \
1165         tls.o                   \
1166         tsd.o                   \
1167         unwind.o

1169 $(TIL:%=pics/%) := CFLAGS64 += $(LIBCBASE)/threads/sparcv9.il

1171 # Files in fp, port/fp subdirectories that need base.il inline template
1172 IL=                             \
1173         __flt_decim.o           \
1174         decimal_bin.o

1176 $(IL:%=pics/%) := CFLAGS64 += $(LIBCBASE)/fp/base.il

1178 # Files in fp subdirectory which need __quad.il inline template
1179 QIL=                            \
1180         _Q_add.o                \
```

```
1181         _Q_cmp.o                \
1182         _Q_cmpe.o               \
1183         _Q_div.o                \
1184         _Q_dtoq.o               \
1185         _Q_fcc.o                \
1186         _Q_mul.o                \
1187         _Q_qtod.o               \
1188         _Q_qtoi.o               \
1189         _Q_qtos.o               \
1190         _Q_qtou.o               \
1191         _Q_sqrt.o               \
1192         _Q_stoq.o               \
1193         _Q_sub.o                \
1194         _Qp_qtox.o              \
1195         _Qp_qtoux.o

1197 $(QIL:%=pics/%) := CFLAGS64 += $(LIBCDIR)/$(MACH)/fp/__quad.il
1198 pics/_Qp%.o := CFLAGS64 += -I$(LIBCDIR)/$(MACH)/fp
1199 pics/_Q%.o := sparcv9_COPTFLAG = -xO4 -xchip=ultra

1201 # Files in crt subdirectory which need muldiv64.il inline template
1202 #CIL=   mul64.o divrem64.o
1203 #$(CIL:%=pics/%) := CFLAGS += $(LIBCBASE)/crt/mul64.il

1205 # large-file-aware components that should be built large

1207 #$(COMSYSOBJS64:%=pics/%) := \
1208 #       CPPFLAGS += -D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64

1210 #$(SYSOBJS64:%=pics/%) := \
1211 #       CPPFLAGS += -D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64

1213 #$(PORTGEN64:%=pics/%) := \
1214 #       CPPFLAGS += -D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64

1216 #$(PORTSTDIO64:%=pics/%) := \
1217 #       CPPFLAGS += -D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64

1219 #$(PORTSYS64:%=pics/%) := \
1220 #       CPPFLAGS += -D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64

1222 $(PORTSTDIO_W:%=pics/%) := \
1223         CPPFLAGS += -D_WIDE

1225 $(PORTPRINT_W:%=pics/%) := \
1226         CPPFLAGS += -D_WIDE

1228 $(PORTI18N_COND:%=pics/%) := \
1229         CPPFLAGS += -D_WCS_LONGLONG

1231 pics/arc4random.o :=    CPPFLAGS += -I$(SRC)/common/crypto/chacha

1233 # Files which need extra optimization
1234 pics/getenv.o := sparcv9_COPTFLAG = -xO4

1236 .KEEP_STATE:

1238 all: $(LIBS) $(LIB_PIC)

1240 lint    :=      CPPFLAGS += -I$(LIBCDIR)/$(MACH)/fp
1241 lint    :=      CPPFLAGS += -D_MSE_INT_H -D_LCONV_C99
1242 lint    :=      LINTFLAGS64 += -mn

1244 lint:
1245         @echo $(LINT.c) ... $(LDLIBS)
1246         @$(LINT.c) $(SRCS) $(LDLIBS)
```



```
1248 $(LINTLIB):= SRCS=$(LIBCDIR)/port/llib-lc
1249 $(LINTLIB):= CPPFLAGS += -D_MSE_INT_H
1250 $(LINTLIB):= LINTFLAGS64=-nvx -m64

1252 # object files that depend on inline template
1253 $(TIL:%=pics/%): $(LIBCBASE)/threads/sparcv9.il
1254 $(IL:%=pics/%): $(LIBCBASE)/fp/base.il
1255 $(QIL:%=pics/%): $(LIBCDIR)/$(MACH)/fp/__quad.il
1256 #$(CIL:%=pics/%): $(LIBCBASE)/crt/muldiv64.il

1258 # include common libc targets
1259 include $(LIBCDIR)/Makefile.targ

1261 # We need to strip out all CTF and DOF data from the static library
1262 $(LIB_PIC) := DIR = pics
1263 $(LIB_PIC): pics $$(PICS)
1264         $(BUILD.AR)
1265         $(MCS) -d -n .SUNW_ctf $@ > /dev/null 2>&1
1266         $(MCS) -d -n .SUNW_dof $@ > /dev/null 2>&1
1267         $(AR) -ts $@ > /dev/null
1268         $(POST_PROCESS_A)

1270 # special cases
1271 #$(STRETS:%=pics/%): crt/stret.s
1272 #       $(AS) $(ASFLAGS) -DSTRET$(@F:stret%.o=%) crt/stret.s -o $@
1273 #       $(POST_PROCESS_O)

1275 #crt/_rtbootld.s:       crt/_rtboot.s crt/_rtld.c
1276 #       $(CC) $(CPPFLAGS) -O -S -K pic crt/_rtld.c -o crt/_rtld.s
1277 #       $(CAT) crt/_rtboot.s crt/_rtld.s > $@
1278 #       $(RM) crt/_rtld.s

1280 ASSYMDEP_OBJS=                  \
1281         _lwp_mutex_unlock.o     \
1282         _stack_grow.o           \
1283         asm_subr.o              \
1284         setjmp.o                \
1285         smt_pause.o             \
1286         tls_get_addr.o          \
1287         unwind_frame.o          \
1288         vforkx.o

1290 $(ASSYMDEP_OBJS:%=pics/%)       :=      CPPFLAGS += -I.

1292 $(ASSYMDEP_OBJS:%=pics/%): assym.h

1294 # assym.h build rules

1296 assym.h := CFLAGS64 += -g

1298 GENASSYM_C = $(LIBCDIR)/$(MACH)/genassym.c

1300 genassym: $(GENASSYM_C)
1301         $(NATIVECC) -I$(LIBCBASE)/inc -I$(LIBCDIR)/inc \
1302                 $(CPPFLAGS.native) -o $@ $(GENASSYM_C)

1304 OFFSETS = $(LIBCDIR)/$(MACH)/offsets.in

1306 assym.h: $(OFFSETS) genassym
1307         $(OFFSETS_CREATE) <$(OFFSETS) >$@
1308         ./genassym >>$@

1310 # derived C source and related explicit dependencies
1311 $(LIBCDIR)/port/gen/new_list.c: $(LIBCDIR)/port/gen/errlist $(LIBCDIR)/port/gen/
1312         cd $(LIBCDIR)/port/gen; pwd; $(AWK) -f errlist.awk < errlist
```

```
1314 pics/new_list.o: $(LIBCDIR)/port/gen/new_list.c
```

```
**********************************************************
   22181 Wed May 27 19:49:15 2015
new/usr/src/lib/libproc/common/P32ton.c
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright 2006 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

#pragma ident   "%Z%%M% %I%     %E% SMI"

#include <sys/types.h>
#include <sys/mkdev.h>
#include <sys/regset.h>
#include <string.h>

#if defined(__amd64)
#include <sys/fp.h>
#include <ieeefp.h>
#endif

#include "P32ton.h"

dev_t
prexpldev(dev32_t d)
{
        if (d != (dev32_t)-1L)
                return (makedev((d >> NBITSMINOR32) & MAXMAJ32, d & MAXMIN32));

        return ((dev_t)PRNODEV);
}
```

*unchanged_portion_omitted_*

```
void
pstatus_32_to_n(const pstatus32_t *src, pstatus_t *dst)
{
        dst->pr_flags = src->pr_flags;
        dst->pr_nlwp = src->pr_nlwp;
        dst->pr_nzomb = src->pr_nzomb;
        dst->pr_pid = src->pr_pid;
        dst->pr_ppid = src->pr_ppid;
```



```
        dst->pr_pgid = src->pr_pgid;
        dst->pr_sid = src->pr_sid;
        dst->pr_taskid = src->pr_taskid;
        dst->pr_projid = src->pr_projid;
        dst->pr_zoneid = src->pr_zoneid;
        dst->pr_aslwpid = src->pr_aslwpid;
        dst->pr_agentid = src->pr_agentid;
        dst->pr_sigpend = src->pr_sigpend;
        dst->pr_brkbase = src->pr_brkbase;
        dst->pr_brksize = src->pr_brksize;
        dst->pr_stkbase = src->pr_stkbase;
        dst->pr_stksize = src->pr_stksize;

        timestruc_32_to_n(&src->pr_utime, &dst->pr_utime);
        timestruc_32_to_n(&src->pr_stime, &dst->pr_stime);
        timestruc_32_to_n(&src->pr_cutime, &dst->pr_cutime);
        timestruc_32_to_n(&src->pr_cstime, &dst->pr_cstime);

        dst->pr_sigtrace = src->pr_sigtrace;
        dst->pr_flttrace = src->pr_flttrace;
        dst->pr_sysentry = src->pr_sysentry;
        dst->pr_sysexit = src->pr_sysexit;
        dst->pr_dmodel = src->pr_dmodel;

        (void) memcpy(&dst->pr_secflags, &src->pr_secflags, sizeof (psecflags_t)

#endif /* ! codereview */
        lwpstatus_32_to_n(&src->pr_lwp, &dst->pr_lwp);
}

void
lwpsinfo_32_to_n(const lwpsinfo32_t *src, lwpsinfo_t *dst)
{
        dst->pr_flag = src->pr_flag;
        dst->pr_lwpid = src->pr_lwpid;
        dst->pr_addr = src->pr_addr;
        dst->pr_wchan = src->pr_wchan;
        dst->pr_stype = src->pr_stype;
        dst->pr_state = src->pr_state;
        dst->pr_sname = src->pr_sname;
        dst->pr_nice = src->pr_nice;
        dst->pr_syscall = src->pr_syscall;
        dst->pr_oldpri = src->pr_oldpri;
        dst->pr_cpu = src->pr_cpu;
        dst->pr_pri = src->pr_pri;
        dst->pr_pctcpu = src->pr_pctcpu;

        timestruc_32_to_n(&src->pr_start, &dst->pr_start);
        timestruc_32_to_n(&src->pr_time, &dst->pr_time);

        (void) memcpy(&dst->pr_clname[0], &src->pr_clname[0], PRCLSZ);
        (void) memcpy(&dst->pr_name[0], &src->pr_name[0], PRFNSZ);

        dst->pr_onpro = src->pr_onpro;
        dst->pr_bindpro = src->pr_bindpro;
        dst->pr_bindpset = src->pr_bindpset;
        dst->pr_lgrp = src->pr_lgrp;
}

void
psinfo_32_to_n(const psinfo32_t *src, psinfo_t *dst)
{
        dst->pr_flag = src->pr_flag;
        dst->pr_nlwp = src->pr_nlwp;
        dst->pr_nzomb = src->pr_nzomb;
        dst->pr_pid = src->pr_pid;
```

```
414         dst->pr_pgid = src->pr_pgid;
415         dst->pr_sid = src->pr_sid;
416         dst->pr_taskid = src->pr_taskid;
417         dst->pr_projid = src->pr_projid;
418         dst->pr_zoneid = src->pr_zoneid;
419         dst->pr_uid = src->pr_uid;
420         dst->pr_euid = src->pr_euid;
421         dst->pr_gid = src->pr_gid;
422         dst->pr_egid = src->pr_egid;
423         dst->pr_addr = src->pr_addr;
424         dst->pr_size = src->pr_size;
425         dst->pr_rssize = src->pr_rssize;

427         dst->pr_ttydev = prexpldev(src->pr_ttydev);

429         dst->pr_pctcpu = src->pr_pctcpu;
430         dst->pr_pctmem = src->pr_pctmem;

432         timestruc_32_to_n(&src->pr_start, &dst->pr_start);
433         timestruc_32_to_n(&src->pr_time, &dst->pr_time);
434         timestruc_32_to_n(&src->pr_ctime, &dst->pr_ctime);

436         (void) memcpy(&dst->pr_fname[0], &src->pr_fname[0], PRFNSZ);
437         (void) memcpy(&dst->pr_psargs[0], &src->pr_psargs[0], PRARGSZ);

439         dst->pr_wstat = src->pr_wstat;
440         dst->pr_argc = src->pr_argc;
441         dst->pr_argv = src->pr_argv;
442         dst->pr_envp = src->pr_envp;
443         dst->pr_dmodel = src->pr_dmodel;

445         lwpsinfo_32_to_n(&src->pr_lwp, &dst->pr_lwp);
446 }

448 void
449 timestruc_n_to_32(const timestruc_t *src, timestruc32_t *dst)
450 {
451         dst->tv_sec = (time32_t)src->tv_sec;
452         dst->tv_nsec = (int32_t)src->tv_nsec;
453 }

455 void
456 stack_n_to_32(const stack_t *src, stack32_t *dst)
457 {
458         dst->ss_sp = (caddr32_t)(uintptr_t)src->ss_sp;
459         dst->ss_size = src->ss_size;
460         dst->ss_flags = src->ss_flags;
461 }

463 void
464 sigaction_n_to_32(const struct sigaction *src, struct sigaction32 *dst)
465 {
466         (void) memset(dst, 0, sizeof (struct sigaction32));
467         dst->sa_flags = src->sa_flags;
468         dst->sa_handler = (caddr32_t)(uintptr_t)src->sa_handler;
469         (void) memcpy(&dst->sa_mask, &src->sa_mask, sizeof (dst->sa_mask));
470 }

472 void
473 siginfo_n_to_32(const siginfo_t *src, siginfo32_t *dst)
474 {
475         (void) memset(dst, 0, sizeof (siginfo32_t));

477         /*
478          * The absolute minimum content is si_signo and si_code.
479          */
```



```
480         dst->si_signo = src->si_signo;
481         if ((dst->si_code = src->si_code) == SI_NOINFO)
482                 return;

484         /*
485          * A siginfo generated by user level is structured
486          * differently from one generated by the kernel.
487          */
488         if (SI_FROMUSER(src)) {
489                 dst->si_pid = src->si_pid;
490                 dst->si_ctid = src->si_ctid;
491                 dst->si_zoneid = src->si_zoneid;
492                 dst->si_uid = src->si_uid;
493                 if (SI_CANQUEUE(src->si_code)) {
494                         dst->si_value.sival_int =
495                             (int32_t)src->si_value.sival_int;
496                 }
497                 return;
498         }

500         dst->si_errno = src->si_errno;

502         switch (src->si_signo) {
503         default:
504                 dst->si_pid = src->si_pid;
505                 dst->si_ctid = src->si_ctid;
506                 dst->si_zoneid = src->si_zoneid;
507                 dst->si_uid = src->si_uid;
508                 dst->si_value.sival_int =
509                     (int32_t)src->si_value.sival_int;
510                 break;
511         case SIGCLD:
512                 dst->si_pid = src->si_pid;
513                 dst->si_ctid = src->si_ctid;
514                 dst->si_zoneid = src->si_zoneid;
515                 dst->si_status = src->si_status;
516                 dst->si_stime = src->si_stime;
517                 dst->si_utime = src->si_utime;
518                 break;
519         case SIGSEGV:
520         case SIGBUS:
521         case SIGILL:
522         case SIGTRAP:
523         case SIGFPE:
524         case SIGEMT:
525                 dst->si_addr = (caddr32_t)(uintptr_t)src->si_addr;
526                 dst->si_trapno = src->si_trapno;
527                 dst->si_pc = (caddr32_t)(uintptr_t)src->si_pc;
528                 break;
529         case SIGPOLL:
530         case SIGXFSZ:
531                 dst->si_fd = src->si_fd;
532                 dst->si_band = src->si_band;
533                 break;
534         case SIGPROF:
535                 dst->si_faddr = (caddr32_t)(uintptr_t)src->si_faddr;
536                 dst->si_tstamp.tv_sec = src->si_tstamp.tv_sec;
537                 dst->si_tstamp.tv_nsec = src->si_tstamp.tv_nsec;
538                 dst->si_syscall = src->si_syscall;
539                 dst->si_nsysarg = src->si_nsysarg;
540                 dst->si_fault = src->si_fault;
541                 break;
542         }
543 }

545 void
```

```
546 auxv_n_to_32(const auxv_t *src, auxv32_t *dst)
547 {
548         dst->a_type = src->a_type;
549         dst->a_un.a_ptr = (caddr32_t)(uintptr_t)src->a_un.a_ptr;
550 }

552 void
553 prgregset_n_to_32(const prgreg_t *src, prgreg32_t *dst)
554 {
555 #ifdef __amd64
556         (void) memset(dst, 0, NPRGREG32 * sizeof (prgreg32_t));
557         dst[GS] = src[REG_GS];
558         dst[FS] = src[REG_FS];
559         dst[DS] = src[REG_DS];
560         dst[ES] = src[REG_ES];
561         dst[EDI] = src[REG_RDI];
562         dst[ESI] = src[REG_RSI];
563         dst[EBP] = src[REG_RBP];
564         dst[EBX] = src[REG_RBX];
565         dst[EDX] = src[REG_RDX];
566         dst[ECX] = src[REG_RCX];
567         dst[EAX] = src[REG_RAX];
568         dst[TRAPNO] = src[REG_TRAPNO];
569         dst[ERR] = src[REG_ERR];
570         dst[EIP] = src[REG_RIP];
571         dst[CS] = src[REG_CS];
572         dst[EFL] = src[REG_RFL];
573         dst[UESP] = src[REG_RSP];
574         dst[SS] = src[REG_SS];
575 #else
576         int i;

578         for (i = 0; i < NPRGREG; i++)
579                 dst[i] = (prgreg32_t)src[i];
580 #endif
581 }

583 void
584 prfpregset_n_to_32(const prfpregset_t *src, prfpregset32_t *dst)
585 {
586 #if defined(__sparc)
587         int i;

589         (void) memset(dst, 0, sizeof (prfpregset32_t));

591         for (i = 0; i < 32; i++)
592                 dst->pr_fr.pr_regs[i] = src->pr_fr.pr_regs[i];

594         dst->pr_filler = src->pr_filler;
595         dst->pr_fsr = src->pr_fsr;
596         dst->pr_q_entrysize = src->pr_q_entrysize;
597         dst->pr_en = src->pr_en;

599 #elif defined(__amd64)

601         struct _fpstate32 *dst32 = (struct _fpstate32 *)dst;
602         struct fpchip_state *src64 = (struct fpchip_state *)src;
603         uint32_t top;
604         int i;

606         (void) memcpy(dst32->_st, src64->st, sizeof (dst32->_st));
607         (void) memcpy(dst32->xmm, src64->xmm, sizeof (dst32->xmm));
608         dst32->cw = src64->cw;
609         dst32->sw = src64->sw;
610         dst32->ipoff = (unsigned int)src64->rip;
611         dst32->cssel = 0;
```



```
612         dst32->dataoff = (unsigned int)src64->rdp;
613         dst32->datasel = 0;
614         dst32->status = src64->status;
615         dst32->mxcsr = src64->mxcsr;
616         dst32->xstatus = src64->xstatus;

618         /*
619          * AMD64 stores the tag in a compressed form. It is
620          * necessary to extract the original 2-bit tag value.
621          * See AMD64 Architecture Programmer's Manual Volume 2:
622          * System Programming, Chapter 11.
623          */

625         top = (src64->sw & FPS_TOP) >> 11;
626         dst32->tag = 0;
627         for (i = 0; i < 8; i++) {
628                 /*
629                  * Recall that we need to use the current TOP-of-stack value to
630                  * associate the _st[] index back to a physical register number,
631                  * since tag word indices are physical register numbers.  Then
632                  * to get the tag value, we shift over two bits for each tag
633                  * index, and then grab the bottom two bits.
634                  */
635                 uint_t tag_index = (i + top) & 7;
636                 uint_t tag_fctw = (src64->fctw >> tag_index) & 1;
637                 uint_t tag_value;
638                 uint_t exp;

640                 /*
641                  * Union for overlaying _fpreg structure on to quad-precision
642                  * floating-point value (long double).
643                  */
644                 union {
645                         struct _fpreg reg;
646                         long double ld;
647                 } fpru;

649                 fpru.ld = src64->st[i].__fpr_pad._q;
650                 exp = fpru.reg.exponent & 0x7fff;

652                 if (tag_fctw == 0) {
653                         tag_value = 3; /* empty */
654                 } else if (exp == 0) {
655                         if (fpru.reg.significand[0] == 0 &&
656                             fpru.reg.significand[1] == 0 &&
657                             fpru.reg.significand[2] == 0 &&
658                             fpru.reg.significand[3] == 0)
659                                 tag_value = 1; /* zero */
660                         else
661                                 tag_value = 2; /* special: denormal */
662                 } else if (exp == 0x7fff) {
663                         tag_value = 2; /* special: infinity or NaN */
664                 } else if (fpru.reg.significand[3] & 0x8000) {
665                         tag_value = 0; /* valid */
666                 } else {
667                         tag_value = 2; /* special: unnormal */
668                 }
669                 dst32->tag |= tag_value << (tag_index * 2);
670         }
671 #else
672 #error "unrecognized ISA"
673 #endif
674 }

676 void
677 lwpstatus_n_to_32(const lwpstatus_t *src, lwpstatus32_t *dst)
```

```
678 {
679         int i;

681         dst->pr_flags = src->pr_flags;
682         dst->pr_lwpid = src->pr_lwpid;
683         dst->pr_why = src->pr_why;
684         dst->pr_what = src->pr_what;
685         dst->pr_cursig = src->pr_cursig;

687         siginfo_n_to_32(&src->pr_info, &dst->pr_info);

689         dst->pr_lwppend = src->pr_lwppend;
690         dst->pr_lwphold = src->pr_lwphold;

692         sigaction_n_to_32(&src->pr_action, &dst->pr_action);
693         stack_n_to_32(&src->pr_altstack, &dst->pr_altstack);

695         dst->pr_oldcontext = (caddr32_t)src->pr_oldcontext;
696         dst->pr_syscall = src->pr_syscall;
697         dst->pr_nsysarg = src->pr_nsysarg;
698         dst->pr_errno = src->pr_errno;

700         for (i = 0; i < PRSYSARGS; i++)
701                 dst->pr_sysarg[i] = (int32_t)src->pr_sysarg[i];

703         dst->pr_rval1 = (int32_t)src->pr_rval1;
704         dst->pr_rval2 = (int32_t)src->pr_rval2;

706         (void) memcpy(&dst->pr_clname[0], &src->pr_clname[0], PRCLSZ);
707         timestruc_n_to_32(&src->pr_tstamp, &dst->pr_tstamp);

709         dst->pr_ustack = (caddr32_t)src->pr_ustack;
710         dst->pr_instr = src->pr_instr;

712         prgregset_n_to_32(src->pr_reg, dst->pr_reg);
713         prfpregset_n_to_32(&src->pr_fpreg, &dst->pr_fpreg);
714 }

716 void
717 pstatus_n_to_32(const pstatus_t *src, pstatus32_t *dst)
718 {
719         dst->pr_flags = src->pr_flags;
720         dst->pr_nlwp = src->pr_nlwp;
721         dst->pr_nzomb = src->pr_nzomb;
722         dst->pr_pid = (pid32_t)src->pr_pid;
723         dst->pr_ppid = (pid32_t)src->pr_ppid;
724         dst->pr_pgid = (pid32_t)src->pr_pgid;
725         dst->pr_sid = (pid32_t)src->pr_sid;
726         dst->pr_taskid = (id32_t)src->pr_taskid;
727         dst->pr_projid = (id32_t)src->pr_projid;
728         dst->pr_zoneid = (id32_t)src->pr_zoneid;
729         dst->pr_aslwpid = (id32_t)src->pr_aslwpid;
730         dst->pr_agentid = (id32_t)src->pr_agentid;
731         dst->pr_sigpend = src->pr_sigpend;
732         dst->pr_brkbase = (caddr32_t)src->pr_brkbase;
733         dst->pr_brksize = (size32_t)src->pr_brksize;
734         dst->pr_stkbase = (caddr32_t)src->pr_stkbase;
735         dst->pr_stksize = (size32_t)src->pr_stksize;

737         timestruc_n_to_32(&src->pr_utime, &dst->pr_utime);
738         timestruc_n_to_32(&src->pr_stime, &dst->pr_stime);
739         timestruc_n_to_32(&src->pr_cutime, &dst->pr_cutime);
740         timestruc_n_to_32(&src->pr_cstime, &dst->pr_cstime);

742         dst->pr_sigtrace = src->pr_sigtrace;
743         dst->pr_flttrace = src->pr_flttrace;
```



```
744         dst->pr_sysentry = src->pr_sysentry;
745         dst->pr_sysexit = src->pr_sysexit;
746         dst->pr_dmodel = src->pr_dmodel;

748         (void) memcpy(&dst->pr_secflags, &src->pr_secflags, sizeof (psecflags_t)

750 #endif /* ! codereview */
751         lwpstatus_n_to_32(&src->pr_lwp, &dst->pr_lwp);
752 }

754 void
755 lwpsinfo_n_to_32(const lwpsinfo_t *src, lwpsinfo32_t *dst)
756 {
757         dst->pr_flag = src->pr_flag;
758         dst->pr_lwpid = (id32_t)src->pr_lwpid;
759         dst->pr_addr = (caddr32_t)src->pr_addr;
760         dst->pr_wchan = (caddr32_t)src->pr_wchan;
761         dst->pr_stype = src->pr_stype;
762         dst->pr_state = src->pr_state;
763         dst->pr_sname = src->pr_sname;
764         dst->pr_nice = src->pr_nice;
765         dst->pr_syscall = src->pr_syscall;
766         dst->pr_oldpri = src->pr_oldpri;
767         dst->pr_cpu = src->pr_cpu;
768         dst->pr_pri = src->pr_pri;
769         dst->pr_pctcpu = src->pr_pctcpu;

771         timestruc_n_to_32(&src->pr_start, &dst->pr_start);
772         timestruc_n_to_32(&src->pr_time, &dst->pr_time);

774         (void) memcpy(&dst->pr_clname[0], &src->pr_clname[0], PRCLSZ);
775         (void) memcpy(&dst->pr_name[0], &src->pr_name[0], PRFNSZ);

777         dst->pr_onpro = src->pr_onpro;
778         dst->pr_bindpro = src->pr_bindpro;
779         dst->pr_bindpset = src->pr_bindpset;
780         dst->pr_lgrp = src->pr_lgrp;
781 }

783 void
784 psinfo_n_to_32(const psinfo_t *src, psinfo32_t *dst)
785 {
786         dst->pr_flag = src->pr_flag;
787         dst->pr_nlwp = src->pr_nlwp;
788         dst->pr_nzomb = src->pr_nzomb;
789         dst->pr_pid = (pid32_t)src->pr_pid;
790         dst->pr_pgid = (pid32_t)src->pr_pgid;
791         dst->pr_sid = (pid32_t)src->pr_sid;
792         dst->pr_taskid = (id32_t)src->pr_taskid;
793         dst->pr_projid = (id32_t)src->pr_projid;
794         dst->pr_zoneid = (id32_t)src->pr_zoneid;
795         dst->pr_uid = (uid32_t)src->pr_uid;
796         dst->pr_euid = (uid32_t)src->pr_euid;
797         dst->pr_gid = (gid32_t)src->pr_gid;
798         dst->pr_egid = (gid32_t)src->pr_egid;
799         dst->pr_addr = (caddr32_t)src->pr_addr;
800         dst->pr_size = (size32_t)src->pr_size;
801         dst->pr_rssize = (size32_t)src->pr_rssize;

803         dst->pr_ttydev = prcmpldev(src->pr_ttydev);

805         dst->pr_pctcpu = src->pr_pctcpu;
806         dst->pr_pctmem = src->pr_pctmem;

808         timestruc_n_to_32(&src->pr_start, &dst->pr_start);
809         timestruc_n_to_32(&src->pr_time, &dst->pr_time);
```

```
810         timestruc_n_to_32(&src->pr_ctime, &dst->pr_ctime);

812         (void) memcpy(&dst->pr_fname[0], &src->pr_fname[0], PRFNSZ);
813         (void) memcpy(&dst->pr_psargs[0], &src->pr_psargs[0], PRARGSZ);

815         dst->pr_wstat = src->pr_wstat;
816         dst->pr_argc = src->pr_argc;
817         dst->pr_argv = (caddr32_t)src->pr_argv;
818         dst->pr_envp = (caddr32_t)src->pr_envp;
819         dst->pr_dmodel = src->pr_dmodel;

821         lwpsinfo_n_to_32(&src->pr_lwp, &dst->pr_lwp);
822 }


825 #endif  /* _LP64 */
```

```
**********************************************************
   14426 Wed May 27 19:49:15 2015
new/usr/src/lib/libsecdb/auth_attr.txt
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
```

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#

# Copyright (c) 1999, 2010, Oracle and/or its affiliates. All rights reserved.


#
# /etc/security/auth_attr
#
# authorizations. see auth_attr(4)
#
solaris.:::All Solaris Authorizations::help=AllSolAuthsHeader.html
solaris.grant:::Grant All Solaris Authorizations::help=PriAdmin.html
#
solaris.admin.idmap.rules:::Manage Identity Mapping Rules::help=IdmapRules.html
#
solaris.admin.wusb.:::Administer Wireless USB::help=WUSBHeader.html
solaris.admin.wusb.read:::Read Wireless USB Host and Device Information::help=WU
solaris.admin.wusb.modify:::Add or delete information of Wireless USB Device::he
solaris.admin.wusb.host:::Manage Wireless USB Host::help=WUSBhost.html
#
solaris.audit.:::Audit System-wide Management::help=AuditHeader.html
#
solaris.device.:::Device Allocation::help=DevAllocHeader.html
solaris.device.allocate:::Allocate Device::help=DevAllocate.html
solaris.device.config:::Configure Device Attributes::help=DevConfig.html
solaris.device.grant:::Delegate Device Administration::help=DevGrant.html
solaris.device.revoke:::Revoke or Reclaim Device::help=DevRevoke.html
solaris.device.cdrw:::CD-R/RW Recording Authorizations::help=DevCDRW.html
solaris.device.mount.:::Device Mount::help=DevMount.html
solaris.device.mount.alloptions.fixed:::Device Mount Fixed With All Options::hel
solaris.device.mount.alloptions.removable:::Device Mount Removable With All Opti
solaris.device.mount.fixed:::Device Mount Fixed::help=DevMount.html
solaris.device.mount.removable:::Device Mount Removable::help=DevMount.html
#
solaris.dhcpmgr.:::DHCP Service Management::help=DhcpmgrHeader.html
solaris.dhcpmgr.write:::Modify DHCP Service Configuration::help=DhcpmgrWrite.htm
#
solaris.file.:::File Operations::help=FileHeader.html
solaris.file.chown:::Change File Owner::help=FileChown.html
```



```
solaris.file.owner:::Act as File Owner::help=FileOwner.html
#
solaris.hotplug.:::Hotplug::help=HotplugHeader.html
solaris.hotplug.modify:::Modify Hotplug Connections::help=HotplugModify.html
#
solaris.jobs.:::Job Scheduler::help=JobHeader.html
solaris.jobs.admin:::Manage All Jobs::help=AuthJobsAdmin.html
solaris.jobs.grant:::Delegate Cron & At Administration::help=JobsGrant.html
solaris.jobs.user:::Manage Owned Jobs::help=AuthJobsUser.html
#
solaris.label.:::Label Management::help=LabelHeader.html
solaris.label.file.downgrade:::Downgrade File Label::help=LabelFileDowngrade.htm
solaris.label.file.upgrade:::Upgrade File Label::help=LabelFileUpgrade.html
solaris.label.print:::View Printer Queue at All Labels::help=LabelPrint.html
solaris.label.range:::Set Label Outside User Accred Range::help=LabelRange.html
solaris.label.win.downgrade:::Downgrade DragNDrop or CutPaste Info::help=LabelWi
solaris.label.win.noview:::DragNDrop or CutPaste without viewing contents::help=
solaris.label.win.upgrade:::Upgrade DragNDrop or CutPaste Info::help=LabelWinUpg
#
solaris.login.:::Login Control::help=LoginHeader.html
solaris.login.enable:::Enable Logins::help=LoginEnable.html
solaris.login.remote:::Remote Login::help=LoginRemote.html
#
solaris.mail.:::Mail::help=MailHeader.html
solaris.mail.mailq:::Mail Queue::help=MailQueue.html
#
solaris.network.:::Network::help=NetworkHeader.html
solaris.network.autoconf.read:::View Network Auto-Magic Config::help=NetworkAuto
solaris.network.autoconf.select:::Enable/Disable Network Auto-Magic Config::help
solaris.network.autoconf.wlan:::Create Network Auto-Magic Config for Known WLANs
solaris.network.autoconf.write:::Create Network Auto-Magic Config::help=NetworkA
solaris.network.ilb.config:::Network ILB Configuration::help=NetworkILBconf.html
solaris.network.ilb.enable:::Network ILB Enable Configuration::help=NetworkILBen
solaris.network.interface.config:::Network Interface Configuration::help=Network
solaris.network.link.security:::Link Security::help=LinkSecurity.html
solaris.network.wifi.config:::Wifi Config::help=WifiConfig.html
solaris.network.wifi.wep:::Wifi Wep::help=WifiWep.html
solaris.network.vrrp:::Administer VRRP::help=NetworkVRRP.html
#
solaris.print.:::Printer Management::help=PrintHeader.html
solaris.print.admin:::Administer Printer::help=PrintAdmin.html
solaris.print.cancel:::Cancel Print Job::help=PrintCancel.html
solaris.print.list:::List Jobs in Printer Queue::help=PrintList.html
solaris.print.nobanner:::Print without Banner::help=PrintNoBanner.html
solaris.print.ps:::Print Postscript::help=PrintPs.html
solaris.print.unlabeled:::Print without Label::help=PrintUnlabeled.html
#
solaris.profmgr.:::Rights::help=ProfmgrHeader.html
solaris.profmgr.assign:::Assign All Rights::help=AuthProfmgrAssign.html
solaris.profmgr.delegate:::Assign Owned Rights::help=AuthProfmgrDelegate.html
solaris.profmgr.write:::Manage Rights::help=AuthProfmgrWrite.html
solaris.profmgr.read:::View Rights::help=AuthProfmgrRead.html
solaris.profmgr.execattr.write:::Manage Commands::help=AuthProfmgrExecattrWrite.
#
solaris.role.:::Roles::help=RoleHeader.html
solaris.role.assign:::Assign All Roles::help=AuthRoleAssign.html
solaris.role.delegate:::Assign Owned Roles::help=AuthRoleDelegate.html
solaris.role.write:::Manage Roles::help=AuthRoleWrite.html
#
solaris.smf.:::SMF Management::help=SmfHeader.html
solaris.smf.modify.:::Modify All SMF Service Properties::help=SmfModifyHeader.ht
solaris.smf.modify.method:::Modify Service Methods::help=SmfModifyMethod.html
solaris.smf.modify.dependency:::Modify Service Dependencies::help=SmfModifyDepen
solaris.smf.modify.application:::Modify Application Type Properties::help=SmfMod
solaris.smf.modify.framework:::Modify Framework Type Properties::help=SmfModifyF
solaris.smf.manage.:::Manage All SMF Service States::help=SmfManageHeader.html
```

```
125 solaris.smf.manage.allocate:::Manage Device Allocation Service::help=SmfAllocate
126 solaris.smf.manage.audit:::Manage Audit Service States::help=SmfManageAudit.html
127 solaris.smf.manage.autofs:::Manage Automount Service States::help=SmfAutofsState
128 solaris.smf.manage.bind:::Manage DNS Service States::help=BindStates.html
129 solaris.smf.manage.coreadm:::Manage Coreadm Service States::help=SmfCoreadmState
130 solaris.smf.manage.cron:::Manage Cron Service States::help=SmfCronStates.html
131 solaris.smf.manage.discovery.printers.snmp:::Manage Network Attached Device Disc
132 solaris.smf.manage.extended-accounting.flow:::Manage Flow Extended Accounting Se
133 solaris.smf.manage.extended-accounting.process:::Manage Process Extended Account
134 solaris.smf.manage.extended-accounting.flow:::Manage Task Extended Accounting Se
135 solaris.smf.manage.hal:::Manage HAL Service States::help=SmfHALStates.html
136 solaris.smf.manage.hotplug:::Manage Hotplug Service::help=SmfManageHotplug.html
137 solaris.smf.manage.idmap:::Manage Identity Mapping Service States::help=SmfIdmap
138 solaris.smf.manage.ilb:::Manage Integrated Load Balancer Service States::help=Sm
139 solaris.smf.manage.inetd:::Manage inetd and inetd managed services States::help=
140 solaris.smf.manage.ipsec:::Manage IPsec Service States::help=SmfIPsecStates.html
141 solaris.smf.manage.labels:::Manage label server::help=LabelServer.html
142 solaris.smf.manage.location:::Manage Network Location Service States::help=SmfLo
143 solaris.smf.manage.mdns:::Manage Multicast DNS Service States::help=SmfMDNSState
144 solaris.smf.manage.name-service-cache:::Manage Name Service Cache Daemon Service
145 solaris.smf.manage.nwam:::Manage Network Auto-Magic Service States::help=SmfNWAM
146 solaris.smf.manage.power:::Manage Power Management Service States::help=SmfPower
147 solaris.smf.manage.smb:::Manage SMB Service States::help=SmfSMBStates.html
148 solaris.smf.manage.smbfs:::Manage SMB Client States::help=SmfSMBFSStates.html
149 solaris.smf.manage.reparse:::Manage Reparse Service States::help=SmfReparseState
150 solaris.smf.manage.rmvolmgr:::Manage Rmvolmgr Service States::help=SmfRmvolmgrSt
151 solaris.smf.manage.routing:::Manage Routing Service States::help=SmfRoutingState
152 solaris.smf.manage.rpc.bind:::Manage RPC Program number mapper::help=SmfRPCBind.
153 solaris.smf.manage.sendmail:::Manage Sendmail Service States::help=SmfSendmailSt
154 solaris.smf.manage.smtp-notify:::Manage Email Event Notification Agent::
155 solaris.smf.manage.snmp-notify:::Manage SNMP Event Notification Agent::
156 solaris.smf.manage.ssh:::Manage Secure Shell Service States::help=SmfSshStates.h
157 solaris.smf.manage.stmf:::Manage STMF Service States::help=SmfSTMFStates.html
158 solaris.smf.manage.system-log:::Manage Syslog Service States::help=SmfSyslogStat
159 solaris.smf.manage.tnctl:::Manage Refresh of Trusted Network Parameters::help=TN
160 solaris.smf.manage.tnd:::Manage Trusted Network Daemon::help=TNDaemon.html
161 solaris.smf.manage.vrrp:::Manage VRRP Service States::help=SmfVRRPStates.html
162 solaris.smf.manage.vscan:::Manage VSCAN Service States::help=SmfVscanStates.html
163 solaris.smf.manage.vt:::Manage Virtual Console Service States::help=SmfVtStates.
164 solaris.smf.manage.wpa:::Manage WPA Service States::help=SmfWpaStates.html
165 solaris.smf.manage.ndmp:::Manage NDMP Service States::help=SmfNDMPStates.html
166 solaris.smf.value.:::Change Values of SMF Service Properties::help=SmfValueHeade
167 solaris.smf.value.audit:::Configure the Audit Service::help=SmfValueAudit.html
168 solaris.smf.value.coreadm:::Change Values of SMF Coreadm Properties::help=SmfVal
169 solaris.smf.value.discovery.printers.snmp:::Manage Network Attached Device Disco
170 solaris.smf.value.extended-accounting.flow:::Change Values of Flow Extended Acco
171 solaris.smf.value.extended-accounting.process:::Change Values of Process Extende
172 solaris.smf.value.extended-accounting.task:::Change Values of Task Extended Acco
173 solaris.smf.value.firewall.config:::Change Service Firewall Config::help=SmfValu
174 solaris.smf.value.idmap:::Change Values of SMF Identity Mapping Service Properti
175 solaris.smf.value.inetd:::Change values of SMF Inetd configuration paramaters::h
176 solaris.smf.value.ipsec:::Change Values of SMF IPsec Properties::help=SmfValueIP
177 solaris.smf.value.mdns:::Change Values of MDNS Service Properties::help=SmfValue
178 solaris.smf.value.nwam:::Change Values of SMF Network Auto-Magic Properties::hel
179 solaris.smf.value.process-security:::Change Values of Process Security propertie
180 #endif /* ! codereview */
181 solaris.smf.value.smb:::Change Values of SMB Service Properties::help=SmfValueSM
182 solaris.smf.read.smb:::Read permission for protected SMF SMB Service Properties:
183 solaris.smf.value.smtp-notify:::Change values of Email Event Notification Agent
184 solaris.smf.value.snmp-notify:::Change values of SNMP Event Notification Agent p
185 solaris.smf.read.stmf:::Read STMF Provider Private Data::help=SmfSTMFRead.html
186 solaris.smf.value.routing:::Change Values of SMF Routing Properties::help=SmfVal
187 solaris.smf.value.tnd:::Change Trusted Network Daemon Service Property Values::h
188 solaris.smf.value.vscan:::Change Values of VSCAN Properties::help=SmfValueVscan.
189 solaris.smf.value.vt:::Change Values of Virtual Console Service Properties::help
190 solaris.smf.value.ndmp:::Change Values of SMF NDMP Service Properties::help=SmfV
```



```
191 solaris.smf.read.ndmp:::Read permission for protected SMF NDMP Service Propertie
192 #
193 solaris.system.:::Machine Administration::help=SysHeader.html
194 solaris.system.date:::Set Date & Time::help=SysDate.html
195 solaris.system.maintenance:::Enter Maintenance (single-user) Mode::help=SysMaint
196 solaris.system.shutdown:::Shutdown the System::help=SysShutdown.html
197 solaris.system.power.:::System Power Management::help=SysPowerMgmtHeader.html
198 solaris.system.power.suspend.:::Suspend the System::help=SysPowerMgmtSuspend.htm
199 solaris.system.power.suspend.disk:::Suspend to Disk::help=SysPowerMgmtSuspendtoD
200 solaris.system.power.suspend.ram:::Suspend to RAM::help=SysPowerMgmtSuspendToRAM
201 solaris.system.power.brightness:::Control LCD Brightness::help=SysPowerMgmtBrigh
202 solaris.system.power.cpu:::Manage CPU related power::help=SysCpuPowerMgmt.html
203 solaris.system.sysevent.read:::Retrieve Sysevents::help=SysSyseventRead.html
204 solaris.system.sysevent.write:::Publish Sysevents::help=SysSyseventWrite.html
205 #
206 solaris.smf.modify.stmf:::Modify STMF Properties::help=SmfSTMFValue.html
207 #
208 solaris.smf.manage.isns:::Manage iSNS Service States::help=isnsStates.html
209 solaris.smf.value.isns:::Modify iSNS Service Property Values::help=isnsValue.htm
210 solaris.isnsmgr.write:::Modify iSNS configuration::help=AuthISNSmgrWrite.html
211 solaris.smf.manage.wusb:::Manage Wireless USB Service::help=SmfWusbStates.html
212 solaris.zone.:::Zone Management::help=ZoneHeader.html
213 solaris.zone.clonefrom:::Clone another Zone::help=ZoneCloneFrom.html
214 solaris.zone.login:::Zone Login::help=ZoneLogin.html
215 solaris.zone.manage:::Zone Deployment::help=ZoneManage.html
```

```
**********************************************************
    4783 Wed May 27 19:49:16 2015
new/usr/src/lib/libsecdb/help/auths/Makefile
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#

# Copyright (c) 1999, 2010, Oracle and/or its affiliates. All rights reserved.

#
# lib/libsecdb/help/auths/Makefile
#

include ../../../../Makefile.master

HTMLENTS = \
        AuditHeader.html \
        DevAllocHeader.html \
        DevAllocate.html \
        DevConfig.html \
        DevCDRW.html \
        DevGrant.html \
        DevRevoke.html \
        HotplugHeader.html \
        HotplugModify.html \
        JobHeader.html \
        AuthJobsAdmin.html \
        JobsGrant.html \
        AuthJobsUser.html \
        LoginEnable.html \
        LoginHeader.html \
        LoginRemote.html \
        MailHeader.html \
        MailQueue.html \
        PriAdmin.html \
        AuthProfmgrAssign.html \
        AuthProfmgrDelegate.html \
        AuthProfmgrExecattrWrite.html \
        AuthProfmgrRead.html \
        ProfmgrHeader.html \
        AuthProfmgrWrite.html \
        AuthRoleAssign.html \
        AuthRoleDelegate.html \
        RoleHeader.html \
```



```
        AuthRoleWrite.html \
        SysDate.html \
        SysHeader.html \
        SysShutdown.html \
        AllSolAuthsHeader.html \
        SysMaintenance.html \
        DhcpmgrHeader.html \
        DhcpmgrWrite.html \
        BindStates.html \
        SmfAllocate.html \
        SmfAutofsStates.html \
        SmfCoreadmStates.html \
        SmfCronStates.html \
        SmfExAcctFlowStates.html \
        SmfExAcctProcessStates.html \
        SmfExAcctTaskStates.html \
        SmfExAcctNetStates.html \
        SmfHeader.html \
        SmfILBStates.html \
        SmfInetdStates.html \
        SmfIPsecStates.html \
        SmfLocationStates.html \
        SmfManageAudit.html \
        SmfManageHeader.html \
        SmfManageHotplug.html \
        SmfMDNSStates.html \
        SmfModifyAppl.html \
        SmfModifyDepend.html \
        SmfModifyFramework.html \
        SmfModifyHeader.html \
        SmfModifyMethod.html \
        SmfNscdStates.html \
        SmfNADDStates.html \
        SmfNDMPStates.html \
        SmfNWAMStates.html \
        SmfPowerStates.html \
        SmfReparseStates.html \
        SmfRoutingStates.html \
        SmfSendmailStates.html \
        SmfSshStates.html \
        SmfSyslogStates.html \
        SmfValueAudit.html \
        SmfValueCoreadm.html \
        SmfValueExAcctFlow.html \
        SmfValueExAcctProcess.html \
        SmfValueExAcctTask.html \
        SmfValueExAcctNet.html \
        SmfValueFirewall.html \
        SmfVtStates.html \
        SmfValueHeader.html \
        SmfValueInetd.html \
        SmfValueIPsec.html \
        SmfValueMDNS.html \
        SmfValueNADD.html \
        SmfValueNDMP.html \
        AuthReadNDMP.html \
        SmfValueNWAM.html \
        SmfValueProcSec.html \
#endif /* ! codereview */
        SmfValueRouting.html \
        SmfValueSMB.html \
        AuthReadSMB.html \
        SmfSMBFSStates.html \
        SmfSMBStates.html \
        SmfValueVscan.html \
        SmfVscanStates.html \
```

```
125         SmfValueVt.html \
126         SmfVRRPStates.html \
127         SmfWpaStates.html \
128         NetworkAutoconfRead.html \
129         NetworkAutoconfSelect.html \
130         NetworkAutoconfWlan.html \
131         NetworkAutoconfWrite.html \
132         NetworkILBconf.html \
133         NetworkILBenable.html \
134         NetworkHeader.html \
135         NetworkVRRP.html \
136         NetworkInterfaceConfig.html \
137         WifiConfig.html \
138         WifiWep.html \
139         LinkSecurity.html \
140         IdmapRules.html \
141         SmfIdmapStates.html \
142         SmfValueIdmap.html \
143         FileChown.html \
144         FileHeader.html \
145         FileOwner.html \
146         LabelFileDowngrade.html \
147         LabelFileUpgrade.html \
148         LabelHeader.html \
149         LabelPrint.html \
150         LabelRange.html \
151         LabelServer.html \
152         LabelWinDowngrade.html \
153         LabelWinNoView.html \
154         LabelWinUpgrade.html \
155         PrintAdmin.html \
156         PrintCancel.html \
157         PrintHeader.html \
158         PrintList.html \
159         PrintNoBanner.html \
160         PrintPs.html \
161         PrintUnlabeled.html \
162         TNDaemon.html \
163         TNctl.html \
164         ValueTND.html \
165         SysPowerMgmtHeader.html \
166         SysPowerMgmtSuspend.html \
167         SysPowerMgmtSuspendtoDisk.html \
168         SysPowerMgmtSuspendtoRAM.html \
169         SysPowerMgmtBrightness.html \
170         SysCpuPowerMgmt.html \
171         SysSyseventRead.html \
172         SysSyseventWrite.html \
173         SmfManageZFSSnap.html \
174         ZoneCloneFrom.html \
175         ZoneHeader.html \
176         ZoneLogin.html \
177         ZoneManage.html

179 HELPDIR=$(ROOT)/usr/lib/help
180 AUTHDIR=$(HELPDIR)/auths
181 LOCALEDIR=$(AUTHDIR)/locale
182 CDIR=$(LOCALEDIR)/C
183 DIRS=$(HELPDIR) $(AUTHDIR) $(LOCALEDIR) $(CDIR)
184 HELPFILES=$(HTMLENTS:%=$(CDIR)/%)

186 MSGDIR=         $(LOCALEDIR)
187 MSGDIRS =       $(HELPDIR) $(AUTHDIR) $(LOCALEDIR)

189 MSGFILES=       $(HTMLENTS)
190 MSGS=           $(MSGFILES:%=$(MSGDIR)/%)
```



```
192 FILEMODE= 0444

194 .KEEP_STATE:

196 all:    $(HTMLENTS)

198 install:        all $(DIRS) $(HELPFILES)

200 _msg: $(MSGDIRS) $(MSGS)

202 $(CDIR)/%: %
203         $(INS.file)

205 $(DIRS):
206         $(INS.dir)

208 $(MSGDIR)/%: %
209         $(INS.file)

211 clean clobber lint:
```

```
**********************************************************
     821 Wed May 27 19:49:16 2015
new/usr/src/lib/libsecdb/help/auths/SmfValueProcSec.html
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
```

```
<HTML>
<!--
 This file and its contents are supplied under the terms of the
 Common Development and Distribution License ("CDDL"), version 1.0.
 You may only use this file in accordance with the terms of version
 1.0 of the CDDL.

 A full copy of the text of the CDDL should have accompanied this
 source.  A copy of the CDDL is also available via the Internet at
 http://www.illumos.org/license/CDDL.
-->
<!--
   <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso-8859-1">
-->
<BODY>
When Value Process Security Properties is in the Authorizations Include
column, it grants the the authorization to change the default security-flags
for processes on this system
<P>
If Value Process Security Properties is grayed, then you are not entitled to
Add or Remove this authorization.
<BR> 
</BODY>
</HTML>
#endif /* ! codereview */
```

```
**********************************************************
   18136 Wed May 27 19:49:16 2015
new/usr/src/lib/pam_modules/unix_cred/unix_cred.c
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
______unchanged_portion_omitted_

 153 /*
 154  *      unix_cred - pam_sm_setcred
 155  *
 156  *      Entry flags =   PAM_ESTABLISH_CRED, set up Solaris Unix cred.
 157  *                      PAM_DELETE_CRED, NOP, return PAM_SUCCESS.
 158  *                      PAM_REINITIALIZE_CRED, set up Solaris Unix cred,
 159  *                              or merge the current context with the new
 160  *                              user.
 161  *                      PAM_REFRESH_CRED, set up Solaris Unix cred.
 162  *                      PAM_SILENT, print no messages to user.
 163  *
 164  *      Returns PAM_SUCCESS, if all successful.
 165  *              PAM_CRED_ERR, if unable to set credentials.
 166  *              PAM_USER_UNKNOWN, if PAM_USER not set, or unable to find
 167  *                      user in databases.
 168  *              PAM_SYSTEM_ERR, if no valid flag, or unable to get/set
 169  *                      user's audit state.
 170  */

 172 int
 173 pam_sm_setcred(pam_handle_t *pamh, int flags, int argc, const char **argv)
 174 {
 175         int     i;
 176         int     debug = 0;
 177         uint_t  nowarn = flags & PAM_SILENT;
 178         int     ret = PAM_SUCCESS;
 179         char    *user;
 180         char    *auser;
 181         char    *rhost;
 182         char    *tty;
 183         au_id_t auid;
 184         adt_session_data_t *ah;
 185         adt_termid_t    *termid = NULL;
 186         priv_set_t      *lim, *def, *tset;
 187         char            messages[PAM_MAX_NUM_MSG][PAM_MAX_MSG_SIZE];
 188         char            buf[PROJECT_BUFSZ];
 189         struct project  proj, *pproj;
 190         int             error;
 191         char            *projname;
 192         char            *kvs;
 193         struct passwd   pwd;
 194         char            pwbuf[NSS_BUFLEN_PASSWD];
 195         deflim_t        deflim;

 197         for (i = 0; i < argc; i++) {
 198                 if (strcmp(argv[i], "debug") == 0)
 199                         debug = 1;
 200                 else if (strcmp(argv[i], "nowarn") == 0)
 201                         nowarn |= 1;
 202         }

 204         if (debug)
 205                 syslog(LOG_AUTH | LOG_DEBUG,
 206                     "pam_unix_cred: pam_sm_setcred(flags = %x, argc= %d)",
 207                     flags, argc);
```



```
 209         (void) pam_get_item(pamh, PAM_USER, (void **)&user);

 211         if (user == NULL || *user == '\0') {
 212                 syslog(LOG_AUTH | LOG_ERR,
 213                     "pam_unix_cred: USER NULL or empty!\n");
 214                 return (PAM_USER_UNKNOWN);
 215         }
 216         (void) pam_get_item(pamh, PAM_AUSER, (void **)&auser);
 217         (void) pam_get_item(pamh, PAM_RHOST, (void **)&rhost);
 218         (void) pam_get_item(pamh, PAM_TTY, (void **)&tty);
 219         if (debug)
 220                 syslog(LOG_AUTH | LOG_DEBUG,
 221                     "pam_unix_cred: user = %s, auser = %s, rhost = %s, "
 222                     "tty = %s", user,
 223                     (auser == NULL) ? "NULL" : (*auser == '\0') ? "ZERO" :
 224                     auser,
 225                     (rhost == NULL) ? "NULL" : (*rhost == '\0') ? "ZERO" :
 226                     rhost,
 227                     (tty == NULL) ? "NULL" : (*tty == '\0') ? "ZERO" :
 228                     tty);

 230         /* validate flags */
 231         switch (flags & (PAM_ESTABLISH_CRED | PAM_DELETE_CRED |
 232             PAM_REINITIALIZE_CRED | PAM_REFRESH_CRED)) {
 233         case 0:
 234                 /* set default flag */
 235                 flags |= PAM_ESTABLISH_CRED;
 236                 break;
 237         case PAM_ESTABLISH_CRED:
 238         case PAM_REINITIALIZE_CRED:
 239         case PAM_REFRESH_CRED:
 240                 break;
 241         case PAM_DELETE_CRED:
 242                 return (PAM_SUCCESS);
 243         default:
 244                 syslog(LOG_AUTH | LOG_ERR,
 245                     "pam_unix_cred: invalid flags %x", flags);
 246                 return (PAM_SYSTEM_ERR);
 247         }

 249         /*
 250          * if auditing on and process audit state not set,
 251          * setup audit context for process.
 252          */
 253         if (adt_start_session(&ah, NULL, ADT_USE_PROC_DATA) != 0) {
 254                 syslog(LOG_AUTH | LOG_ERR,
 255                     "pam_unix_cred: cannot create start audit session %m");
 256                 return (PAM_SYSTEM_ERR);
 257         }
 258         adt_get_auid(ah, &auid);
 259         if (debug) {
 260                 int     auditstate;

 262                 if (auditon(A_GETCOND, (caddr_t)&auditstate,
 263                     sizeof (auditstate)) != 0) {
 264                         auditstate = AUC_DISABLED;
 265                 }
 266                 syslog(LOG_AUTH | LOG_DEBUG,
 267                     "pam_unix_cred: state = %d, auid = %d", auditstate,
 268                     auid);
 269         }
 270         if (getpwnam_r(user, &pwd, pwbuf, sizeof (pwbuf)) == NULL) {
 271                 syslog(LOG_AUTH | LOG_ERR,
 272                     "pam_unix_cred: cannot get passwd entry for user = %s",
 273                     user);
 274                 ret = PAM_USER_UNKNOWN;
```

```
275                 goto adt_done;
276         }

278         if ((auid == AU_NOAUDITID) &&
279             (flags & PAM_ESTABLISH_CRED)) {
280                 struct passwd   apwd;
281                 char    apwbuf[NSS_BUFLEN_PASSWD];

283                 errno = 0;
284                 if ((rhost == NULL || *rhost == '\0')) {
285                         if (adt_load_ttyname(tty, &termid) != 0) {
286                                 if (errno == ENETDOWN) {
287                                         /*
288                                          * tolerate not being able to
289                                          * translate local hostname
290                                          * to a termid -- it will be
291                                          * "loopback".
292                                          */
293                                         syslog(LOG_AUTH | LOG_ERR,
294                                             "pam_unix_cred: cannot load "
295                                             "ttyname: %m, continuing.");
296                                         goto adt_setuser;
297                                 } else if (errno != 0) {
298                                         syslog(LOG_AUTH | LOG_ERR,
299                                             "pam_unix_cred: cannot load "
300                                             "ttyname: %m.");
301                                 } else {
302                                         syslog(LOG_AUTH | LOG_ERR,
303                                             "pam_unix_cred: cannot load "
304                                             "ttyname.");
305                                 }
306                                 ret = PAM_SYSTEM_ERR;
307                                 goto adt_done;
308                         }
309                 } else {
310                         if (adt_load_hostname(rhost, &termid) != 0) {
311                                 if (errno != 0) {
312                                         syslog(LOG_AUTH | LOG_ERR,
313                                             "pam_unix_cred: cannot load "
314                                             "hostname: %m.");
315                                 } else {
316                                         syslog(LOG_AUTH | LOG_ERR,
317                                             "pam_unix_cred: cannot load "
318                                             "hostname.");
319                                 }
320                                 ret = PAM_SYSTEM_ERR;
321                                 goto adt_done;
322                         }
323                 }
324 adt_setuser:
325                 if ((auser != NULL) && (*auser != '\0') &&
326                     (getpwnam_r(auser, &apwd, apwbuf,
327                     sizeof (apwbuf)) != NULL)) {
328                         /*
329                          * set up the initial audit for user coming
330                          * from another user
331                          */
332                         if (adt_set_user(ah, apwd.pw_uid, apwd.pw_gid,
333                             apwd.pw_uid, apwd.pw_gid, termid, ADT_NEW) != 0) {
334                                 syslog(LOG_AUTH | LOG_ERR,
335                                     "pam_unix_cred: cannot set auser audit "
336                                     "%m");
337                                 ret = PAM_SYSTEM_ERR;
338                                 goto adt_done;
339                         }
340                         if (adt_set_user(ah, pwd.pw_uid, pwd.pw_gid,
```



```
341                             pwd.pw_uid, pwd.pw_gid, NULL,
342                             ADT_UPDATE) != 0) {
343                                 syslog(LOG_AUTH | LOG_ERR,
344                                     "pam_unix_cred: cannot merge user audit "
345                                     "%m");
346                                 ret = PAM_SYSTEM_ERR;
347                                 goto adt_done;
348                         }
349                         if (debug) {
350                                 syslog(LOG_AUTH | LOG_DEBUG,
351                                     "pam_unix_cred: new audit set for %d:%d",
352                                     apwd.pw_uid, pwd.pw_uid);
353                         }
354                 } else {
355                         /*
356                          * No authenticated user or authenticated user is
357                          * not a local user, no remote attribution, set
358                          * up the initial audit as for direct user login
359                          */
360                         if (adt_set_user(ah, pwd.pw_uid, pwd.pw_gid,
361                             pwd.pw_uid, pwd.pw_gid, termid, ADT_NEW) != 0) {
362                                 syslog(LOG_AUTH | LOG_ERR,
363                                     "pam_unix_cred: cannot set user audit %m");
364                                 ret = PAM_SYSTEM_ERR;
365                                 goto adt_done;
366                         }
367                 }
368                 if (adt_set_proc(ah) != 0) {
369                         syslog(LOG_AUTH | LOG_ERR,
370                             "pam_unix_cred: cannot set process audit %m");
371                         ret = PAM_CRED_ERR;
372                         goto adt_done;
373                 }
374                 if (debug) {
375                         syslog(LOG_AUTH | LOG_DEBUG,
376                             "pam_unix_cred: new audit set for %d",
377                             pwd.pw_uid);
378                 }
379         } else if ((auid != AU_NOAUDITID) &&
380             (flags & PAM_REINITIALIZE_CRED)) {
381                 if (adt_set_user(ah, pwd.pw_uid, pwd.pw_gid, pwd.pw_uid,
382                     pwd.pw_gid, NULL, ADT_UPDATE) != 0) {
383                         syslog(LOG_AUTH | LOG_ERR,
384                             "pam_unix_cred: cannot set user audit %m");
385                         ret = PAM_SYSTEM_ERR;
386                         goto adt_done;
387                 }
388                 if (adt_set_proc(ah) != 0) {
389                         syslog(LOG_AUTH | LOG_ERR,
390                             "pam_unix_cred: cannot set process audit %m");
391                         ret = PAM_CRED_ERR;
392                         goto adt_done;
393                 }
394                 if (debug) {
395                         syslog(LOG_AUTH | LOG_DEBUG,
396                             "pam_unix_cred: audit merged for %d:%d",
397                             auid, pwd.pw_uid);
398                 }
399         } else if (debug) {
400                 syslog(LOG_AUTH | LOG_DEBUG,
401                     "pam_unix_cred: audit already set for %d", auid);
402         }
403 adt_done:
404         if (termid != NULL)
405                 free(termid);
406         if (adt_end_session(ah) != 0) {
```

```
407                 syslog(LOG_AUTH | LOG_ERR,
408                     "pam_unix_cred: unable to end audit session");
409         }
410
411         if (ret != PAM_SUCCESS)
412                 return (ret);
413
414         /* Initialize the user's project */
415         (void) pam_get_item(pamh, PAM_RESOURCE, (void **)&kvs);
416         if (kvs != NULL) {
417                 char *tmp, *lasts, *tok;
418
419                 kvs = tmp = strdup(kvs);
420                 if (kvs == NULL)
421                         return (PAM_BUF_ERR);
422
423                 while ((tok = strtok_r(tmp, ";", &lasts)) != NULL) {
424                         if (strncmp(tok, PROJECT, PROJSZ) == 0) {
425                                 projname = tok + PROJSZ;
426                                 break;
427                         }
428                         tmp = NULL;
429                 }
430         } else {
431                 projname = NULL;
432         }
433
434         if (projname == NULL || *projname == '\0') {
435                 pproj = getdefaultproj(user, &proj, (void *)&buf,
436                     PROJECT_BUFSZ);
437         } else {
438                 pproj = getprojbyname(projname, &proj, (void *)&buf,
439                     PROJECT_BUFSZ);
440         }
441         /* projname points into kvs, so this is the first opportunity to free */
442         if (kvs != NULL)
443                 free(kvs);
444         if (pproj == NULL) {
445                 syslog(LOG_AUTH | LOG_ERR,
446                     "pam_unix_cred: no default project for user %s", user);
447                 if (!nowarn) {
448                         (void) snprintf(messages[0], sizeof (messages[0]),
449                             dgettext(TEXT_DOMAIN, "No default project!"));
450                         (void) __pam_display_msg(pamh, PAM_ERROR_MSG,
451                             1, messages, NULL);
452                 }
453                 return (PAM_SYSTEM_ERR);
454         }
455         if ((error = setproject(proj.pj_name, user, TASK_NORMAL)) != 0) {
456                 kva_t *kv_array;
457
458                 switch (error) {
459                 case SETPROJ_ERR_TASK:
460                         if (errno == EAGAIN) {
461                                 syslog(LOG_AUTH | LOG_ERR,
462                                     "pam_unix_cred: project \"%s\" resource "
463                                     "control limit has been reached",
464                                     proj.pj_name);
465                                 (void) snprintf(messages[0],
466                                     sizeof (messages[0]), dgettext(
467                                     TEXT_DOMAIN,
468                                     "Resource control limit has been "
469                                     "reached"));
470                         } else {
471                                 syslog(LOG_AUTH | LOG_ERR,
472                                     "pam_unix_cred: user %s could not join "
```



```
473                                     "project \"%s\": %m", user, proj.pj_name);
474                                 (void) snprintf(messages[0],
475                                     sizeof (messages[0]), dgettext(
476                                     TEXT_DOMAIN,
477                                     "Could not join default project"));
478                         }
479                         if (!nowarn)
480                                 (void) __pam_display_msg(pamh, PAM_ERROR_MSG, 1,
481                                     messages, NULL);
482                         break;
483                 case SETPROJ_ERR_POOL:
484                         (void) snprintf(messages[0], sizeof (messages[0]),
485                             dgettext(TEXT_DOMAIN,
486                             "Could not bind to resource pool"));
487                         switch (errno) {
488                         case EACCES:
489                                 syslog(LOG_AUTH | LOG_ERR,
490                                     "pam_unix_cred: project \"%s\" could not "
491                                     "bind to resource pool: No resource pool "
492                                     "accepting default bindings exists",
493                                     proj.pj_name);
494                                 (void) snprintf(messages[1],
495                                     sizeof (messages[1]),
496                                     dgettext(TEXT_DOMAIN,
497                                     "No resource pool accepting "
498                                     "default bindings exists"));
499                                 break;
500                         case ESRCH:
501                                 syslog(LOG_AUTH | LOG_ERR,
502                                     "pam_unix_cred: project \"%s\" could not "
503                                     "bind to resource pool: The resource pool "
504                                     "is unknown", proj.pj_name);
505                                 (void) snprintf(messages[1],
506                                     sizeof (messages[1]),
507                                     dgettext(TEXT_DOMAIN,
508                                     "The specified resource pool "
509                                     "is unknown"));
510                                 break;
511                         default:
512                                 (void) snprintf(messages[1],
513                                     sizeof (messages[1]),
514                                     dgettext(TEXT_DOMAIN,
515                                     "Failure during pool binding"));
516                                 syslog(LOG_AUTH | LOG_ERR,
517                                     "pam_unix_cred: project \"%s\" could not "
518                                     "bind to resource pool: %m", proj.pj_name);
519                         }
520                         if (!nowarn)
521                                 (void) __pam_display_msg(pamh, PAM_ERROR_MSG,
522                                     2, messages, NULL);
523                         break;
524                 default:
525                         /*
526                          * Resource control assignment failed.  Unlike
527                          * newtask(1m), we treat this as an error.
528                          */
529                         if (error < 0) {
530                                 /*
531                                  * This isn't supposed to happen, but in
532                                  * case it does, this error message
533                                  * doesn't use error as an index, like
534                                  * the others might.
535                                  */
536                                 syslog(LOG_AUTH | LOG_ERR,
537                                     "pam_unix_cred: unkwown error joining "
538                                     "project \"%s\" (%d)", proj.pj_name, error);
```

```
539                                 (void) snprintf(messages[0],
540                                     sizeof (messages[0]),
541                                     dgettext(TEXT_DOMAIN,
542                                     "unkwown error joining project \"%s\""
543                                     " (%d)"), proj.pj_name, error);
544                         } else if ((kv_array = _str2kva(proj.pj_attr, KV_ASSIGN,
545                             KV_DELIMITER)) != NULL) {
546                                 syslog(LOG_AUTH | LOG_ERR,
547                                     "pam_unix_cred: %s resource control "
548                                     "assignment failed for project \"%s\"",
549                                     kv_array->data[error - 1].key,
550                                     proj.pj_name);
551                                 (void) snprintf(messages[0],
552                                     sizeof (messages[0]),
553                                     dgettext(TEXT_DOMAIN,
554                                     "%s resource control assignment failed for "
555                                     "project \"%s\""),
556                                     kv_array->data[error - 1].key,
557                                     proj.pj_name);
558                                 _kva_free(kv_array);
559                         } else {
560                                 syslog(LOG_AUTH | LOG_ERR,
561                                     "pam_unix_cred: resource control "
562                                     "assignment failed for project \"%s\""
563                                     "attribute %d", proj.pj_name, error);
564                                 (void) snprintf(messages[0],
565                                     sizeof (messages[0]),
566                                     dgettext(TEXT_DOMAIN,
567                                     "resource control assignment failed for "
568                                     "project \"%s\" attribute %d"),
569                                     proj.pj_name, error);
570                         }
571                         if (!nowarn)
572                                 (void) __pam_display_msg(pamh, PAM_ERROR_MSG,
573                                     1, messages, NULL);
574                 }
575                 return (PAM_SYSTEM_ERR);
576         }

578         tset = def = lim = NULL;
579         deflim.def = deflim.lim = NULL;

581         (void) _enum_attrs(user, finddeflim, NULL, &deflim);

583         if (getset(deflim.lim, &lim) != 0 || getset(deflim.def, &def) != 0) {
584                 ret = PAM_SYSTEM_ERR;
585                 goto out;
586         }

588         if (def == NULL) {
589                 def = priv_allocset();
590                 if (def == NULL) {
591                         ret = PAM_SYSTEM_ERR;
592                         goto out;
593                 }
594                 priv_basicset(def);
595                 (void) priv_addset(def, PRIV_PROC_SECFLAGS);
596 #endif /* ! codereview */
597                 errno = 0;
598                 if ((pathconf("/", _PC_CHOWN_RESTRICTED) == -1) && (errno == 0))
599                         (void) priv_addset(def, PRIV_FILE_CHOWN_SELF);
600         }
601         /*
602          * Silently limit the privileges to those actually available
603          * in the current zone.
604          */
```



```
605         tset = priv_allocset();
606         if (tset == NULL) {
607                 ret = PAM_SYSTEM_ERR;
608                 goto out;
609         }
610         if (getppriv(PRIV_PERMITTED, tset) != 0) {
611                 ret = PAM_SYSTEM_ERR;
612                 goto out;
613         }
614         if (!priv_issubset(def, tset))
615                 priv_intersect(tset, def);
616         /*
617          * We set privilege awareness here so that I gets copied to
618          * P & E when the final setuid(uid) happens.
619          */
620         (void) setpflags(PRIV_AWARE, 1);
621         if (setppriv(PRIV_SET, PRIV_INHERITABLE, def) != 0) {
622                 syslog(LOG_AUTH | LOG_ERR,
623                     "pam_setcred: setppriv(defaultpriv) failed: %m");
624                 ret = PAM_CRED_ERR;
625         }

627         if (lim != NULL) {
628                 /*
629                  * Silently limit the privileges to the limit set available.
630                  */
631                 if (getppriv(PRIV_LIMIT, tset) != 0) {
632                         ret = PAM_SYSTEM_ERR;
633                         goto out;
634                 }
635                 if (!priv_issubset(lim, tset))
636                         priv_intersect(tset, lim);
637                 if (setppriv(PRIV_SET, PRIV_LIMIT, lim) != 0) {
638                         syslog(LOG_AUTH | LOG_ERR,
639                             "pam_setcred: setppriv(limitpriv) failed: %m");
640                         ret = PAM_CRED_ERR;
641                         goto out;
642                 }
643                 /*
644                  * In order not to surprise certain applications, we
645                  * need to get rid of privilege awareness and thus we must
646                  * set this flag which will cause a reset on set*uid().
647                  */
648                 (void) setpflags(PRIV_AWARE_RESET, 1);
649         }
650         /*
651          * This may fail but we do not care as this will be reset later
652          * when the uids are set to their final values.
653          */
654         (void) setpflags(PRIV_AWARE, 0);
655         /*
656          * Remove PRIV_PFEXEC; stop running as if we are under a profile
657          * shell.  A user with a profile shell will set PRIV_PFEXEC.
658          */
659         (void) setpflags(PRIV_PFEXEC, 0);

661 out:
662         free(deflim.lim);
663         free(deflim.def);

665         if (lim != NULL)
666                 priv_freeset(lim);
667         if (def != NULL)
668                 priv_freeset(def);
669         if (tset != NULL)
670                 priv_freeset(tset);
```

```
672         return (ret);
673 }
```

```
**********************************************************
   12562 Wed May 27 19:49:17 2015
new/usr/src/man/man1/Makefile
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
```

```
#
# This file and its contents are supplied under the terms of the
# Common Development and Distribution License ("CDDL"), version 1.0.
# You may only use this file in accordance with the terms of version
# 1.0 of the CDDL.
#
#
# Copyright 2011, Richard Lowe
# Copyright 2015 Nexenta Systems, Inc. All rights reserved.
# Copyright 2014 Garrett D'Amore <garrett@damore.org>
#

include         $(SRC)/Makefile.master

MANSECT=        1

MANFILES=       acctcom.1                       \
                adb.1                           \
                addbib.1                        \
                alias.1                         \
                allocate.1                      \
                amt.1                           \
                appcert.1                       \
                apptrace.1                      \
                apropos.1                       \
                ar.1                            \
                arch.1                          \
                asa.1                           \
                at.1                            \
                atq.1                           \
                atrm.1                          \
                audioconvert.1                  \
                audioctl.1                      \
                audioplay.1                     \
                audiorecord.1                   \
                audiotest.1                     \
                auths.1                         \
                awk.1                           \
                banner.1                        \
                basename.1                      \
                bc.1                            \
                bdiff.1                         \
                break.1                         \
                builtin.1                       \
                cal.1                           \
                calendar.1                      \
                cancel.1                        \
                cat.1                           \
                cd.1                            \
                cdrw.1                          \
                checknr.1                       \
                chgrp.1                         \
                chkey.1                         \
```



```
                chmod.1                         \
                chown.1                         \
                ckdate.1                        \
                ckgid.1                         \
                ckint.1                         \
                ckitem.1                        \
                ckkeywd.1                       \
                ckpath.1                        \
                ckrange.1                       \
                ckstr.1                         \
                cksum.1                         \
                cktime.1                        \
                ckuid.1                         \
                ckyorn.1                        \
                clear.1                         \
                cmp.1                           \
                col.1                           \
                comm.1                          \
                command.1                       \
                compress.1                      \
                cp.1                            \
                cpio.1                          \
                cputrack.1                      \
                crle.1                          \
                crontab.1                       \
                crypt.1                         \
                csh.1                           \
                csplit.1                        \
                ctags.1                         \
                ctrun.1                         \
                ctstat.1                        \
                ctwatch.1                       \
                cut.1                           \
                date.1                          \
                dc.1                            \
                deallocate.1                    \
                deroff.1                        \
                dhcpinfo.1                      \
                diff.1                          \
                diff3.1                         \
                diffmk.1                        \
                digest.1                        \
                dircmp.1                        \
                dis.1                           \
                disown.1                        \
                dispgid.1                       \
                dispuid.1                       \
                dos2unix.1                      \
                download.1                      \
                dpost.1                         \
                du.1                            \
                dump.1                          \
                dumpcs.1                        \
                echo.1                          \
                ed.1                            \
                egrep.1                         \
                eject.1                         \
                elfdump.1                       \
                elfedit.1                       \
                elfsign.1                       \
                elfwrap.1                       \
                enable.1                        \
                encrypt.1                       \
                enhance.1                       \
                env.1                           \
                eqn.1                           \
```

```
125                exec.1                        \
126                exit.1                        \
127                expand.1                      \
128                expr.1                        \
129                exstr.1                       \
130                factor.1                      \
131                fdformat.1                    \
132                fgrep.1                       \
133                file.1                        \
134                filesync.1                    \
135                find.1                        \
136                finger.1                      \
137                fmt.1                         \
138                fmtmsg.1                      \
139                fold.1                        \
140                ftp.1                         \
141                gcore.1                       \
142                gencat.1                      \
143                genmsg.1                      \
144                getconf.1                     \
145                getfacl.1                     \
146                getlabel.1                    \
147                getopt.1                      \
148                getoptcvt.1                   \
149                getopts.1                     \
150                gettext.1                     \
151                gettxt.1                      \
152                getzonepath.1                 \
153                glob.1                        \
154                gprof.1                       \
155                grep.1                        \
156                groups.1                      \
157                hash.1                        \
158                head.1                        \
159                history.1                     \
160                hostid.1                      \
161                hostname.1                    \
162                iconv.1                       \
163                indxbib.1                     \
164                Intro.1                       \
165                ipcrm.1                       \
166                ipcs.1                        \
167                isainfo.1                     \
168                isalist.1                     \
169                jobs.1                        \
170                join.1                        \
171                kbd.1                         \
172                kdestroy.1                    \
173                keylogin.1                    \
174                keylogout.1                   \
175                kill.1                        \
176                kinit.1                       \
177                klist.1                       \
178                kmdb.1                        \
179                kmfcfg.1                      \
180                kpasswd.1                     \
181                krb5-config.1                 \
182                ksh93.1                       \
183                ktutil.1                      \
184                kvmstat.1                     \
185                lari.1                        \
186                last.1                        \
187                lastcomm.1                    \
188                ld.1                          \
189                ldap.1                        \
190                ldapdelete.1                  \
```



```
191                ldaplist.1                    \
192                ldapmodify.1                  \
193                ldapmodrdn.1                  \
194                ldapsearch.1                  \
195                ldd.1                         \
196                ld.so.1.1                     \
197                let.1                         \
198                lex.1                         \
199                lgrpinfo.1                    \
200                limit.1                       \
201                line.1                        \
202                list_devices.1                \
203                listusers.1                   \
204                ln.1                          \
205                loadkeys.1                    \
206                locale.1                      \
207                localedef.1                   \
208                logger.1                      \
209                login.1                       \
210                logname.1                     \
211                logout.1                      \
212                look.1                        \
213                lookbib.1                     \
214                lorder.1                      \
215                lp.1                          \
216                lpstat.1                      \
217                ls.1                          \
218                m4.1                          \
219                mac.1                         \
220                mach.1                        \
221                madv.so.1.1                   \
222                mail.1                        \
223                mailcompat.1                  \
224                mailq.1                       \
225                mailstats.1                   \
226                mailx.1                       \
227                makekey.1                     \
228                man.1                         \
229                mandoc.1                      \
230                mconnect.1                    \
231                mcs.1                         \
232                mdb.1                         \
233                mesg.1                        \
234                mkdir.1                       \
235                mkmsgs.1                      \
236                mktemp.1                      \
237                moe.1                         \
238                more.1                        \
239                mpss.so.1.1                   \
240                msgcc.1                       \
241                msgcpp.1                      \
242                msgcvt.1                      \
243                msgfmt.1                      \
244                msggen.1                      \
245                msgget.1                      \
246                mt.1                          \
247                mv.1                          \
248                nawk.1                        \
249                nc.1                          \
250                nca.1                         \
251                ncab2clf.1                    \
252                ncakmod.1                     \
253                newform.1                     \
254                newgrp.1                      \
255                news.1                        \
256                newtask.1                     \
```

```
257                 nice.1                          \
258                 nl.1                            \
259                 nm.1                            \
260                 nohup.1                         \
261                 nroff.1                         \
262                 od.1                            \
263                 on.1                            \
264                 optisa.1                        \
265                 pack.1                          \
266                 pagesize.1                      \
267                 pargs.1                         \
268                 passwd.1                        \
269                 paste.1                         \
270                 pathchk.1                       \
271                 pax.1                           \
272                 pfexec.1                        \
273                 pg.1                            \
274                 pgrep.1                         \
275                 pkginfo.1                       \
276                 pkgmk.1                         \
277                 pkgparam.1                      \
278                 pkgproto.1                      \
279                 pkgtrans.1                      \
280                 pktool.1                        \
281                 plabel.1                        \
282                 plgrp.1                         \
283                 plimit.1                        \
284                 pmadvise.1                      \
285                 pmap.1                          \
286                 postio.1                        \
287                 postprint.1                     \
288                 postreverse.1                   \
289                 ppgsz.1                         \
290                 ppriv.1                         \
291                 pr.1                            \
292                 praliases.1                     \
293                 prctl.1                         \
294                 preap.1                         \
295                 prex.1                          \
296                 print.1                         \
297                 printf.1                        \
298                 priocntl.1                      \
299                 proc.1                          \
300                 prof.1                          \
301                 profiles.1                      \
302                 projects.1                      \
303                 ps.1                            \
304                 psecflags.1                     \
305 #endif /* ! codereview */
306                 ptree.1                         \
307                 pvs.1                           \
308                 pwd.1                           \
309                 ranlib.1                        \
310                 rcapstat.1                      \
311                 rcp.1                           \
312                 rdist.1                         \
313                 read.1                          \
314                 readonly.1                      \
315                 refer.1                         \
316                 regcmp.1                        \
317                 renice.1                        \
318                 rev.1                           \
319                 rlogin.1                        \
320                 rm.1                            \
321                 rmformat.1                      \
322                 rmmount.1                       \
```



```
323                 roffbib.1                       \
324                 roles.1                         \
325                 rpcgen.1                        \
326                 rsh.1                           \
327                 runat.1                         \
328                 rup.1                           \
329                 ruptime.1                       \
330                 rusers.1                        \
331                 rwho.1                          \
332                 sar.1                           \
333                 scp.1                           \
334                 script.1                        \
335                 sdiff.1                         \
336                 sed.1                           \
337                 set.1                           \
338                 setfacl.1                       \
339                 setlabel.1                      \
340                 setpgrp.1                       \
341                 sftp.1                          \
342                 shcomp.1                        \
343                 shell_builtins.1                \
344                 shift.1                         \
345                 size.1                          \
346                 sleep.1                         \
347                 smbutil.1                       \
348                 soelim.1                        \
349                 sort.1                          \
350                 sortbib.1                       \
351                 sotruss.1                       \
352                 spell.1                         \
353                 split.1                         \
354                 srchtxt.1                       \
355                 ssh.1                           \
356                 ssh-add.1                       \
357                 ssh-agent.1                     \
358                 ssh-http-proxy-connect.1        \
359                 ssh-keygen.1                    \
360                 ssh-keyscan.1                   \
361                 ssh-socks5-proxy-connect.1      \
362                 strchg.1                        \
363                 strings.1                       \
364                 strip.1                         \
365                 stty.1                          \
366                 sum.1                           \
367                 suspend.1                       \
368                 svcprop.1                       \
369                 svcs.1                          \
370                 symorder.1                      \
371                 sys-suspend.1                   \
372                 tabs.1                          \
373                 tail.1                          \
374                 talk.1                          \
375                 tar.1                           \
376                 tbl.1                           \
377                 tcopy.1                         \
378                 tee.1                           \
379                 telnet.1                        \
380                 test.1                          \
381                 tftp.1                          \
382                 time.1                          \
383                 times.1                         \
384                 timex.1                         \
385                 tip.1                           \
386                 tnfdump.1                       \
387                 tnfxtract.1                     \
388                 touch.1                         \
```

```
389                 tput.1                          \
390                 tr.1                            \
391                 trap.1                          \
392                 troff.1                         \
393                 true.1                          \
394                 truss.1                         \
395                 tsort.1                         \
396                 tty.1                           \
397                 type.1                          \
398                 typeset.1                       \
399                 ul.1                            \
400                 umask.1                         \
401                 uname.1                         \
402                 unifdef.1                       \
403                 uniq.1                          \
404                 units.1                         \
405                 unix2dos.1                      \
406                 uptime.1                        \
407                 vacation.1                      \
408                 vgrind.1                        \
409                 volcheck.1                      \
410                 volrmmount.1                    \
411                 w.1                             \
412                 wait.1                          \
413                 wc.1                            \
414                 which.1                         \
415                 who.1                           \
416                 whocalls.1                      \
417                 whois.1                         \
418                 write.1                         \
419                 xargs.1                         \
420                 xgettext.1                      \
421                 xstr.1                          \
422                 yacc.1                          \
423                 yes.1                           \
424                 ypcat.1                         \
425                 ypmatch.1                       \
426                 yppasswd.1                      \
427                 ypwhich.1                       \
428                 zlogin.1                        \
429                 zonename.1

431 MANLINKS=       batch.1         \
432                 bg.1            \
433                 case.1          \
434                 chdir.1         \
435                 checkeq.1       \
436                 continue.1      \
437                 decrypt.1       \
438                 dirname.1       \
439                 dirs.1          \
440                 disable.1       \
441                 dumpkeys.1      \
442                 edit.1          \
443                 errange.1       \
444                 errdate.1       \
445                 errgid.1        \
446                 errint.1        \
447                 erritem.1       \
448                 errpath.1       \
449                 errstr.1        \
450                 errtime.1       \
451                 erruid.1        \
452                 erryorn.1       \
453                 eval.1          \
454                 export.1        \
```



```
455                 false.1         \
456                 fc.1            \
457                 fg.1            \
458                 for.1           \
459                 foreach.1       \
460                 function.1      \
461                 goto.1          \
462                 hashcheck.1     \
463                 hashmake.1      \
464                 hashstat.1      \
465                 helpdate.1      \
466                 helpgid.1       \
467                 helpint.1       \
468                 helpitem.1      \
469                 helppath.1      \
470                 helprange.1     \
471                 helpstr.1       \
472                 helptime.1      \
473                 helpuid.1       \
474                 helpyorn.1      \
475                 hist.1          \
476                 if.1            \
477                 intro.1         \
478                 jsh.1           \
479                 ksh.1           \
480                 ldapadd.1       \
481                 neqn.1          \
482                 notify.1        \
483                 onintr.1        \
484                 page.1          \
485                 pcat.1          \
486                 pcred.1         \
487                 pfcsh.1         \
488                 pfiles.1        \
489                 pfksh.1         \
490                 pflags.1        \
491                 pfsh.1          \
492                 pkill.1         \
493                 pldd.1          \
494                 popd.1          \
495                 prun.1          \
496                 psig.1          \
497                 pstack.1        \
498                 pstop.1         \
499                 ptime.1         \
500                 pushd.1         \
501                 pwait.1         \
502                 pwdx.1          \
503                 red.1           \
504                 rehash.1        \
505                 remote_shell.1  \
506                 remsh.1         \
507                 repeat.1        \
508                 return.1        \
509                 rksh.1          \
510                 rksh93.1        \
511                 rmail.1         \
512                 rmdir.1         \
513                 rmumount.1      \
514                 select.1        \
515                 setenv.1        \
516                 settime.1       \
517                 sh.1            \
518                 snca.1          \
519                 source.1        \
520                 spellin.1       \
```

```
521                 stop.1          \
522                 strconf.1       \
523                 switch.1        \
524                 ulimit.1        \
525                 unalias.1       \
526                 uncompress.1    \
527                 unexpand.1      \
528                 unhash.1        \
529                 unlimit.1       \
530                 unpack.1        \
531                 unset.1         \
532                 unsetenv.1      \
533                 until.1         \
534                 valdate.1       \
535                 valgid.1        \
536                 valint.1        \
537                 valpath.1       \
538                 valrange.1      \
539                 valstr.1        \
540                 valtime.1       \
541                 valuid.1        \
542                 valyorn.1       \
543                 vedit.1         \
544                 whatis.1        \
545                 whence.1        \
546                 while.1         \
547                 zcat.1

549 intro.1          := LINKSRC = Intro.1

551 whatis.1         := LINKSRC = apropos.1

553 unalias.1        := LINKSRC = alias.1

555 batch.1          := LINKSRC = at.1

557 dirname.1        := LINKSRC = basename.1

559 continue.1       := LINKSRC = break.1

561 chdir.1          := LINKSRC = cd.1
562 dirs.1           := LINKSRC = cd.1
563 popd.1           := LINKSRC = cd.1
564 pushd.1          := LINKSRC = cd.1

566 errdate.1        := LINKSRC = ckdate.1
567 helpdate.1      := LINKSRC = ckdate.1
568 valdate.1       := LINKSRC = ckdate.1

570 errgid.1         := LINKSRC = ckgid.1
571 helpgid.1       := LINKSRC = ckgid.1
572 valgid.1        := LINKSRC = ckgid.1

574 errint.1         := LINKSRC = ckint.1
575 helpint.1       := LINKSRC = ckint.1
576 valint.1        := LINKSRC = ckint.1

578 erritem.1        := LINKSRC = ckitem.1
579 helpitem.1      := LINKSRC = ckitem.1

581 errpath.1        := LINKSRC = ckpath.1
582 helppath.1      := LINKSRC = ckpath.1
583 valpath.1       := LINKSRC = ckpath.1

585 errange.1        := LINKSRC = ckrange.1
586 helprange.1     := LINKSRC = ckrange.1
```



```
587 valrange.1      := LINKSRC = ckrange.1

589 errstr.1        := LINKSRC = ckstr.1
590 helpstr.1       := LINKSRC = ckstr.1
591 valstr.1        := LINKSRC = ckstr.1

593 errtime.1       := LINKSRC = cktime.1
594 helptime.1      := LINKSRC = cktime.1
595 valtime.1       := LINKSRC = cktime.1

597 erruid.1        := LINKSRC = ckuid.1
598 helpuid.1       := LINKSRC = ckuid.1
599 valuid.1        := LINKSRC = ckuid.1

601 erryorn.1       := LINKSRC = ckyorn.1
602 helpyorn.1      := LINKSRC = ckyorn.1
603 valyorn.1       := LINKSRC = ckyorn.1

605 uncompress.1    := LINKSRC = compress.1
606 zcat.1          := LINKSRC = compress.1

608 red.1           := LINKSRC = ed.1

610 disable.1       := LINKSRC = enable.1

612 decrypt.1       := LINKSRC = encrypt.1

614 checkeq.1       := LINKSRC = eqn.1
615 neqn.1          := LINKSRC = eqn.1

617 eval.1          := LINKSRC = exec.1
618 source.1        := LINKSRC = exec.1

620 goto.1          := LINKSRC = exit.1
621 return.1        := LINKSRC = exit.1

623 unexpand.1      := LINKSRC = expand.1

625 hashstat.1      := LINKSRC = hash.1
626 rehash.1        := LINKSRC = hash.1
627 unhash.1        := LINKSRC = hash.1

629 fc.1            := LINKSRC = history.1
630 hist.1          := LINKSRC = history.1

632 bg.1            := LINKSRC = jobs.1
633 fg.1            := LINKSRC = jobs.1
634 notify.1        := LINKSRC = jobs.1
635 stop.1          := LINKSRC = jobs.1

637 jsh.1           := LINKSRC = ksh93.1
638 ksh.1           := LINKSRC = ksh93.1
639 rksh.1          := LINKSRC = ksh93.1
640 rksh93.1        := LINKSRC = ksh93.1
641 sh.1            := LINKSRC = ksh93.1

643 ldapadd.1       := LINKSRC = ldapmodify.1

645 ulimit.1        := LINKSRC = limit.1
646 unlimit.1       := LINKSRC = limit.1

648 dumpkeys.1      := LINKSRC = loadkeys.1

650 rmail.1         := LINKSRC = mail.1

652 page.1          := LINKSRC = more.1
```

```
654 snca.1          := LINKSRC = nca.1

656 pcat.1          := LINKSRC = pack.1
657 unpack.1        := LINKSRC = pack.1

659 pfcsh.1         := LINKSRC = pfexec.1
660 pfksh.1         := LINKSRC = pfexec.1
661 pfsh.1          := LINKSRC = pfexec.1

663 pkill.1         := LINKSRC = pgrep.1

665 pcred.1         := LINKSRC = proc.1
666 pfiles.1        := LINKSRC = proc.1
667 pflags.1        := LINKSRC = proc.1
668 pldd.1          := LINKSRC = proc.1
669 prun.1          := LINKSRC = proc.1
670 psig.1          := LINKSRC = proc.1
671 pstack.1        := LINKSRC = proc.1
672 pstop.1         := LINKSRC = proc.1
673 ptime.1         := LINKSRC = proc.1
674 pwait.1         := LINKSRC = proc.1
675 pwdx.1          := LINKSRC = proc.1

677 rmdir.1         := LINKSRC = rm.1

679 rmumount.1      := LINKSRC = rmmount.1

681 remote_shell.1  := LINKSRC = rsh.1
682 remsh.1         := LINKSRC = rsh.1

684 export.1        := LINKSRC = set.1
685 setenv.1        := LINKSRC = set.1
686 unset.1         := LINKSRC = set.1
687 unsetenv.1      := LINKSRC = set.1

689 case.1          := LINKSRC = shell_builtins.1
690 for.1           := LINKSRC = shell_builtins.1
691 foreach.1       := LINKSRC = shell_builtins.1
692 function.1      := LINKSRC = shell_builtins.1
693 if.1            := LINKSRC = shell_builtins.1
694 repeat.1        := LINKSRC = shell_builtins.1
695 select.1        := LINKSRC = shell_builtins.1
696 switch.1        := LINKSRC = shell_builtins.1
697 until.1         := LINKSRC = shell_builtins.1
698 while.1         := LINKSRC = shell_builtins.1

700 hashcheck.1     := LINKSRC = spell.1
701 hashmake.1      := LINKSRC = spell.1
702 spellin.1       := LINKSRC = spell.1

704 strconf.1       := LINKSRC = strchg.1

706 settime.1       := LINKSRC = touch.1

708 onintr.1        := LINKSRC = trap.1

710 false.1         := LINKSRC = true.1

712 whence.1        := LINKSRC = typeset.1

714 # Links to usr/has/man

716 edit.1          := LINKSRC = ../../../has/man/man1has/edit.1has

718 vedit.1         := LINKSRC = ../../../has/man/man1has/vi.1has
```



```
720 .KEEP_STATE:

722 include         $(SRC)/man/Makefile.man

724 install:        $(ROOTMANFILES) $(ROOTMANLINKS)
```

```
**********************************************************
   59600 Wed May 27 19:49:17 2015
new/usr/src/man/man1/ld.1
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
```

```
   1 '\" te
   2 .\" Copyright 1989 AT&T
   3 .\" Copyright (c) 2009, Sun Microsystems, Inc. All Rights Reserved
   4 .\" Copyright (c) 2012, Joyent, Inc. All Rights Reserved
   5 .\" The contents of this file are subject to the terms of the Common Development
   6 .\"  See the License for the specific language governing permissions and limitat
   7 .\" the fields enclosed by brackets "[]" replaced with your own identifying info
   8 .TH LD 1 "Sep 10, 2013"
   9 .SH NAME
  10 ld \- link-editor for object files
  11 .SH SYNOPSIS
  12 .LP
  13 .nf
  14 \fBld\fR [\fB-32\fR | \fB-64\fR] [\fB-a\fR | \fB-r\fR] [\fB-b\fR] [\fB-B\fRdirec
  15 [\fB-B\fR dynamic | static] [\fB-B\fR eliminate] [\fB-B\fR group] [\fB-B\fR loca
  16 [\fB-B\fR reduce] [\fB-B\fR symbolic] [\fB-c\fR \fIname\fR] [\fB-C\fR] [\fB-d\fR
  17 [\fB-D\fR \fItoken\fR,...] [\fB-e\fR \fIepsym\fR] [\fB-f\fR \fIname\fR | \fB-F\f
  18 [\fB-i\fR] [\fB-I\fR \fIname\fR] [\fB-l\fR \fIx\fR] [\fB-L\fR \fIpath\fR] [\fB-m
  19 [\fB-N\fR \fIstring\fR] [\fB-o\fR \fIoutfile\fR] [\fB-p\fR \fIauditlib\fR] [\fB-
  20 [\fB-Q\fR y | n] [\fB-R\fR \fIpath\fR] [\fB-s\fR] [\fB-S\fR \fIsupportlib\fR] [\
  21 [\fB-u\fR \fIsymname\fR] [\fB-V\fR] [\fB-Y P\fR\fI,dirlist\fR] [\fB-z\fR absexec
  22 [\fB-z\fR allextract | defaultextract | weakextract ] [\fB-z\fR altexec64]
  23 [\fB-z\fR aslr[=\fIstate\fR]] [\fB-z\fR assert-deflib] [ \fB-z\fR assert-deflib=
  23 [\fB-z\fR assert-deflib ] [ \fB-z\fR assert-deflib=\fIlibname\fR ]
  24 [\fB-z\fR combreloc | nocombreloc ] [\fB-z\fR defs | nodefs]
  25 [\fB-z\fR direct | nodirect] [\fB-z\fR endfiltee]
  26 [\fB-z\fR fatal-warnings | nofatal-warnings ] [\fB-z\fR finiarray=\fIfunction\fR
  27 [\fB-z\fR globalaudit] [\fB-z\fR groupperm | nogroupperm]
  28 [\fB-z\fR guidance[=\fIid1\fR,\fIid2\fR...] [\fB-z\fR help ]
  29 [\fB-z\fR ignore | record] [\fB-z\fR initarray=\fIfunction\fR] [\fB-z\fR initfir
  30 [\fB-z\fR interpose] [\fB-z\fR lazyload | nolazyload]
  31 [\fB-z\fR ld32=\fIarg1\fR,\fIarg2\fR,...] [\fB-z\fR ld64=\fIarg1\fR,\fIarg2\fR,.
  32 [\fB-z\fR loadfltr] [\fB-z\fR muldefs] [\fB-z\fR nocompstrtab] [\fB-z\fR nodefau
  33 [\fB-z\fR nodelete] [\fB-z\fR nodlopen] [\fB-z\fR nodump] [\fB-z\fR noldynsym]
  34 [\fB-z\fR nopartial] [\fB-z\fR noversion] [\fB-z\fR now] [\fB-z\fR origin]
  35 [\fB-z\fR preinitarray=\fIfunction\fR] [\fB-z\fR redlocsym] [\fB-z\fR relaxreloc
  36 [\fB-z\fR rescan-now] [\fB-z\fR recan] [\fB-z\fR rescan-start \fI\&...\fR \fB-z\
  37 [\fB-z\fR target=sparc|x86] [\fB-z\fR text | textwarn | textoff]
  38 [\fB-z\fR verbose] [\fB-z\fR wrap=\fIsymbol\fR] \fIfilename\fR...
  39 .fi

  41 .SH DESCRIPTION
  42 .sp
  43 .LP
  44 The link-editor, \fBld\fR, combines relocatable object files by resolving
  45 symbol references to symbol definitions, together with performing relocations.
  46 \fBld\fR operates in two modes, static or dynamic, as governed by the \fB-d\fR
  47 option. In all cases, the output of \fBld\fR is left in the file \fBa.out\fR by
  48 default. See NOTES.
  49 .sp
  50 .LP
  51 In dynamic mode, \fB-dy\fR, the default, relocatable object files that are
  52 provided as arguments are combined to produce an executable object file. This
  53 file is linked at execution with any shared object files that are provided as
  54 arguments. If the \fB-G\fR option is specified, relocatable object files are
  55 combined to produce a shared object. Without the \fB-G\fR option, a dynamic
  56 executable is created.
  57 .sp
```



```
  58 .LP
  59 In static mode, \fB-dn\fR, relocatable object files that are provided as
  60 arguments are combined to produce a static executable file. If the \fB-r\fR
  61 option is specified, relocatable object files are combined to produce one
  62 relocatable object file. See \fBStatic Executables\fR.
  63 .sp
  64 .LP
  65 Dynamic linking is the most common model for combining relocatable objects, and
  66 the eventual creation of processes within Solaris. This environment tightly
  67 couples the work of the link-editor and the runtime linker, \fBld.so.1\fR(1).
  68 Both of these utilities, together with their related technologies and
  69 utilities, are extensively documented in the \fILinker and Libraries Guide\fR.
  70 .sp
  71 .LP
  72 If any argument is a library, \fBld\fR by default searches the library exactly
  73 once at the point the library is encountered on the argument list. The library
  74 can be either a shared object or relocatable archive. See \fBar.h\fR(3HEAD)).
  75 .sp
  76 .LP
  77 A shared object consists of an indivisible, whole unit that has been generated
  78 by a previous link-edit of one or more input files. When the link-editor
  79 processes a shared object, the entire contents of the shared object become a
  80 logical part of the resulting output file image. The shared object is not
  81 physically copied during the link-edit as its actual inclusion is deferred
  82 until process execution. This logical inclusion means that all symbol entries
  83 defined in the shared object are made available to the link-editing process.
  84 See Chapter 4, \fIShared Objects,\fR in \fILinker and Libraries Guide\fR
  85 .sp
  86 .LP
  87 For an archive library, \fBld\fR loads only those routines that define an
  88 unresolved external reference. \fBld\fR searches the symbol table of the
  89 archive library sequentially to resolve external references that can be
  90 satisfied by library members. This search is repeated until no external
  91 references can be resolved by the archive. Thus, the order of members in the
  92 library is functionally unimportant, unless multiple library members exist that
  93 define the same external symbol. Archive libraries that have interdependencies
  94 can require multiple command line definitions, or the use of one of the
  95 \fB-z\fR \fBrescan\fR options. See \fIArchive Processing\fR in \fILinker and
  96 Libraries Guide\fR.
  97 .sp
  98 .LP
  99 \fBld\fR is a cross link-editor, able to link 32-bit objects or 64-bit objects,
 100 for Sparc or x86 targets. \fBld\fR uses the \fBELF\fR class and machine type of
 101 the first relocatable object on the command line to govern the mode in which to
 102 operate. The mixing of 32-bit objects and 64-bit objects is not permitted.
 103 Similarly, only objects of a single machine type are allowed. See the
 104 \fB-32\fR, \fB-64\fR and \fB-z target\fR options, and the \fBLD_NOEXEC_64\fR
 105 environment variable.
 106 .SS "Static Executables"
 107 .sp
 108 .LP
 109 The creation of static executables has been discouraged for many releases. In
 110 fact, 64-bit system archive libraries have never been provided. Because a
 111 static executable is built against system archive libraries, the executable
 112 contains system implementation details. This self-containment has a number of
 113 drawbacks.
 114 .RS +4
 115 .TP
 116 .ie t \(bu
 117 .el o
 118 The executable is immune to the benefits of system patches delivered as shared
 119 objects. The executable therefore, must be rebuilt to take advantage of many
 120 system improvements.
 121 .RE
 122 .RS +4
 123 .TP
```

```
.ie t \(bu
.el o
The ability of the executable to run on future releases can be compromised.
.RE
.RS +4
.TP
.ie t \(bu
.el o
The duplication of system implementation details negatively affects system
performance.
.RE
.sp
.LP
With Solaris 10, 32-bit system archive libraries are no longer provided.
Without these libraries, specifically \fBlibc.a\fR, the creation of static
executables is no longer achievable without specialized system knowledge.
However, the capability of \fBld\fR to process static linking options, and the
processing of archive libraries, remains unchanged.
.SH OPTIONS
.sp
.LP
The following options are supported.
.sp
.ne 2
.na
\fB\fB-32\fR | \fB-64\fR\fR
.ad
.sp .6
.RS 4n
Creates a 32-bit, or 64-bit object.
.sp
By default, the class of the object being generated is determined from the
first \fBELF\fR object processed from the command line. If no objects are
specified, the class is determined by the first object encountered within the
first archive processed from the command line. If there are no objects or
archives, the link-editor creates a 32-bit object.
.sp
The \fB-64\fR option is required to create a 64-bit object solely from a
mapfile.
.sp
This \fB-32\fR or \fB-64\fR options can also be used in the rare case of
linking entirely from an archive that contains a mixture of 32 and 64-bit
objects. If the first object in the archive is not the class of the object that
is required to be created, then the \fB-32\fR or \fB-64\fR option can be used
to direct the link-editor. See \fIThe 32-bit link-editor and 64-bit
link-editor\fR in \fILinker and Libraries Guide\fR.
.RE

.sp
.ne 2
.na
\fB\fB-a\fR\fR
.ad
.sp .6
.RS 4n
In static mode only, produces an executable object file. Undefined references
are not permitted. This option is the default behavior for static mode. The
\fB-a\fR option can not be used with the \fB-r\fR option. See \fBStatic
Executables\fR under DESCRIPTION.
.RE

.sp
.ne 2
.na
\fB\fB-b\fR\fR
.ad
```



```
.sp .6
.RS 4n
In dynamic mode only, provides no special processing for dynamic executable
relocations that reference symbols in shared objects. Without the \fB-b\fR
option, the link-editor applies techniques within a dynamic executable so that
the text segment can remain read-only. One technique is the creation of special
position-independent relocations for references to functions that are defined
in shared objects. Another technique arranges for data objects that are defined
in shared objects to be copied into the memory image of an executable at
runtime.
.sp
The \fB-b\fR option is intended for specialized dynamic objects and is not
recommended for general use. Its use suppresses all specialized processing
required to ensure an object's shareability, and can even prevent the
relocation of 64-bit executables.
.RE

.sp
.ne 2
.na
\fB\fB-B\fR \fBdirect\fR | \fBnodirect\fR\fR
.ad
.sp .6
.RS 4n
These options govern direct binding. \fB-B\fR \fBdirect\fR establishes direct
binding information by recording the relationship between each symbol reference
together with the dependency that provides the definition. In addition, direct
binding information is established between each symbol reference and an
associated definition within the object being created. The runtime linker uses
this information to search directly for a symbol in the associated object
rather than to carry out a default symbol search.
.sp
Direct binding information can only be established to dependencies specified
with the link-edit. Thus, you should use the \fB-z\fR \fBdefs\fR option.
Objects that wish to interpose on symbols in a direct binding environment
should identify themselves as interposers with the \fB-z\fR \fBinterpose\fR
option. The use of \fB-B\fR \fBdirect\fR enables \fB-z\fR \fBlazyload\fR for
all dependencies.
.sp
The \fB-B\fR \fBnodirect\fR option prevents any direct binding to the
interfaces offered by the object being created. The object being created can
continue to directly bind to external interfaces by specifying the \fB-z\fR
\fBdirect\fR option. See Appendix D, \fIDirect Bindings,\fR in \fILinker and
Libraries Guide\fR.
.RE

.sp
.ne 2
.na
\fB\fB-B\fR \fBdynamic\fR | \fBstatic\fR\fR
.ad
.sp .6
.RS 4n
Options governing library inclusion. \fB-B\fR \fBdynamic\fR is valid in dynamic
mode only. These options can be specified any number of times on the command
line as toggles: if the \fB-B\fR \fBstatic\fR option is given, no shared
objects are accepted until \fB-B\fR \fBdynamic\fR is seen. See the \fB-l\fR
option.
.RE

.sp
.ne 2
.na
\fB\fB-B\fR \fBeliminate\fR\fR
.ad
.sp .6
```

```
256 .RS 4n
257 Causes any global symbols, not assigned to a version definition, to be
258 eliminated from the symbol table. Version definitions can be supplied by means
259 of a \fBmapfile\fR to indicate the global symbols that should remain visible in
260 the generated object. This option achieves the same symbol elimination as the
261 \fIauto-elimination\fR directive that is available as part of a \fBmapfile\fR
262 version definition. This option can be useful when combining versioned and
263 non-versioned relocatable objects. See also the \fB-B\fR \fBlocal\fR option and
264 the \fB-B\fR \fBreduce\fR option. See \fIDefining Additional Symbols with a
265 mapfile\fR in \fILinker and Libraries Guide\fR.
266 .RE

268 .sp
269 .ne 2
270 .na
271 \fB\fB-B\fR \fBgroup\fR\fR
272 .ad
273 .sp .6
274 .RS 4n
275 Establishes a shared object and its dependencies as a group. Objects within the
276 group are bound to other members of the group at runtime. This mode is similar
277 to adding the object to the process by using \fBdlopen\fR(3C) with the
278 \fBRTLD_GROUP\fR mode. An object that has an explicit dependency on a object
279 identified as a group, becomes a member of the group.
280 .sp
281 As the group must be self contained, use of the \fB-B\fR \fBgroup\fR option
282 also asserts the \fB-z\fR \fBdefs\fR option.
283 .RE

285 .sp
286 .ne 2
287 .na
288 \fB\fB-B\fR \fBlocal\fR\fR
289 .ad
290 .sp .6
291 .RS 4n
292 Causes any global symbols, not assigned to a version definition, to be reduced
293 to local. Version definitions can be supplied by means of a \fBmapfile\fR to
294 indicate the global symbols that should remain visible in the generated object.
295 This option achieves the same symbol reduction as the \fIauto-reduction\fR
296 directive that is available as part of a \fBmapfile\fR version definition. This
297 option can be useful when combining versioned and non-versioned relocatable
298 objects. See also the \fB-B\fR \fBeliminate\fR option and the \fB-B\fR
299 \fBreduce\fR option. See \fIDefining Additional Symbols with a mapfile\fR in
300 \fILinker and Libraries Guide\fR.
301 .RE

303 .sp
304 .ne 2
305 .na
306 \fB\fB-B\fR \fBreduce\fR\fR
307 .ad
308 .sp .6
309 .RS 4n
310 When generating a relocatable object, causes the reduction of symbolic
311 information defined by any version definitions. Version definitions can be
312 supplied by means of a \fBmapfile\fR to indicate the global symbols that should
313 remain visible in the generated object. By default, when a relocatable object
314 is generated, version definitions are only recorded in the output image. The
315 actual reduction of symbolic information is carried out when the object is used
316 in the construction of a dynamic executable or shared object. The \fB-B\fR
317 \fBreduce\fR option is applied automatically when a dynamic executable or
318 shared object is created.
319 .RE

321 .sp
```



```
322 .ne 2
323 .na
324 \fB\fB-B\fR \fBsymbolic\fR\fR
325 .ad
326 .sp .6
327 .RS 4n
328 In dynamic mode only. When building a shared object, binds references to global
329 symbols to their definitions, if available, within the object. Normally,
330 references to global symbols within shared objects are not bound until runtime,
331 even if definitions are available. This model allows definitions of the same
332 symbol in an executable or other shared object to override the object's own
333 definition. \fBld\fR issues warnings for undefined symbols unless \fB-z\fR
334 \fBdefs\fR overrides.
335 .sp
336 The \fB-B\fR \fBsymbolic\fR option is intended for specialized dynamic objects
337 and is not recommended for general use. To reduce the runtime relocation
338 processing that is required an object, the creation of a version definition is
339 recommended.
340 .RE

342 .sp
343 .ne 2
344 .na
345 \fB\fB-c\fR \fIname\fR\fR
346 .ad
347 .sp .6
348 .RS 4n
349 Records the configuration file \fIname\fR for use at runtime. Configuration
350 files can be employed to alter default search paths, provide a directory cache,
351 together with providing alternative object dependencies. See \fBcrle\fR(1).
352 .RE

354 .sp
355 .ne 2
356 .na
357 \fB\fB-C\fR\fR
358 .ad
359 .sp .6
360 .RS 4n
361 Demangles C++ symbol names displayed in diagnostic messages.
362 .RE

364 .sp
365 .ne 2
366 .na
367 \fB\fB-d\fR \fBy\fR | \fBn\fR\fR
368 .ad
369 .sp .6
370 .RS 4n
371 When \fB-d\fR \fBy\fR, the default, is specified, \fBld\fR uses dynamic
372 linking. When \fB-d\fR \fBn\fR is specified, \fBld\fR uses static linking. See
373 \fBStatic Executables\fR under DESCRIPTION, and \fB-B\fR
374 \fBdynamic\fR|\fBstatic\fR.
375 .RE

377 .sp
378 .ne 2
379 .na
380 \fB\fB-D\fR \fItoken\fR,...\fR
381 .ad
382 .sp .6
383 .RS 4n
384 Prints debugging information as specified by each \fItoken\fR, to the standard
385 error. The special token \fBhelp\fR indicates the full list of tokens
386 available. See \fIDebugging Aids\fR in \fILinker and Libraries Guide\fR.
387 .RE
```

```
.sp
.ne 2
.na
\fB\fB-e\fR \fIepsym\fR\fR
.ad
.br
.na
\fB\fB--entry\fR \fIepsym\fR\fR
.ad
.sp .6
.RS 4n
Sets the entry point address for the output file to be the symbol \fIepsym\fR.
.RE

.sp
.ne 2
.na
\fB\fB-f\fR \fIname\fR\fR
.ad
.br
.na
\fB\fB--auxiliary\fR \fIname\fR\fR
.ad
.sp .6
.RS 4n
Useful only when building a shared object. Specifies that the symbol table of
the shared object is used as an auxiliary filter on the symbol table of the
shared object specified by \fIname\fR. Multiple instances of this option are
allowed. This option can not be combined with the \fB-F\fR option. See
\fIGenerating Auxiliary Filters\fR in \fILinker and Libraries Guide\fR.
.RE

.sp
.ne 2
.na
\fB\fB-F\fR \fIname\fR\fR
.ad
.br
.na
\fB\fB--filter\fR \fIname\fR\fR
.ad
.sp .6
.RS 4n
Useful only when building a shared object. Specifies that the symbol table of
the shared object is used as a filter on the symbol table of the shared object
specified by \fIname\fR. Multiple instances of this option are allowed. This
option can not be combined with the \fB-f\fR option. See \fIGenerating Standard
Filters\fR in \fILinker and Libraries Guide\fR.
.RE

.sp
.ne 2
.na
\fB\fB-G\fR\fR
.ad
.br
.na
\fB\fB-shared\fR\fR
.ad
.sp .6
.RS 4n
In dynamic mode only, produces a shared object. Undefined symbols are allowed.
See Chapter 4, \fIShared Objects,\fR in \fILinker and Libraries Guide\fR.
.RE
```



```
.sp
.ne 2
.na
\fB\fB-h\fR \fIname\fR\fR
.ad
.br
.na
\fB\fB--soname\fR \fIname\fR\fR
.ad
.sp .6
.RS 4n
In dynamic mode only, when building a shared object, records \fIname\fR in the
object's dynamic section. \fIname\fR is recorded in any dynamic objects that
are linked with this object rather than the object's file system name.
Accordingly, \fIname\fR is used by the runtime linker as the name of the shared
object to search for at runtime. See \fIRecording a Shared Object Name\fR in
\fILinker and Libraries Guide\fR.
.RE

.sp
.ne 2
.na
\fB\fB-i\fR\fR
.ad
.sp .6
.RS 4n
Ignores \fBLD_LIBRARY_PATH\fR. This option is useful when an
\fBLD_LIBRARY_PATH\fR setting is in effect to influence the runtime library
search, which would interfere with the link-editing being performed.
.RE

.sp
.ne 2
.na
\fB\fB-I\fR \fIname\fR\fR
.ad
.br
.na
\fB\fB--dynamic-linker\fR \fIname\fR\fR
.ad
.sp .6
.RS 4n
When building an executable, uses \fIname\fR as the path name of the
interpreter to be written into the program header. The default in static mode
is no interpreter. In dynamic mode, the default is the name of the runtime
linker, \fBld.so.1\fR(1). Either case can be overridden by \fB-I\fR \fIname\fR.
\fBexec\fR(2) loads this interpreter when the \fBa.out\fR is loaded, and passes
control to the interpreter rather than to the \fBa.out\fR directly.
.RE

.sp
.ne 2
.na
\fB\fB-l\fR \fIx\fR\fR
.ad
.br
.na
\fB\fB--library\fR \fIx\fR\fR
.ad
.sp .6
.RS 4n
Searches a library \fBlib\fR\fIx\fR\fB\&.so\fR or \fBlib\fR\fIx\fR\fB\&.a\fR,
the conventional names for shared object and archive libraries, respectively.
In dynamic mode, unless the \fB-B\fR \fBstatic\fR option is in effect, \fBld\fR
searches each directory specified in the library search path for a
\fBlib\fR\fIx\fR\fB\&.so\fR or \fBlib\fR\fIx\fR\fB\&.a\fR file. The directory
```

search stops at the first directory containing either. \fBld\fR chooses the file ending in \fB\&.so\fR if \fB-l\fR\fIx\fR expands to two files with names of the form \fBlib\fR\fIx\fR\fB\&.so\fR and \fBlib\fR\fIx\fR\fB\&.a\fR. If no \fBlib\fR\fIx\fR\fB\&.so\fR is found, then \fBld\fR accepts \fBlib\fR\fIx\fR\fB\&.a\fR. In static mode, or when the \fB-B\fR \fBstatic\fR option is in effect, \fBld\fR selects only the file ending in \fB\&.a\fR. \fBld\fR searches a library when the library is encountered, so the placement of \fB-l\fR is significant. See \fILinking With Additional Libraries\fR in \fILinker and Libraries Guide\fR.
.RE

.sp
.ne 2
.na
\fB\fB-L\fR \fIpath\fR\fR
.ad
.br
.na
\fB\fB--library-path\fR \fIpath\fR\fR
.ad
.sp .6
.RS 4n
Adds \fIpath\fR to the library search directories. \fBld\fR searches for libraries first in any directories specified by the \fB-L\fR options and then in the standard directories. This option is useful only if the option precedes the \fB-l\fR options to which the \fB-L\fR option applies. See \fIDirectories Searched by the Link-Editor\fR in \fILinker and Libraries Guide\fR.
.sp
The environment variable \fBLD_LIBRARY_PATH\fR can be used to supplement the library search path, however the \fB-L\fR option is recommended, as the environment variable is also interpreted by the runtime environment. See \fBLD_LIBRARY_PATH\fR under ENVIRONMENT VARIABLES.
.RE

.sp
.ne 2
.na
\fB\fB-m\fR\fR
.ad
.sp .6
.RS 4n
Produces a memory map or listing of the input/output sections, together with any non-fatal multiply-defined symbols, on the standard output.
.RE

.sp
.ne 2
.na
\fB\fB-M\fR \fImapfile\fR\fR
.ad
.sp .6
.RS 4n
Reads \fImapfile\fR as a text file of directives to \fBld\fR. This option can be specified multiple times. If \fImapfile\fR is a directory, then all regular files, as defined by \fBstat\fR(2), within the directory are processed. See Chapter 9, \fIMapfile Option,\fR in \fILinker and Libraries Guide\fR. Example mapfiles are provided in \fB/usr/lib/ld\fR. See FILES.
.RE

.sp
.ne 2
.na
\fB\fB-N\fR \fIstring\fR\fR
.ad
.sp .6
.RS 4n

This option causes a \fBDT_NEEDED\fR entry to be added to the \fB\&.dynamic\fR section of the object being built. The value of the \fBDT_NEEDED\fR string is the \fIstring\fR that is specified on the command line. This option is position dependent, and the \fBDT_NEEDED\fR \fB\&.dynamic\fR entry is relative to the other dynamic dependencies discovered on the link-edit line. This option is useful for specifying dependencies within device driver relocatable objects when combined with the \fB-dy\fR and \fB-r\fR options.
.RE

.sp
.ne 2
.na
\fB\fB-o\fR \fIoutfile\fR\fR
.ad
.br
.na
\fB\fB--output\fR \fIoutfile\fR\fR
.ad
.sp .6
.RS 4n
Produces an output object file that is named \fIoutfile\fR. The name of the default object file is \fBa.out\fR.
.RE

.sp
.ne 2
.na
\fB\fB-p\fR \fIauditlib\fR\fR
.ad
.sp .6
.RS 4n
Identifies an audit library, \fIauditlib\fR. This audit library is used to audit the object being created at runtime. A shared object identified as requiring auditing with the \fB-p\fR option, has this requirement inherited by any object that specifies the shared object as a dependency. See the \fB-P\fR option. See \fIRuntime Linker Auditing Interface\fR in \fILinker and Libraries Guide\fR.
.RE

.sp
.ne 2
.na
\fB\fB-P\fR \fIauditlib\fR\fR
.ad
.sp .6
.RS 4n
Identifies an audit library, \fIauditlib\fR. This audit library is used to audit the dependencies of the object being created at runtime. Dependency auditing can also be inherited from dependencies that are identified as requiring auditing. See the \fB-p\fR option, and the \fB-z\fR \fBglobalaudit\fR option. See \fIRuntime Linker Auditing Interface\fR in \fILinker and Libraries Guide\fR.
.RE

.sp
.ne 2
.na
\fB\fB-Q\fR \fBy\fR | \fBn\fR\fR
.ad
.sp .6
.RS 4n
Under \fB-Q\fR \fBy\fR, an \fBident\fR string is added to the \fB\&.comment\fR section of the output file. This string identifies the version of the \fBld\fR used to create the file. This results in multiple \fBld\fR \fBidents\fR when there have been multiple linking steps, such as when using \fBld\fR \fB-r\fR. This identification is identical with the default action of the \fBcc\fR

```
652 command. \fB-Q\fR \fBn\fR suppresses version identification. \fB\&.comment\fR
653 sections can be manipulated by the \fBmcs\fR(1) utility.
654 .RE

656 .sp
657 .ne 2
658 .na
659 \fB\fB-r\fR\fR
660 .ad
661 .br
662 .na
663 \fB\fB--relocatable\fR\fR
664 .ad
665 .sp .6
666 .RS 4n
667 Combines relocatable object files to produce one relocatable object file.
668 \fBld\fR does not complain about unresolved references. This option cannot be
669 used with the \fB-a\fR option.
670 .RE

672 .sp
673 .ne 2
674 .na
675 \fB\fB-R\fR \fIpath\fR\fR
676 .ad
677 .br
678 .na
679 \fB\fB-rpath\fR \fIpath\fR\fR
680 .ad
681 .sp .6
682 .RS 4n
683 A colon-separated list of directories used to specify library search
684 directories to the runtime linker. If present and not NULL, the path is
685 recorded in the output object file and passed to the runtime linker. Multiple
686 instances of this option are concatenated together with each \fIpath\fR
687 separated by a colon. See \fIDirectories Searched by the Runtime Linker\fR in
688 \fILinker and Libraries Guide\fR.
689 .sp
690 The use of a runpath within an associated object is preferable to setting
691 global search paths such as through the \fBLD_LIBRARY_PATH\fR environment
692 variable. Only the runpaths that are necessary to find the objects dependencies
693 should be recorded. \fBldd\fR(1) can also be used to discover unused runpaths
694 in dynamic objects, when used with the \fB-U\fR option.
695 .sp
696 Various tokens can also be supplied with a runpath that provide a flexible
697 means of identifying system capabilities or an objects location. See Appendix
698 C, \fIEstablishing Dependencies with Dynamic String Tokens,\fR in \fILinker and
699 Libraries Guide\fR. The \fB$ORIGIN\fR token is especially useful in allowing
700 dynamic objects to be relocated to different locations in the file system.
701 .RE

703 .sp
704 .ne 2
705 .na
706 \fB\fB-s\fR\fR
707 .ad
708 .br
709 .na
710 \fB\fB--strip-all\fR\fR
711 .ad
712 .sp .6
713 .RS 4n
714 Strips symbolic information from the output file. Any debugging information,
715 that is, \fB\&.line\fR, \fB\&.debug*\fR, and \fB\&.stab*\fR sections, and their
716 associated relocation entries are removed. Except for relocatable files, a
717 symbol table \fBSHT_SYMTAB\fR and its associated string table section are not
```



```
718 created in the output object file. The elimination of a \fBSHT_SYMTAB\fR symbol
719 table can reduce the \fB\&.stab*\fR debugging information that is generated
720 using the compiler drivers \fB-g\fR option. See the \fB-z\fR \fBredlocsym\fR
721 and \fB-z\fR \fBnoldynsym\fR options.
722 .RE

724 .sp
725 .ne 2
726 .na
727 \fB\fB-S\fR \fIsupportlib\fR\fR
728 .ad
729 .sp .6
730 .RS 4n
731 The shared object \fIsupportlib\fR is loaded with \fBld\fR and given
732 information regarding the linking process. Shared objects that are defined by
733 using the \fB-S\fR option can also be supplied using the \fBSGS_SUPPORT\fR
734 environment variable. See \fILink-Editor Support Interface\fR in \fILinker and
735 Libraries Guide\fR.
736 .RE

738 .sp
739 .ne 2
740 .na
741 \fB\fB-t\fR\fR
742 .ad
743 .sp .6
744 .RS 4n
745 Turns off the warning for multiply-defined symbols that have different sizes or
746 different alignments.
747 .RE

749 .sp
750 .ne 2
751 .na
752 \fB\fB-u\fR \fIsymname\fR\fR
753 .ad
754 .br
755 .na
756 \fB\fB--undefined\fR \fIsymname\fR\fR
757 .ad
758 .sp .6
759 .RS 4n
760 Enters \fIsymname\fR as an undefined symbol in the symbol table. This option is
761 useful for loading entirely from an archive library. In this instance, an
762 unresolved reference is needed to force the loading of the first routine. The
763 placement of this option on the command line is significant. This option must
764 be placed before the library that defines the symbol. See \fIDefining
765 Additional Symbols with the u option\fR in \fILinker and Libraries Guide\fR.
766 .RE

768 .sp
769 .ne 2
770 .na
771 \fB\fB-V\fR\fR
772 .ad
773 .br
774 .na
775 \fB\fB--version\fR\fR
776 .ad
777 .sp .6
778 .RS 4n
779 Outputs a message giving information about the version of \fBld\fR being used.
780 .RE

782 .sp
783 .ne 2
```

```
.na
\fB\fB-Y\fR \fBP,\fR\fIdirlist\fR\fR
.ad
.sp .6
.RS 4n
Changes the default directories used for finding libraries. \fIdirlist\fR is a
colon-separated path list.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBabsexec\fR\fR
.ad
.sp .6
.RS 4n
Useful only when building a dynamic executable. Specifies that references to
external absolute symbols should be resolved immediately instead of being left
for resolution at runtime. In very specialized circumstances, this option
removes text relocations that can result in excessive swap space demands by an
executable.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBallextract\fR | \fBdefaultextract\fR | \fBweakextract\fR\fR
.ad
.br
.na
\fB\fB--whole-archive\fR | \fB--no-whole-archive\fR\fR
.ad
.sp .6
.RS 4n
Alters the extraction criteria of objects from any archives that follow. By
default, archive members are extracted to satisfy undefined references and to
promote tentative definitions with data definitions. Weak symbol references do
not trigger extraction. Under the \fB-z\fR \fBallextract\fR or
\fB--whole-archive\fR options, all archive members are extracted from the
archive. Under \fB-z\fR \fBweakextract\fR, weak references trigger archive
extraction. The \fB-z\fR \fBdefaultextract\fR or \fB--no-whole-archive\fR
options provide a means of returning to the default following use of the former
extract options. See \fIArchive Processing\fR in \fILinker and Libraries
Guide\fR.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBaltexec64\fR\fR
.ad
.sp .6
.RS 4n
Execute the 64-bit \fBld\fR. The creation of very large 32-bit objects can
exhaust the virtual memory that is available to the 32-bit \fBld\fR. The
\fB-z\fR \fBaltexec64\fR option can be used to force the use of the associated
64-bit \fBld\fR. The 64-bit \fBld\fR provides a larger virtual address space
for building 32-bit objects. See \fIThe 32-bit link-editor and 64-bit
link-editor\fR in \fILinker and Libraries Guide\fR.
.RE

.sp
.ne 2
.na
\fB-z\fR \fBaslr[=\fIstate\fR]\fR
.ad
```



```
.sp .6
.RS 4n
Specify whether the executable's address space should be randomized on
execution.  If \fIstate\fR is "enabled" randomization will always occur when
this executable is run (regardless of inherited settings).  If \fIstate\fR is
"disabled" randomization will never occur when this executable is run.  If
\fIstate\fR is omitted, ASLR is enabled.

An executable that should simple use the settings inherited from its
environment should not use this flag at all.
.RE

.sp
.ne 2
.na
#endif /* ! codereview */
\fB\fB-z\fR \fBcombreloc\fR | \fBnocombreloc\fR\fR
.ad
.sp .6
.RS 4n
By default, \fBld\fR combines multiple relocation sections when building
executables or shared objects. This section combination differs from
relocatable objects, in which relocation sections are maintained in a
one-to-one relationship with the sections to which the relocations must be
applied. The \fB-z\fR \fBnocombreloc\fR option disables this merging of
relocation sections, and preserves the one-to-one relationship found in the
original relocatable objects.
.sp
\fBld\fR sorts the entries of data relocation sections by their symbol
reference. This sorting reduces runtime symbol lookup. When multiple relocation
sections are combined, this sorting produces the least possible relocation
overhead when objects are loaded into memory, and speeds the runtime loading of
dynamic objects.
.sp
Historically, the individual relocation sections were carried over to any
executable or shared object, and the \fB-z\fR \fBcombreloc\fR option was
required to enable the relocation section merging previously described.
Relocation section merging is now the default. The \fB-z\fR \fBcombreloc\fR
option is still accepted for the benefit of old build environments, but the
option is unnecessary, and has no effect.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBassert-deflib\fR\fR
.ad
.br
.na
\fB\fB-z\fR \fBassert-deflib=\fR\fIlibname\fR\fR
.ad
.sp .6
.RS 4n
Enables warnings that check the location of where libraries passed in with
\fB-l\fR are found. If the link-editor finds a library on its default search
path it will emit a warning. This warning can be made fatal in conjunction with
the option \fB-z fatal-warnings\fR. Passing \fIlibname\fR white lists a library
from this check. The library must be the full name of the library, e.g.
\fIlibc.so\fR. To white list multiple libraries, the \fB-z
assert-deflib=\fR\fIlibname\fR option can be repeated multiple times. This
option is useful when trying to build self-contained objects where a referenced
library might exist in the default system library path and in alternate paths
specified by \fB-L\fR, but you only want the alternate paths to be used.
.RE

.sp
```

```
.ne 2
.na
\fB\fB-z\fR \fBdefs\fR | \fBnodefs\fR\fR
.ad
.br
.na
\fB\fB--no-undefined\fR\fR
.ad
.sp .6
.RS 4n
The \fB-z\fR \fBdefs\fR option and the \fB--no-undefined\fR option force a
fatal error if any undefined symbols remain at the end of the link. This mode
is the default when an executable is built. For historic reasons, this mode is
\fBnot\fR the default when building a shared object. Use of the \fB-z\fR
\fBdefs\fR option is recommended, as this mode assures the object being built
is self-contained. A self-contained object has all symbolic references resolved
internally, or to the object's immediate dependencies.
.sp
The \fB-z\fR \fBnodefs\fR option allows undefined symbols. For historic
reasons, this mode is the default when a shared object is built. When used with
executables, the behavior of references to such undefined symbols is
unspecified. Use of the \fB-z\fR \fBnodefs\fR option is not recommended.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBdirect\fR | \fBnodirect\fR\fR
.ad
.sp .6
.RS 4n
Enables or disables direct binding to any dependencies that follow on the
command line. These options allow finer control over direct binding than the
global counterpart \fB-B\fR \fBdirect\fR. The \fB-z\fR \fBdirect\fR option also
differs from the \fB-B\fR \fBdirect\fR option in the following areas. Direct
binding information is not established between a symbol reference and an
associated definition within the object being created. Lazy loading is not
enabled.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBendfiltee\fR\fR
.ad
.sp .6
.RS 4n
Marks a filtee so that when processed by a filter, the filtee terminates any
further filtee searches by the filter. See \fIReducing Filtee Searches\fR in
\fILinker and Libraries Guide\fR.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBfatal-warnings\fR | \fBnofatal-warnings\fR\fR
.ad
.br
.na
\fB\fB--fatal-warnings\fR | \fB--no-fatal-warnings\fR
.ad
.sp .6
.RS 4n
Controls the behavior of warnings emitted from the link-editor. Setting \fB-z
fatal-warnings\fR promotes warnings emitted by the link-editor to fatal errors
that will cause the link-editor to fail before linking. \fB-z
```



```
nofatal-warnings\fR instead demotes these warnings such that they will not cause
the link-editor to exit prematurely.
.RE


.sp
.ne 2
.na
\fB\fB-z\fR \fBfiniarray=\fR\fIfunction\fR\fR
.ad
.sp .6
.RS 4n
Appends an entry to the \fB\&.finiarray\fR section of the object being built.
If no \fB\&.finiarray\fR section is present, a section is created. The new
entry is initialized to point to \fIfunction\fR. See \fIInitialization and
Termination Sections\fR in \fILinker and Libraries Guide\fR.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBglobalaudit\fR\fR
.ad
.sp .6
.RS 4n
This option supplements an audit library definition that has been recorded with
the \fB-P\fR option. This option is only meaningful when building a dynamic
executable. Audit libraries that are defined within an object with the \fB-P\fR
option typically allow for the auditing of the immediate dependencies of the
object. The \fB-z\fR \fBglobalaudit\fR promotes the auditor to a global
auditor, thus allowing the auditing of all dependencies. See \fIInvoking the
Auditing Interface\fR in \fILinker and Libraries Guide\fR.
.sp
An auditor established with the \fB-P\fR option and the \fB-z\fR
\fBglobalaudit\fR option, is equivalent to the auditor being established with
the \fBLD_AUDIT\fR environment variable. See \fBld.so.1\fR(1).
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBgroupperm\fR | \fBnogroupperm\fR\fR
.ad
.sp .6
.RS 4n
Assigns, or deassigns each dependency that follows to a unique group. The
assignment of a dependency to a group has the same effect as if the dependency
had been built using the \fB-B\fR \fBgroup\fR option.
.RE

.sp
.ne 2
.na
\fB-z\fR \fBguidance\fR[=\fIid1\fR,\fIid2\fR...]
.ad
.sp .6
.RS 4n
Give messages suggesting link-editor features that could improve the resulting
dynamic object.
.LP
Specific classes of suggestion can be silenced by specifying an optional comma s
list of guidance identifiers.
.LP
The current classes of suggestion provided are:

.sp
```

```
.ne 2
.na
Enable use of direct binding
.ad
.sp .6
.RS 4n
Suggests that \fB-z direct\fR or \fB-B direct\fR be present prior to any
specified dependency.  This allows predictable symbol binding at runtime.

Can be disabled with \fB-z guidance=nodirect\fR
.RE

.sp
.ne 2
.na
Enable lazy dependency loading
.ad
.sp .6
.RS 4n
Suggests that \fB-z lazyload\fR be present prior to any specified dependency.
This allows the dynamic object to be loaded more quickly.

Can be disabled with \fB-z guidance=nolazyload\fR.
.RE

.sp
.ne 2
.na
Shared objects should define all their dependencies.
.ad
.sp .6
.RS 4n
Suggests that \fB-z defs\fR be specified on the link-editor command line.
Shared objects that explicitly state all their dependencies behave more
predictably when used.

Can be be disabled with \fB-z guidance=nodefs\fR
.RE

.sp
.ne 2
.na
Version 2 mapfile syntax
.ad
.sp .6
.RS 4n
Suggests that any specified mapfiles use the more readable version 2 syntax.

Can be disabled with \fB-z guidance=nomapfile\fR.
.RE

.sp
.ne 2
.na
Read-only text segment
.ad
.sp .6
.RS 4n
Should any runtime relocations within the text segment exist, suggests that
the object be compiled with position independent code (PIC).  Keeping large
allocatable sections read-only allows them to be shared between processes
using a given shared object.

Can be disabled with \fB-z guidance=notext\fR
.RE
```



```
.sp
.ne 2
.na
No unused dependencies
.ad
.sp .6
.RS 4n
Suggests that any dependency not referenced by the resulting dynamic object be
removed from the link-editor command line.

Can be disabled with \fB-z guidance=nounused\fR.
.RE
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBhelp\fR\fR
.ad
.br
.na
\fB\fB--help\fR\fR
.ad
.sp .6
.RS 4n
Print a summary of the command line options on the standard output and exit.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBignore\fR | \fBrecord\fR\fR
.ad
.sp .6
.RS 4n
Ignores, or records, dynamic dependencies that are not referenced as part of
the link-edit. Ignores, or records, unreferenced \fBELF\fR sections from the
relocatable objects that are read as part of the link-edit. By default,
\fB-z\fR \fBrecord\fR is in effect.
.sp
If an \fBELF\fR section is ignored, the section is eliminated from the output
file being generated. A section is ignored when three conditions are true. The
eliminated section must contribute to an allocatable segment. The eliminated
section must provide no global symbols. No other section from any object that
contributes to the link-edit, must reference an eliminated section.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBinitarray=\fR\fIfunction\fR\fR
.ad
.sp .6
.RS 4n
Appends an entry to the \fB\&.initarray\fR section of the object being built.
If no \fB\&.initarray\fR section is present, a section is created. The new
entry is initialized to point to \fIfunction\fR. See \fIInitialization and
Termination Sections\fR in \fILinker and Libraries Guide\fR.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBinitfirst\fR\fR
.ad
.sp .6
```

```
.RS 4n
Marks the object so that its runtime initialization occurs before the runtime
initialization of any other objects brought into the process at the same time.
In addition, the object runtime finalization occurs after the runtime
finalization of any other objects removed from the process at the same time.
This option is only meaningful when building a shared object.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBinterpose\fR\fR
.ad
.sp .6
.RS 4n
Marks the object as an interposer. At runtime, an object is identified as an
explicit interposer if the object has been tagged using the \fB-z interpose\fR
option. An explicit interposer is also established when an object is loaded
using the \fBLD_PRELOAD\fR environment variable. Implicit interposition can
occur because of the load order of objects, however, this implicit
interposition is unknown to the runtime linker. Explicit interposition can
ensure that interposition takes place regardless of the order in which objects
are loaded. Explicit interposition also ensures that the runtime linker
searches for symbols in any explicit interposers when direct bindings are in
effect.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBlazyload\fR | \fBnolazyload\fR\fR
.ad
.sp .6
.RS 4n
Enables or disables the marking of dynamic dependencies to be lazily loaded.
Dynamic dependencies which are marked \fBlazyload\fR are not loaded at initial
process start-up. These dependencies are delayed until the first binding to the
object is made. \fBNote:\fR Lazy loading requires the correct declaration of
dependencies, together with associated runpaths for each dynamic object used
within a process. See \fILazy Loading of Dynamic Dependencies\fR in \fILinker
and Libraries Guide\fR.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBld32\fR=\fIarg1\fR,\fIarg2\fR,...\fR
.ad
.br
.na
\fB\fB-z\fR \fBld64\fR=\fIarg1\fR,\fIarg2\fR,...\fR
.ad
.sp .6
.RS 4n
The class of the link-editor is affected by the class of the output file being
created and by the capabilities of the underlying operating system. The
\fB-z\fR \fBld\fR[\fB32\fR|\fB64\fR] options provide a means of defining any
link-editor argument. The defined argument is only interpreted, respectively,
by the 32-bit class or 64-bit class of the link-editor.
.sp
For example, support libraries are class specific, so the correct class of
support library can be ensured using:
.sp
.in +2
.nf
\fBld ... -z ld32=-Saudit32.so.1 -z ld64=-Saudit64.so.1 ...\fR
```



```
.fi
.in -2
.sp

The class of link-editor that is invoked is determined from the \fBELF\fR class
of the first relocatable file that is seen on the command line. This
determination is carried out \fBprior\fR to any \fB-z\fR
\fBld\fR[\fB32\fR|\fB64\fR] processing.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBloadfltr\fR\fR
.ad
.sp .6
.RS 4n
Marks a filter to indicate that filtees must be processed immediately at
runtime. Normally, filter processing is delayed until a symbol reference is
bound to the filter. The runtime processing of an object that contains this
flag mimics that which occurs if the \fBLD_LOADFLTR\fR environment variable is
in effect. See the \fBld.so.1\fR(1).
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBmuldefs\fR\fR
.ad
.br
.na
\fB\fB--allow-multiple-definition\fR\fR
.ad
.sp .6
.RS 4n
Allows multiple symbol definitions. By default, multiple symbol definitions
that occur between relocatable objects result in a fatal error condition. This
option, suppresses the error condition, allowing the first symbol definition to
be taken.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBnocompstrtab\fR\fR
.ad
.sp .6
.RS 4n
Disables the compression of \fBELF\fR string tables. By default, string
compression is applied to \fBSHT_STRTAB\fR sections, and to \fBSHT_PROGBITS\fR
sections that have their \fBSHF_MERGE\fR and \fBSHF_STRINGS\fR section flags
set.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBnodefaultlib\fR\fR
.ad
.sp .6
.RS 4n
Marks the object so that the runtime default library search path, used after
any \fBLD_LIBRARY_PATH\fR or runpaths, is ignored. This option implies that all
dependencies of the object can be satisfied from its runpath.
.RE
```

```
.sp
.ne 2
.na
\fB\fB-z\fR \fBnodelete\fR\fR
.ad
.sp .6
.RS 4n
Marks the object as non-deletable at runtime. This mode is similar to adding
the object to the process by using \fBdlopen\fR(3C) with the
\fBRTLD_NODELETE\fR mode.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBnodlopen\fR\fR
.ad
.sp .6
.RS 4n
Marks the object as not available to \fBdlopen\fR(3C), either as the object
specified by the \fBdlopen()\fR, or as any form of dependency required by the
object specified by the \fBdlopen()\fR. This option is only meaningful when
building a shared object.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBnodump\fR\fR
.ad
.sp .6
.RS 4n
Marks the object as not available to \fBdldump\fR(3C).
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBnoldynsym\fR\fR
.ad
.sp .6
.RS 4n
Prevents the inclusion of a \fB\&.SUNW_ldynsym\fR section in dynamic
executables or sharable libraries. The \fB\&.SUNW_ldynsym\fR section augments
the \fB\&.dynsym\fR section by providing symbols for local functions. Local
function symbols allow debuggers to display local function names in stack
traces from stripped programs. Similarly, \fBdladdr\fR(3C) is able to supply
more accurate results.
.sp
The \fB-z\fR \fBnoldynsym\fR option also prevents the inclusion of the two
symbol sort sections that are related to the \fB\&.SUNW_ldynsym\fR section. The
\fB\&.SUNW_dynsymsort\fR section provides sorted access to regular function and
variable symbols. The \fB\&.SUNW_dyntlssort\fR section provides sorted access
to thread local storage (\fBTLS\fR) variable symbols.
.sp
The \fB\&.SUNW_ldynsym\fR, \fB\&.SUNW_dynsymsort\fR, and
\fB\&.SUNW_dyntlssort\fR sections, which becomes part of the allocable text
segment of the resulting file, cannot be removed by \fBstrip\fR(1). Therefore,
the \fB-z\fR \fBnoldynsym\fR option is the only way to prevent their inclusion.
See the \fB-s\fR and \fB-z\fR \fBredlocsym\fR options.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBnopartial\fR\fR
```



```
.ad
.sp .6
.RS 4n
Partially initialized symbols, that are defined within relocatable object
files, are expanded in the output file being generated.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBnoversion\fR\fR
.ad
.sp .6
.RS 4n
Does not record any versioning sections. Any version sections or associated
\fB\&.dynamic\fR section entries are not generated in the output image.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBnow\fR\fR
.ad
.sp .6
.RS 4n
Marks the object as requiring non-lazy runtime binding. This mode is similar to
adding the object to the process by using \fBdlopen\fR(3C) with the
\fBRTLD_NOW\fR mode. This mode is also similar to having the \fBLD_BIND_NOW\fR
environment variable in effect. See \fBld.so.1\fR(1).
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBorigin\fR\fR
.ad
.sp .6
.RS 4n
Marks the object as requiring immediate \fB$ORIGIN\fR processing at runtime.
This option is only maintained for historic compatibility, as the runtime
analysis of objects to provide for \fB$ORIGIN\fR processing is now default.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBpreinitarray=\fR\fIfunction\fR\fR
.ad
.sp .6
.RS 4n
Appends an entry to the \fB\&.preinitarray\fR section of the object being
built. If no \fB\&.preinitarray\fR section is present, a section is created.
The new entry is initialized to point to \fIfunction\fR. See \fIInitialization
and Termination Sections\fR in \fILinker and Libraries Guide\fR.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBredlocsym\fR\fR
.ad
.sp .6
.RS 4n
Eliminates all local symbols except for the \fISECT\fR symbols from the symbol
table \fBSHT_SYMTAB\fR. All relocations that refer to local symbols are updated
to refer to the corresponding \fISECT\fR symbol. This option allows specialized
```

```
objects to greatly reduce their symbol table sizes. Eliminated local symbols
can reduce the \fB\&.stab*\fR debugging information that is generated using the
compiler drivers \fB-g\fR option. See the \fB-s\fR and \fB-z\fR \fBnoldynsym\fR
options.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBrelaxreloc\fR\fR
.ad
.sp .6
.RS 4n
\fBld\fR normally issues a fatal error upon encountering a relocation using a
symbol that references an eliminated COMDAT section. If \fB-z\fR
\fBrelaxreloc\fR is enabled, \fBld\fR instead redirects such relocations to the
equivalent symbol in the COMDAT section that was kept. \fB-z\fR
\fBrelaxreloc\fR is a specialized option, mainly of interest to compiler
authors, and is not intended for general use.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBrescan-now\fR\fR
.ad
.br
.na
\fB\fB-z\fR \fBrescan\fR\fR
.ad
.sp .6
.RS 4n
These options rescan the archive files that are provided to the link-edit. By
default, archives are processed once as the archives appear on the command
line. Archives are traditionally specified at the end of the command line so
that their symbol definitions resolve any preceding references. However,
specifying archives multiple times to satisfy their own interdependencies can
be necessary.
.sp
\fB-z\fR \fBrescan-now\fR is a positional option, and is processed by the
link-editor immediately when encountered on the command line. All archives seen
on the command line up to that point are immediately reprocessed in an attempt
to locate additional archive members that resolve symbol references. This
archive rescanning is repeated until a pass over the archives occurs in which
no new members are extracted.
.sp
\fB-z\fR \fBrescan\fR is a position independent option. The link-editor defers
the rescan operation until after it has processed the entire command line, and
then initiates a final rescan operation over all archives seen on the command
line. The \fB-z\fR \fBrescan\fR operation can interact          incorrectly
with objects that contain initialization (.init) or finalization (.fini)
sections, preventing the code in those sections from running. For this reason,
\fB-z\fR \fBrescan\fR is deprecated, and use of \fB-z\fR \fBrescan-now\fR is
advised.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBrescan-start\fR ... \fB-z\fR \fBrescan-end\fR\fR
.ad
.br
.na
\fB\fB--start-group\fR ... \fB--end-group\fR\fR
.ad
.br
```



```
.na
\fB\fB-(\fR ... \fB-)\fR\fR
.ad
.sp .6
.RS 4n
Defines an archive rescan group. This is a positional construct, and is
processed by the link-editor immediately upon encountering the closing
delimiter option.  Archives found within the group delimiter options are
reprocessed as a group in an attempt to locate  additional archive members that
resolve symbol references. This archive rescanning  is repeated  until a  pass
over the archives On the occurs in  which no  new  members are extracted.
Archive rescan groups cannot be nested.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBtarget=sparc|x86\fR \fI\fR\fR
.ad
.sp .6
.RS 4n
Specifies the machine type for the output object. Supported targets are Sparc
and x86. The 32-bit machine type for the specified target is used unless the
\fB-64\fR option is also present, in which case the corresponding 64-bit
machine type is used. By default, the machine type of the object being
generated is determined from the first \fBELF\fR object processed from the
command line. If no objects are specified, the machine type is determined by
the first object encountered within the first archive processed from the
command line. If there are no objects or archives, the link-editor assumes the
native machine. This option is useful when creating an object directly with
\fBld\fR whose input is solely from a \fBmapfile\fR. See the \fB-M\fR option.
It can also be useful in the rare case of linking entirely from an archive that
contains objects of different machine types for which the first object is not
of the desired machine type. See \fIThe 32-bit link-editor and 64-bit
link-editor\fR in \fILinker and Libraries Guide\fR.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBtext\fR\fR
.ad
.sp .6
.RS 4n
In dynamic mode only, forces a fatal error if any relocations against
non-writable, allocatable sections remain. For historic reasons, this mode is
not the default when building an executable or shared object. However, its use
is recommended to ensure that the text segment of the dynamic object being
built is shareable between multiple running processes. A shared text segment
incurs the least relocation overhead when loaded into memory. See
\fIPosition-Independent Code\fR in \fILinker and Libraries Guide\fR.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBtextoff\fR\fR
.ad
.sp .6
.RS 4n
In dynamic mode only, allows relocations against all allocatable sections,
including non-writable ones. This mode is the default when building a shared
object.
.RE

.sp
```

```
.ne 2
.na
\fB\fB-z\fR \fBtextwarn\fR\fR
.ad
.sp .6
.RS 4n
In dynamic mode only, lists a warning if any relocations against non-writable,
allocatable sections remain. This mode is the default when building an
executable.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fBverbose\fR\fR
.ad
.sp .6
.RS 4n
This option provides additional warning diagnostics during a link-edit.
Presently, this option conveys suspicious use of displacement relocations. This
option also conveys the restricted use of static \fBTLS\fR relocations when
building shared objects. In future, this option might be enhanced to provide
additional diagnostics that are deemed too noisy to be generated by default.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR\fBwrap=\fR\fIsymbol\fR\fR
.ad
.br
.na
\fB\fB-wrap=\fR \fIsymbol\fR\fR
.ad
.br
.na
\fB\fB--wrap=\fR \fIsymbol\fR\fR
.ad
.sp .6
.RS 4n
Rename undefined references to \fIsymbol\fR in order to allow wrapper code to
be linked into the output object without having to modify source code. When
\fB-z wrap\fR is specified, all undefined references to \fIsymbol\fR are
modified to reference \fB__wrap_\fR\fIsymbol\fR, and all references to
\fB__real_\fR\fIsymbol\fR are modified to reference \fIsymbol\fR. The user is
expected to provide an object containing the \fB__wrap_\fR\fIsymbol\fR
function. This wrapper function can call \fB__real_\fR\fIsymbol\fR in order to
reference the actual function being wrapped.
.sp
The following is an example of a wrapper for the \fBmalloc\fR(3C) function:
.sp
.in +2
.nf
void *
__wrap_malloc(size_t c)
{
        (void) printf("malloc called with %zu\en", c);
        return (__real_malloc(c));
}
.fi
.in -2

If you link other code with this file using \fB-z\fR \fBwrap=malloc\fR to
compile all the objects, then all calls to \fBmalloc\fR will call the function
\fB__wrap_malloc\fR instead. The call to \fB__real_malloc\fR will call the real
\fBmalloc\fR function.
```



```
.sp
The real and wrapped functions should be maintained in separate source files.
Otherwise, the compiler or assembler may resolve the call instead of leaving
that operation for the link-editor to carry out, and prevent the wrap from
occurring.
.RE

.SH ENVIRONMENT VARIABLES
.sp
.ne 2
.na
\fB\fBLD_ALTEXEC\fR\fR
.ad
.sp .6
.RS 4n
An alternative link-editor path name. \fBld\fR executes, and passes control to
this alternative link-editor. This environment variable provides a generic
means of overriding the default link-editor that is called from the various
compiler drivers. See the \fB-z altexec64\fR option.
.RE

.sp
.ne 2
.na
\fB\fBLD_LIBRARY_PATH\fR\fR
.ad
.sp .6
.RS 4n
A list of directories in which to search for the libraries specified using the
\fB-l\fR option. Multiple directories are separated by a colon. In the most
general case, this environment variable contains two directory lists separated
by a semicolon:
.sp
.in +2
.nf
\fIdirlist1\fR\fB;\fR\fIdirlist2\fR
.fi
.in -2
.sp

If \fBld\fR is called with any number of occurrences of \fB-L\fR, as in:
.sp
.in +2
.nf
\fBld ... -L\fIpath1\fR ... -L\fIpathn\fR ...\fR
.fi
.in -2
.sp

then the search path ordering is:
.sp
.in +2
.nf
\fB\fIdirlist1 path1\fR ... \fIpathn dirlist2\fR LIBPATH\fR
.fi
.in -2
.sp

When the list of directories does not contain a semicolon, the list is
interpreted as \fIdirlist2\fR.
.sp
The \fBLD_LIBRARY_PATH\fR environment variable also affects the runtime linkers
search for dynamic dependencies.
.sp
This environment variable can be specified with a _32 or _64 suffix. This makes
the environment variable specific, respectively, to 32-bit or 64-bit processes
```

```
and overrides any non-suffixed version of the environment variable that is in
effect.
.RE

.sp
.ne 2
.na
\fB\fBLD_NOEXEC_64\fR\fR
.ad
.sp .6
.RS 4n
Suppresses the automatic execution of the 64-bit link-editor. By default, the
link-editor executes the 64-bit version when the \fBELF\fR class of the first
relocatable file identifies a 64-bit object. The 64-bit image that a 32-bit
link-editor can create, has some limitations. However, some link-edits might
find the use of the 32-bit link-editor faster.
.RE

.sp
.ne 2
.na
\fB\fBLD_OPTIONS\fR\fR
.ad
.sp .6
.RS 4n
A default set of options to \fBld\fR. \fBLD_OPTIONS\fR is interpreted by
\fBld\fR just as though its value had been placed on the command line,
immediately following the name used to invoke \fBld\fR, as in:
.sp
.in +2
.nf
\fBld $LD_OPTIONS ... \fIother-arguments\fR ...\fR
.fi
.in -2
.sp

.RE

.sp
.ne 2
.na
\fB\fBLD_RUN_PATH\fR\fR
.ad
.sp .6
.RS 4n
An alternative mechanism for specifying a runpath to the link-editor. See the
\fB-R\fR option. If both \fBLD_RUN_PATH\fR and the \fB-R\fR option are
specified, \fB-R\fR supersedes.
.RE

.sp
.ne 2
.na
\fB\fBSGS_SUPPORT\fR\fR
.ad
.sp .6
.RS 4n
Provides a colon-separated list of shared objects that are loaded with the
link-editor and given information regarding the linking process. This
environment variable can be specified with a _32 or _64 suffix. This makes the
environment variable specific, respectively, to the 32-bit or 64-bit class of
\fBld\fR and overrides any non-suffixed version of the environment variable
that is in effect. See the \fB-S\fR option.
.RE

.sp
```



```
.LP
Notice that environment variable-names that begin with the
characters '\fBLD_\fR' are reserved for possible future enhancements to \fBld\fR
\fBld.so.1\fR(1).
.SH FILES
.sp
.ne 2
.na
\fB\fBlib\fIx\fR.so\fR\fR
.ad
.RS 15n
shared object libraries.
.RE

.sp
.ne 2
.na
\fB\fBlib\fIx\fR.a\fR\fR
.ad
.RS 15n
archive libraries.
.RE

.sp
.ne 2
.na
\fB\fBa.out\fR\fR
.ad
.RS 15n
default output file.
.RE

.sp
.ne 2
.na
\fB\fILIBPATH\fR\fR
.ad
.RS 15n
For 32-bit libraries, the default search path is \fB/usr/ccs/lib\fR, followed
by \fB/lib\fR, and finally \fB/usr/lib\fR. For 64-bit libraries, the default
search path is \fB/lib/64\fR, followed by \fB/usr/lib/64\fR.
.RE

.sp
.ne 2
.na
\fB\fB/usr/lib/ld\fR\fR
.ad
.RS 15n
A directory containing several \fBmapfiles\fR that can be used during
link-editing. These \fBmapfiles\fR provide various capabilities, such as
defining memory layouts, aligning bss, and defining non-executable stacks.
.RE

.SH ATTRIBUTES
.sp
.LP
See \fBattributes\fR(5) for descriptions of the following attributes:
.sp

.sp
.TS
box;
c | c
l | l .
ATTRIBUTE TYPE  ATTRIBUTE VALUE
```

```
1840 _
1841 Interface Stability     Committed
1842 .TE

1844 .SH SEE ALSO
1845 .sp
1846 .LP
1847 \fBas\fR(1), \fBcrle\fR(1), \fBgprof\fR(1), \fBld.so.1\fR(1), \fBldd\fR(1),
1848 \fBmcs\fR(1), \fBpvs\fR(1), \fBexec\fR(2), \fBstat\fR(2), \fBdlopen\fR(3C),
1849 \fBdldump\fR(3C), \fBelf\fR(3ELF), \fBar.h\fR(3HEAD), \fBa.out\fR(4),
1850 \fBattributes\fR(5)
1851 .sp
1852 .LP
1853 \fILinker and Libraries Guide\fR
1854 .SH NOTES
1855 .sp
1856 .LP
1857 Default options applied by \fBld\fR are maintained for historic reasons. In
1858 today's programming environment, where dynamic objects dominate, alternative
1859 defaults would often make more sense. However, historic defaults must be
1860 maintained to ensure compatibility with existing program development
1861 environments. Historic defaults are called out wherever possible in this
1862 manual. For a description of the current recommended options, see Appendix A,
1863 \fILink-Editor Quick Reference,\fR in \fILinker and Libraries Guide\fR.
1864 .sp
1865 .LP
1866 If the file being created by \fBld\fR already exists, the file is unlinked
1867 after all input files have been processed. A new file with the specified name
1868 is then created. This allows \fBld\fR to create a new version of the file,
1869 while simultaneously allowing existing processes that are accessing the old
1870 file contents to continue running. If the old file has no other links, the disk
1871 space of the removed file is freed when the last process referencing the file
1872 terminates.
1873 .sp
1874 .LP
1875 The behavior of \fBld\fR when the file being created already exists was changed
1876 with \fBSXCE\fR build \fB43\fR. In older versions, the existing file was
1877 rewritten in place, an approach with the potential to corrupt any running
1878 processes that is using the file. This change has an implication for output
1879 files that have multiple hard links in the file system. Previously, all links
1880 would remain intact, with all links accessing the new file contents. The new
1881 \fBld\fR behavior \fBbreaks\fR such links, with the result that only the
1882 specified output file name references the new file. All the other links
1883 continue to reference the old file. To ensure consistent behavior, applications
1884 that rely on multiple hard links to linker output files should explicitly
1885 remove and relink the other file names.
```

```
**********************************************************
    5247 Wed May 27 19:49:18 2015
new/usr/src/man/man1/psecflags.1
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
```

```
'\" te
.\" This file and its contents are supplied under the terms of the
.\" Common Development and Distribution License ("CDDL"), version 1.0.
.\" You may only use this file in accordance with the terms of version
.\" 1.0 of the CDDL.
.\"
.\" A full copy of the text of the CDDL should have accompanied this
.\" source.  A copy of the CDDL is also available via the Internet at
.\" http://www.illumos.org/license/CDDL.
.\"
.\"
.TH "PSECFLAGS" "1" "May 3, 2014"
.SH "NAME"
\fBpsecflags\fR - inspect or modify process security flags
.SH "SYNOPSIS"
.LP
.nf
\fB/usr/bin/psecflags\fR \fI-s\fR [-+]flags \fI-e\fR \fIcommand\fR
        [\fIarg\fR]...
.fi
.LP
.nf
\fB/usr/bin/psecflags\fR \fI-s\fR [-+]flags [\fI-i\fR \fIidtype\fR]
        \fIid\fR ...
.fi
.LP
.nf
\fB/usr/bin/psecflags\fR [\fI-F\fR] { \fIpid\fR | \fIcore\fR }
.fi
.LP
.nf
\fB/usr/bin/psecflags\fR \fI-l\fR
.fi

.SH "DESCRIPTION"
The first invocation of the \fBpsecflags\fR command runs the specified
\fIcommand\fR with the security-flags modified as described by the \fI-s\fR
argument.
.P
The second invocation modifies the security-flags of the processes described
by \fIidtype\fR and \fIid\fR according as described by the \fI-s\fR argument.
.P
The third invocation describes the security-flags of the specified processes
or core files.  The effective set is signified by '\fBE\fR', and the
inheritable set by '\fBI\fR'
.P
The fourth invocation lists the supported process security-flags

.SH "OPTIONS"
The following options are supported:
.sp
.ne 2
.na
\fB-e\fR
.ad
.RS 11n
Interpret the remaining arguments as a command line and run the command with
the security-flags specified with the \fI-s\fR flag.
```



```
.RE

.sp
.ne 2
.na
\fB-F\fR
.ad
.RS 11n
Force. Grab the target process even if another process has control.
.RE

.sp
.ne 2
.na
\fB-i\fR \fIidtype\fR
.ad
.RS 11n
This option, together with the \fIid\fR arguments specify one or more
processes whose security-flags will be modified. The interpretation of the
\fIid\fR arguments is based on \fIidtype\fR. If \fIidtype\fR is omitted the
default is \fBpid\fR.

Valid \fIidtype\fR options are:
.sp
.ne 2
.na
\fBall\fR
.ad
.RS 11n
The \fBpsecflags\fR command applies to all processes
.RE

.sp
.ne 2
.na
\fBcontract\fR, \fBctid\fR
.ad
.RS 11n
The security-flags of any process with a contract ID matching the \fIid\fR
arguments are modified.
.RE

.sp
.ne 2
.na
\fBgroup\fR, \fBgid\fR
.ad
.RS 11n
The security-flags of any process with a group ID matching the \fIid\fR
arguments are modified.
.RE

.sp
.ne 2
.na
\fBpid\fR
.ad
.RS 11n
The security-flags of any process with a process ID matching the \fIid\fR
arguments are modified. This is the default.
.RE

.sp
.ne 2
.na
\fBppid\fR
```

```
.ad
.RS 11n
The security-flags of any processes whose parent process ID matches the
\fIid\fR arguments are modified.
.RE

.sp
.ne 2
.na
\fBproject\fR, \fBprojid\fR
.ad
.RS 11n
The security-flags of any process whose project ID matches the \fIid\fR
arguments are modified.
.RE

.sp
.ne 2
.na
\fBsession\fR, \fBsid\fR
.ad
.RS 11n
The security-flags of any process whose session ID matches the \fIid\fR
arguments are modified.
.RE

.sp
.ne 2
.na
\fBtaskid\fR
.ad
.RS 11n
The security-flags of any process whose task ID matches the \fIid\fR arguments
are modified.
.RE

.sp
.ne 2
.na
\fBuser\fR, \fBuid\fR
.ad
.RS 11n
The security-flags of any process belonging to the users matching the \fIid\fR
arguments are modified.
.RE

.sp
.ne 2
.na
\fBzone\fR, \fBzoneid\fR
.ad
.RS 11n
The security-flags of any process running in the zones matching the given
\fIid\fR arguments are modified
.RE
.RE

.sp
.ne 2
.na
\fB-l\fR
.ad
.RS 11n
List all supported process security-flags
.RE
```



```
.sp
.ne 2
.na
\fB-s\fR \fIspecification\fR
.ad
.RS 11n
Modify the process security-flags according to
\fIspecification\fR. Specifications take the form \fB[-+]flagspec\fR. Where
\fB+\fR indicates that the given flags should be enabled in addition to the
current flags, \fB-\fR indicates the given flags should be disabled, and the
default (with neither) the given flags should replace the current flags.
.P
\fBflagspec\fR is a comma-separated list of security flags, or the string
\fB"none"\fR, which indicates that the security-flags are to be cleared.
.P
For a list of valid security-flags, see \fBpsecflags -l\fR
.RE

.SH "EXAMPLES"
.LP
\fBExample 1\fR Display the security-flags of the current shell
.sp
.in +2
.nf
example$ \fBpsecflags $$\fR
100718: -sh
        E:      aslr
        I:      aslr
.fi
.in -2
.sp

.LP
\fBExample 2\fR Run a user command with ASLR enabled in addition to any
inherited security flags.
.sp
.in +2
.nf
example$ \fBpsecflags -s +aslr -e /bin/sh\fR
$ psecflags $$
100724: -sh
        E:      none
        I:      aslr
.fi
.in -2
.sp

.LP
\fBExample 3\fR Remove aslr from the inheritable flags of all Bob’s processes.
.sp
.in +2
.nf
example# \fBpsecflags -s -aslr -i uid bob\fR
.fi
.in -2

.SH "EXIT STATUS"
The following exit values are returned:

.TP
\fB0\fR
.IP
Success

.TP
\fBnon-zero\fR
```

```
.IP
An error has occured

.SH "ATTRIBUTES"
.sp
.LP
See \fBattributes\fR(5) for descriptions of the following attributes:
.sp

.sp
.TS
box;
c | c
l | l .
ATTRIBUTE TYPE  ATTRIBUTE VALUE
_
Interface Stability     Volatile
.TE

.SH "SEE ALSO"
.BR exec (2),
.BR attributes (5),
.BR contract (4),
.BR security-flags (5),
.BR zones (5)
#endif /* ! codereview */
```

```
**********************************************************
    4047 Wed May 27 19:49:19 2015
new/usr/src/man/man5/Makefile
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
#
# This file and its contents are supplied under the terms of the
# Common Development and Distribution License ("CDDL"), version 1.0.
# You may only use this file in accordance with the terms of version
# 1.0 of the CDDL.
#
#
# Copyright 2011, Richard Lowe
# Copyright (c) 2012 by Delphix. All rights reserved.
# Copyright 2014 Nexenta Systems, Inc.
# Copyright 2014 Garrett D'Amore <garrett@damore.org>
#

include         $(SRC)/Makefile.master

MANSECT=        5

MANFILES=       Intro.5                 \
                acl.5                   \
                ad.5                    \
                ascii.5                 \
                attributes.5            \
                audit_binfile.5         \
                audit_remote.5          \
                audit_syslog.5          \
                brands.5                \
                cancellation.5          \
                charmap.5               \
                condition.5             \
                crypt_bsdbf.5           \
                crypt_bsdmd5.5          \
                crypt_sha256.5          \
                crypt_sha512.5          \
                crypt_sunmd5.5          \
                crypt_unix.5            \
                device_clean.5          \
                dhcp.5                  \
                environ.5               \
                eqn.5                   \
                eqnchar.5               \
                extendedFILE.5          \
                filesystem.5            \
                fnmatch.5               \
                formats.5               \
                fsattr.5                \
                grub.5                  \
                gss_auth_rules.5        \
                hal.5                   \
                iconv.5                 \
                iconv_unicode.5         \
                ieee802.3.5             \
                ieee802.11.5            \
                ipfilter.5              \
```



```
                isalist.5               \
                kerberos.5              \
                krb5_auth_rules.5       \
                krb5envvar.5            \
                largefile.5             \
                lf64.5                  \
                lfcompile.5             \
                lfcompile64.5           \
                locale.5                \
                man.5                   \
                mandoc_char.5           \
                mandoc_roff.5           \
                mdoc.5                  \
                me.5                    \
                mech_spnego.5           \
                mm.5                    \
                ms.5                    \
                mutex.5                 \
                nfssec.5                \
                pam_allow.5             \
                pam_authtok_check.5     \
                pam_authtok_get.5       \
                pam_authtok_store.5     \
                pam_deny.5              \
                pam_dhkeys.5            \
                pam_dial_auth.5         \
                pam_krb5.5              \
                pam_krb5_migrate.5      \
                pam_ldap.5              \
                pam_list.5              \
                pam_passwd_auth.5       \
                pam_rhosts_auth.5       \
                pam_roles.5             \
                pam_sample.5            \
                pam_smb_passwd.5        \
                pam_smbfs_login.5       \
                pam_timestamp.5         \
                pam_tsol_account.5      \
                pam_unix_account.5      \
                pam_unix_auth.5         \
                pam_unix_cred.5         \
                pam_unix_session.5      \
                pkcs11_kernel.5         \
                pkcs11_softtoken.5      \
                pkcs11_tpm.5            \
                privileges.5            \
                prof.5                  \
                rbac.5                  \
                regex.5                 \
                regexp.5                \
                resource_controls.5     \
                security-flags.5        \
#endif /* ! codereview */
                smf.5                   \
                smf_bootstrap.5         \
                smf_method.5            \
                smf_restarter.5         \
                smf_security.5          \
                smf_template.5          \
                standards.5             \
                sticky.5                \
                tbl.5                   \
                tecla.5                 \
                term.5                  \
                threads.5               \
                trusted_extensions.5    \
```

```
125                 vgrindefs.5             \
126                 zones.5                 \
127                 zpool-features.5

129 MANLINKS=       ANSI.5          \
130                 C++.5           \
131                 C.5             \
132                 CSI.5           \
133                 ISO.5           \
134                 MT-Level.5      \
135                 POSIX.1.5       \
136                 POSIX.2.5       \
137                 POSIX.5         \
138                 RBAC.5          \
139                 SUS.5           \
140                 SUSv2.5         \
141                 SUSv3.5         \
142                 SVID.5          \
143                 SVID3.5         \
144                 XNS.5           \
145                 XNS4.5          \
146                 XNS5.5          \
147                 XPG.5           \
148                 XPG3.5          \
149                 XPG4.5          \
150                 XPG4v2.5        \
151                 advance.5       \
152                 architecture.5  \
153                 availability.5  \
154                 compile.5       \
155                 intro.5         \
156                 pthreads.5      \
157                 stability.5     \
158                 standard.5      \
159                 step.5          \
160                 teclarc.5

162 intro.5         := LINKSRC = Intro.5

164 CSI.5           := LINKSRC = attributes.5
165 MT-Level.5      := LINKSRC = attributes.5
166 architecture.5  := LINKSRC = attributes.5
167 availability.5  := LINKSRC = attributes.5
168 stability.5     := LINKSRC = attributes.5
169 standard.5      := LINKSRC = attributes.5

171 RBAC.5          := LINKSRC = rbac.5

173 advance.5       := LINKSRC = regexp.5
174 compile.5       := LINKSRC = regexp.5
175 step.5          := LINKSRC = regexp.5

177 ANSI.5          := LINKSRC = standards.5
178 C++.5           := LINKSRC = standards.5
179 C.5             := LINKSRC = standards.5
180 ISO.5           := LINKSRC = standards.5
181 POSIX.1.5       := LINKSRC = standards.5
182 POSIX.2.5       := LINKSRC = standards.5
183 POSIX.5         := LINKSRC = standards.5
184 SUS.5           := LINKSRC = standards.5
185 SUSv2.5         := LINKSRC = standards.5
186 SUSv3.5         := LINKSRC = standards.5
187 SVID.5          := LINKSRC = standards.5
188 SVID3.5         := LINKSRC = standards.5
189 XNS.5           := LINKSRC = standards.5
190 XNS4.5          := LINKSRC = standards.5
```



```
191 XNS5.5          := LINKSRC = standards.5
192 XPG.5           := LINKSRC = standards.5
193 XPG3.5          := LINKSRC = standards.5
194 XPG4.5          := LINKSRC = standards.5
195 XPG4v2.5        := LINKSRC = standards.5

197 teclarc.5       := LINKSRC = tecla.5

199 pthreads.5      := LINKSRC = threads.5

201 .KEEP_STATE:

203 include         $(SRC)/man/Makefile.man

205 install:         $(ROOTMANFILES) $(ROOTMANLINKS)
```

```
**********************************************************
   33356 Wed May 27 19:49:19 2015
new/usr/src/man/man5/privileges.5
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
```

```
   1 '\" te
   2 .\" Copyright (c) 2009, Sun Microsystems, Inc. All Rights Reserved.
   3 .\" Copyright 2013, Joyent, Inc. All Rights Reserved.
   4 .\" The contents of this file are subject to the terms of the Common Development
   5 .\"  See the License for the specific language governing permissions and limitat
   6 .\" the fields enclosed by brackets "[]" replaced with your own identifying info
   7 .TH PRIVILEGES 5 "Feb 3, 2015"
   8 .SH NAME
   9 privileges \- process privilege model
  10 .SH DESCRIPTION
  11 .LP
  12 Solaris software implements a set of privileges that provide fine-grained
  13 control over the actions of processes. The possession of a certain privilege
  14 allows a process to perform a specific set of restricted operations.
  15 .sp
  16 .LP
  17 The change to a primarily privilege-based security model in the Solaris
  18 operating system gives developers an opportunity to restrict processes to those
  19 privileged operations actually needed instead of all (super-user) or no
  20 privileges (non-zero UIDs). Additionally, a set of previously unrestricted
  21 operations now requires a privilege; these privileges are dubbed the "basic"
  22 privileges and are by default given to all processes.
  23 .sp
  24 .LP
  25 Taken together, all defined privileges with the exception of the "basic"
  26 privileges compose the set of privileges that are traditionally associated with
  27 the root user. The "basic" privileges are "privileges" unprivileged processes
  28 were accustomed to having.
  29 .sp
  30 .LP
  31 The defined privileges are:
  32 .sp
  33 .ne 2
  34 .na
  35 \fB\fBPRIV_CONTRACT_EVENT\fR\fR
  36 .ad
  37 .sp .6
  38 .RS 4n
  39 Allow a process to request reliable delivery of events to an event endpoint.
  40 .sp
  41 Allow a process to include events in the critical event set term of a template
  42 which could be generated in volume by the user.
  43 .RE

  45 .sp
  46 .ne 2
  47 .na
  48 \fB\fBPRIV_CONTRACT_IDENTITY\fR\fR
  49 .ad
  50 .sp .6
  51 .RS 4n
  52 Allows a process to set the service FMRI value of a process contract template.
  53 .RE

  55 .sp
  56 .ne 2
  57 .na
  58 \fB\fBPRIV_CONTRACT_OBSERVER\fR\fR
```



```
  59 .ad
  60 .sp .6
  61 .RS 4n
  62 Allow a process to observe contract events generated by contracts created and
  63 owned by users other than the process's effective user ID.
  64 .sp
  65 Allow a process to open contract event endpoints belonging to contracts created
  66 and owned by users other than the process's effective user ID.
  67 .RE

  69 .sp
  70 .ne 2
  71 .na
  72 \fB\fBPRIV_CPC_CPU\fR\fR
  73 .ad
  74 .sp .6
  75 .RS 4n
  76 Allow a process to access per-CPU hardware performance counters.
  77 .RE

  79 .sp
  80 .ne 2
  81 .na
  82 \fB\fBPRIV_DTRACE_KERNEL\fR\fR
  83 .ad
  84 .sp .6
  85 .RS 4n
  86 Allow DTrace kernel-level tracing.
  87 .RE

  89 .sp
  90 .ne 2
  91 .na
  92 \fB\fBPRIV_DTRACE_PROC\fR\fR
  93 .ad
  94 .sp .6
  95 .RS 4n
  96 Allow DTrace process-level tracing. Allow process-level tracing probes to be
  97 placed and enabled in processes to which the user has permissions.
  98 .RE

 100 .sp
 101 .ne 2
 102 .na
 103 \fB\fBPRIV_DTRACE_USER\fR\fR
 104 .ad
 105 .sp .6
 106 .RS 4n
 107 Allow DTrace user-level tracing. Allow use of the syscall and profile DTrace
 108 providers to examine processes to which the user has permissions.
 109 .RE

 111 .sp
 112 .ne 2
 113 .na
 114 \fB\fBPRIV_FILE_CHOWN\fR\fR
 115 .ad
 116 .sp .6
 117 .RS 4n
 118 Allow a process to change a file's owner user ID. Allow a process to change a
 119 file's group ID to one other than the process's effective group ID or one of
 120 the process's supplemental group IDs.
 121 .RE

 123 .sp
 124 .ne 2
```

```
.na
\fB\fBPRIV_FILE_CHOWN_SELF\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to give away its files. A process with this privilege runs as
if {\fB_POSIX_CHOWN_RESTRICTED\fR} is not in effect.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_FILE_DAC_EXECUTE\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to execute an executable file whose permission bits or ACL
would otherwise disallow the process execute permission.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_FILE_DAC_READ\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to read a file or directory whose permission bits or ACL would
otherwise disallow the process read permission.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_FILE_DAC_SEARCH\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to search a directory whose permission bits or ACL would not
otherwise allow the process search permission.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_FILE_DAC_WRITE\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to write a file or directory whose permission bits or ACL do
not allow the process write permission. All privileges are required to write
files owned by UID 0 in the absence of an effective UID of 0.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_FILE_DOWNGRADE_SL\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to set the sensitivity label of a file or directory to a
sensitivity label that does not dominate the existing sensitivity label.
.sp
This privilege is interpreted only if the system is configured with Trusted
Extensions.
```



```
.RE

.sp
.ne 2
.na
\fB\fBPRIV_FILE_FLAG_SET\fR\fR
.ad
.sp .6
.RS 4n
Allows a process to set immutable, nounlink or appendonly file attributes.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_FILE_LINK_ANY\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to create hardlinks to files owned by a UID different from the
process's effective UID.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_FILE_OWNER\fR\fR
.ad
.sp .6
.RS 4n
Allow a process that is not the owner of a file to modify that file's access
and modification times. Allow a process that is not the owner of a directory to
modify that directory's access and modification times. Allow a process that is
not the owner of a file or directory to remove or rename a file or directory
whose parent directory has the "save text image after execution" (sticky) bit
set. Allow a process that is not the owner of a file to mount a \fBnamefs\fR
upon that file. Allow a process that is not the owner of a file or directory to
modify that file's or directory's permission bits or ACL.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_FILE_READ\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to read objects in the filesystem.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_FILE_SETID\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to change the ownership of a file or write to a file without
the set-user-ID and set-group-ID bits being cleared. Allow a process to set the
set-group-ID bit on a file or directory whose group is not the process's
effective group or one of the process's supplemental groups. Allow a process to
set the set-user-ID bit on a file with different ownership in the presence of
\fBPRIV_FILE_OWNER\fR. Additional restrictions apply when creating or modifying
a setuid 0 file.
.RE
```

```
.sp
.ne 2
.na
\fB\fBPRIV_FILE_UPGRADE_SL\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to set the sensitivity label of a file or directory to a
sensitivity label that dominates the existing sensitivity label.
.sp
This privilege is interpreted only if the system is configured with Trusted
Extensions.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_FILE_WRITE\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to modify objects in the filesytem.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_GRAPHICS_ACCESS\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to make privileged ioctls to graphics devices. Typically only
an xserver process needs to have this privilege. A process with this privilege
is also allowed to perform privileged graphics device mappings.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_GRAPHICS_MAP\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to perform privileged mappings through a graphics device.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_IPC_DAC_READ\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to read a System V IPC Message Queue, Semaphore Set, or Shared
Memory Segment whose permission bits would not otherwise allow the process read
permission.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_IPC_DAC_WRITE\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to write a System V IPC Message Queue, Semaphore Set, or Shared
```



```
Memory Segment whose permission bits would not otherwise allow the process
write permission.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_IPC_OWNER\fR\fR
.ad
.sp .6
.RS 4n
Allow a process that is not the owner of a System V IPC Message Queue,
Semaphore Set, or Shared Memory Segment to remove, change ownership of, or
change permission bits of the Message Queue, Semaphore Set, or Shared Memory
Segment.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_NET_ACCESS\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to open a TCP, UDP, SDP, or SCTP network endpoint.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_NET_BINDMLP\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to bind to a port that is configured as a multi-level port
(MLP) for the process’s zone. This privilege applies to both shared address and
zone-specific address MLPs. See \fBtnzonecfg\fR(\fB4\fR) from the Trusted
Extensions manual pages for information on configuring MLP ports.
.sp
This privilege is interpreted only if the system is configured with Trusted
Extensions.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_NET_ICMPACCESS\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to send and receive ICMP packets.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_NET_MAC_AWARE\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to set the \fBNET_MAC_AWARE\fR process flag by using
\fBsetpflags\fR(2). This privilege also allows a process to set the
\fBSO_MAC_EXEMPT\fR socket option by using \fBsetsockopt\fR(3SOCKET). The
\fBNET_MAC_AWARE\fR process flag and the \fBSO_MAC_EXEMPT\fR socket option both
allow a local process to communicate with an unlabeled peer if the local
process’s label dominates the peer’s default label, or if the local process
```

runs in the global zone.
.sp
This privilege is interpreted only if the system is configured with Trusted Extensions.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_NET_MAC_IMPLICIT\fR\fR
.ad
.sp .6
.RS 4n
Allow a proces to set \fBSO_MAC_IMPLICIT\fR option by using \fBsetsockopt\fR(3SOCKET). This allows a privileged process to transmit implicitly-labeled packets to a peer.
.sp
This privilege is interpreted only if the system is configured with Trusted Extensions.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_NET_OBSERVABILITY\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to open a device for just receiving network traffic, sending traffic is disallowed.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_NET_PRIVADDR\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to bind to a privileged port number. The privilege port numbers are 1-1023 (the traditional UNIX privileged ports) as well as those ports marked as "\fBudp/tcp_extra_priv_ports\fR" with the exception of the ports reserved for use by NFS and SMB.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_NET_RAWACCESS\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to have direct access to the network layer.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_PROC_AUDIT\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to generate audit records. Allow a process to get its own audit pre-selection information.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_PROC_CHROOT\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to change its root directory.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_PROC_CLOCK_HIGHRES\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to use high resolution timers.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_PROC_EXEC\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to call \fBexec\fR(2).
.RE

.sp
.ne 2
.na
\fB\fBPRIV_PROC_FORK\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to call \fBfork\fR(2), \fBfork1\fR(2), or \fBvfork\fR(2).
.RE

.sp
.ne 2
.na
\fB\fBPRIV_PROC_INFO\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to examine the status of processes other than those to which it can send signals. Processes that cannot be examined cannot be seen in \fB/proc\fR and appear not to exist.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_PROC_LOCK_MEMORY\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to lock pages in physical memory.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_PROC_OWNER\fR\fR

```
.ad
.sp .6
.RS 4n
Allow a process to send signals to other processes and inspect and modify the
process state in other processes, regardless of ownership. When modifying
another process, additional restrictions apply: the effective privilege set of
the attaching process must be a superset of the target process's effective,
permitted, and inheritable sets; the limit set must be a superset of the
target's limit set; if the target process has any UID set to 0 all privilege
must be asserted unless the effective UID is 0. Allow a process to bind
arbitrary processes to CPUs.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_PROC_PRIOUP\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to elevate its priority above its current level.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_PROC_PRIOCNTL\fR\fR
.ad
.sp .6
.RS 4n
Allows all that PRIV_PROC_PRIOUP allows.
Allow a process to change its scheduling class to any scheduling class,
including the RT class.
.RE

.sp
.ne 2
.na
\fB\PRIV_PROC_SECFLAGS\fR
.ad
.sp .6
.RS 4n
Allow a process to manipulate the secflags of processes (subject to,
additionally, the ability to signal that process)
.RE

.sp
.ne 2
.na
#endif /* ! codereview */
\fB\fBPRIV_PROC_SESSION\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to send signals or trace processes outside its session.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_PROC_SETID\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to set its UIDs at will, assuming UID 0 requires all privileges
to be asserted.
```



```
.RE

.sp
.ne 2
.na
\fB\fBPRIV_PROC_TASKID\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to assign a new task ID to the calling process.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_PROC_ZONE\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to trace or send signals to processes in other zones. See
\fBzones\fR(5).
.RE

.sp
.ne 2
.na
\fB\fBPRIV_SYS_ACCT\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to enable and disable and manage accounting through
\fBacct\fR(2).
.RE

.sp
.ne 2
.na
\fB\fBPRIV_SYS_ADMIN\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to perform system administration tasks such as setting node and
domain name and specifying \fBcoreadm\fR(1M) and \fBnscd\fR(1M) settings
.RE

.sp
.ne 2
.na
\fB\fBPRIV_SYS_AUDIT\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to start the (kernel) audit daemon. Allow a process to view and
set audit state (audit user ID, audit terminal ID, audit sessions ID, audit
pre-selection mask). Allow a process to turn off and on auditing. Allow a
process to configure the audit parameters (cache and queue sizes, event to
class mappings, and policy options).
.RE

.sp
.ne 2
.na
\fB\fBPRIV_SYS_CONFIG\fR\fR
.ad
.sp .6
.RS 4n
```

```
653 Allow a process to perform various system configuration tasks. Allow
654 filesystem-specific administrative procedures, such as filesystem configuration
655 ioctls, quota calls, creation and deletion of snapshots, and manipulating the
656 PCFS bootsector.
657 .RE

659 .sp
660 .ne 2
661 .na
662 \fB\fBPRIV_SYS_DEVICES\fR\fR
663 .ad
664 .sp .6
665 .RS 4n
666 Allow a process to create device special files. Allow a process to successfully
667 call a kernel module that calls the kernel \fBdrv_priv\fR(9F) function to check
668 for allowed access. Allow a process to open the real console device directly.
669 Allow a process to open devices that have been exclusively opened.
670 .RE

672 .sp
673 .ne 2
674 .na
675 \fB\fBPRIV_SYS_DL_CONFIG\fR\fR
676 .ad
677 .sp .6
678 .RS 4n
679 Allow a process to configure a system's datalink interfaces.
680 .RE

682 .sp
683 .ne 2
684 .na
685 \fB\fBPRIV_SYS_IP_CONFIG\fR\fR
686 .ad
687 .sp .6
688 .RS 4n
689 Allow a process to configure a system's IP interfaces and routes. Allow a
690 process to configure network parameters for \fBTCP/IP\fR using \fBndd\fR. Allow
691 a process access to otherwise restricted \fBTCP/IP\fR information using
692 \fBndd\fR. Allow a process to configure \fBIPsec\fR. Allow a process to pop
693 anchored \fBSTREAM\fRs modules with matching \fBzoneid\fR.
694 .RE

696 .sp
697 .ne 2
698 .na
699 \fB\fBPRIV_SYS_IPC_CONFIG\fR\fR
700 .ad
701 .sp .6
702 .RS 4n
703 Allow a process to increase the size of a System V IPC Message Queue buffer.
704 .RE

706 .sp
707 .ne 2
708 .na
709 \fB\fBPRIV_SYS_IPTUN_CONFIG\fR\fR
710 .ad
711 .sp .6
712 .RS 4n
713 Allow a process to configure IP tunnel links.
714 .RE

716 .sp
717 .ne 2
718 .na
```



```
719 \fB\fBPRIV_SYS_LINKDIR\fR\fR
720 .ad
721 .sp .6
722 .RS 4n
723 Allow a process to unlink and link directories.
724 .RE

726 .sp
727 .ne 2
728 .na
729 \fB\fBPRIV_SYS_MOUNT\fR\fR
730 .ad
731 .sp .6
732 .RS 4n
733 Allow a process to mount and unmount filesystems that would otherwise be
734 restricted (that is, most filesystems except \fBnamefs\fR). Allow a process to
735 add and remove swap devices.
736 .RE

738 .sp
739 .ne 2
740 .na
741 \fB\fBPRIV_SYS_NET_CONFIG\fR\fR
742 .ad
743 .sp .6
744 .RS 4n
745 Allow a process to do all that \fBPRIV_SYS_IP_CONFIG\fR,
746 \fBPRIV_SYS_DL_CONFIG\fR, and \fBPRIV_SYS_PPP_CONFIG\fR allow, plus the
747 following: use the \fBrpcmod\fR STREAMS module and insert/remove STREAMS
748 modules on locations other than the top of the module stack.
749 .RE

751 .sp
752 .ne 2
753 .na
754 \fB\fBPRIV_SYS_NFS\fR\fR
755 .ad
756 .sp .6
757 .RS 4n
758 Allow a process to provide NFS service: start NFS kernel threads, perform NFS
759 locking operations, bind to NFS reserved ports: ports 2049 (\fBnfs\fR) and port
760 4045 (\fBlockd\fR).
761 .RE

763 .sp
764 .ne 2
765 .na
766 \fB\fBPRIV_SYS_PPP_CONFIG\fR\fR
767 .ad
768 .sp .6
769 .RS 4n
770 Allow a process to create, configure, and destroy PPP instances with pppd(1M)
771 \fBpppd\fR(1M) and control PPPoE plumbing with \fBsppptun\fR(1M)sppptun(1M).
772 This privilege is granted by default to exclusive IP stack instance zones.
773 .RE

775 .sp
776 .ne 2
777 .na
778 \fB\fBPRIV_SYS_RES_BIND\fR\fR
779 .ad
780 .sp .6
781 .RS 4n
782 Allows a process to bind processes to processor sets.
783 .RE
```

```
.sp
.ne 2
.na
\fB\fBPRIV_SYS_RES_CONFIG\fR\fR
.ad
.sp .6
.RS 4n
Allows all that PRIV_SYS_RES_BIND allows.
Allow a process to create and delete processor sets, assign CPUs to processor
sets and override the \fBPSET_NOESCAPE\fR property. Allow a process to change
the operational status of CPUs in the system using \fBp_online\fR(2). Allow a
process to configure filesystem quotas. Allow a process to configure resource
pools and bind processes to pools.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_SYS_RESOURCE\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to exceed the resource limits imposed on it by
\fBsetrlimit\fR(2) and \fBsetrctl\fR(2).
.RE

.sp
.ne 2
.na
\fB\fBPRIV_SYS_SMB\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to provide NetBIOS or SMB services: start SMB kernel threads or
bind to NetBIOS or SMB reserved ports: ports 137, 138, 139 (NetBIOS) and 445
(SMB).
.RE

.sp
.ne 2
.na
\fB\fBPRIV_SYS_SUSER_COMPAT\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to successfully call a third party loadable module that calls
the kernel \fBsuser()\fR function to check for allowed access. This privilege
exists only for third party loadable module compatibility and is not used by
Solaris proper.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_SYS_TIME\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to manipulate system time using any of the appropriate system
calls: \fBstime\fR(2), \fBadjtime\fR(2), and \fBntp_adjtime\fR(2).
.RE

.sp
.ne 2
.na
\fB\fBPRIV_SYS_TRANS_LABEL\fR\fR
```



```
.ad
.sp .6
.RS 4n
Allow a process to translate labels that are not dominated by the process's
sensitivity label to and from an external string form.
.sp
This privilege is interpreted only if the system is configured with Trusted
Extensions.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_VIRT_MANAGE\fR\fR
.ad
.sp .6
.RS 4n
Allows a process to manage virtualized environments such as \fBxVM\fR(5).
.RE

.sp
.ne 2
.na
\fB\fBPRIV_WIN_COLORMAP\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to override colormap restrictions.
.sp
Allow a process to install or remove colormaps.
.sp
Allow a process to retrieve colormap cell entries allocated by other processes.
.sp
This privilege is interpreted only if the system is configured with Trusted
Extensions.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_WIN_CONFIG\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to configure or destroy resources that are permanently retained
by the X server.
.sp
Allow a process to use SetScreenSaver to set the screen saver timeout value
.sp
Allow a process to use ChangeHosts to modify the display access control list.
.sp
Allow a process to use GrabServer.
.sp
Allow a process to use the SetCloseDownMode request that can retain window,
pixmap, colormap, property, cursor, font, or graphic context resources.
.sp
This privilege is interpreted only if the system is configured with Trusted
Extensions.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_WIN_DAC_READ\fR\fR
.ad
.sp .6
```

```
.RS 4n
Allow a process to read from a window resource that it does not own (has a
different user ID).
.sp
This privilege is interpreted only if the system is configured with Trusted
Extensions.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_WIN_DAC_WRITE\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to write to or create a window resource that it does not own
(has a different user ID). A newly created window property is created with the
window's user ID.
.sp
This privilege is interpreted only if the system is configured with Trusted
Extensions.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_WIN_DEVICES\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to perform operations on window input devices.
.sp
Allow a process to get and set keyboard and pointer controls.
.sp
Allow a process to modify pointer button and key mappings.
.sp
This privilege is interpreted only if the system is configured with Trusted
Extensions.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_WIN_DGA\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to use the direct graphics access (DGA) X protocol extensions.
Direct process access to the frame buffer is still required. Thus the process
must have MAC and DAC privileges that allow access to the frame buffer, or the
frame buffer must be allocated to the process.
.sp
This privilege is interpreted only if the system is configured with Trusted
Extensions.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_WIN_DOWNGRADE_SL\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to set the sensitivity label of a window resource to a
sensitivity label that does not dominate the existing sensitivity label.
.sp
```



```
This privilege is interpreted only if the system is configured with Trusted
Extensions.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_WIN_FONTPATH\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to set a font path.
.sp
This privilege is interpreted only if the system is configured with Trusted
Extensions.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_WIN_MAC_READ\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to read from a window resource whose sensitivity label is not
equal to the process sensitivity label.
.sp
This privilege is interpreted only if the system is configured with Trusted
Extensions.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_WIN_MAC_WRITE\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to create a window resource whose sensitivity label is not
equal to the process sensitivity label. A newly created window property is
created with the window's sensitivity label.
.sp
This privilege is interpreted only if the system is configured with Trusted
Extensions.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_WIN_SELECTION\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to request inter-window data moves without the intervention of
the selection confirmer.
.sp
This privilege is interpreted only if the system is configured with Trusted
Extensions.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_WIN_UPGRADE_SL\fR\fR
.ad
.sp .6
```

```
.RS 4n
Allow a process to set the sensitivity label of a window resource to a
sensitivity label that dominates the existing sensitivity label.
.sp
This privilege is interpreted only if the system is configured with Trusted
Extensions.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_XVM_CONTROL\fR\fR
.ad
.sp .6
.RS 4n
Allows a process access to the \fBxVM\fR(5) control devices for managing guest
domains and the hypervisor. This privilege is used only if booted into xVM on
x86 platforms.
.RE

.sp
.LP
Of the privileges listed above, the privileges \fBPRIV_FILE_LINK_ANY\fR,
\fBPRIV_PROC_INFO\fR, \fBPRIV_PROC_SESSION\fR, \fBPRIV_PROC_FORK\fR and
\fBPRIV_PROC_EXEC\fR are considered "basic" privileges. These are privileges
that used to be always available to unprivileged processes. By default,
processes still have the basic privileges.
.sp
.LP
The privileges \fBPRIV_PROC_SETID\fR and \fBPRIV_PROC_AUDIT\fR must be present
in the Limit set (see below) of a process in order for set-uid root \fBexec\fRs
to be successful, that is, get an effective UID of 0 and additional privileges.
.sp
.LP
The privilege implementation in Solaris extends the process credential with
four privilege sets:
.sp
.ne 2
.na
\fBI, the inheritable set\fR
.ad
.RS 26n
The privileges inherited on \fBexec\fR.
.RE

.sp
.ne 2
.na
\fBP, the permitted set\fR
.ad
.RS 26n
The maximum set of privileges for the process.
.RE

.sp
.ne 2
.na
\fBE, the effective set\fR
.ad
.RS 26n
The privileges currently in effect.
.RE

.sp
.ne 2
.na
```



```
\fBL, the limit set\fR
.ad
.RS 26n
The upper bound of the privileges a process and its offspring can obtain.
Changes to L take effect on the next \fBexec\fR.
.RE

.sp
.LP
The sets I, P and E are typically identical to the basic set of privileges for
unprivileged processes. The limit set is typically the full set of privileges.
.sp
.LP
Each process has a Privilege Awareness State (PAS) that can take the value PA
(privilege-aware) and NPA (not-PA). PAS is a transitional mechanism that allows
a choice between full compatibility with the old superuser model and completely
ignoring the effective UID.
.sp
.LP
To facilitate the discussion, we introduce the notion of "observed effective
set" (oE) and "observed permitted set" (oP) and the implementation sets iE and
iP.
.sp
.LP
A process becomes privilege-aware either by manipulating the effective,
permitted, or limit privilege sets through \fBsetppriv\fR(2) or by using
\fBsetpflags\fR(2). In all cases, oE and oP are invariant in the process of
becoming privilege-aware. In the process of becoming privilege-aware, the
following assignments take place:
.sp
.in +2
.nf
iE = oE
iP = oP
.fi
.in -2

.sp
.LP
When a process is privilege-aware, oE and oP are invariant under UID changes.
When a process is not privilege-aware, oE and oP are observed as follows:
.sp
.in +2
.nf
oE = euid == 0 ? L : iE
oP = (euid == 0 || ruid == 0 || suid == 0) ? L : iP
.fi
.in -2

.sp
.LP
When a non-privilege-aware process has an effective UID of 0, it can exercise
the privileges contained in its limit set, the upper bound of its privileges.
If a non-privilege-aware process has any of the UIDs 0, it appears to be
capable of potentially exercising all privileges in L.
.sp
.LP
It is possible for a process to return to the non-privilege aware state using
\fBsetpflags()\fR. The kernel always attempts this on \fBexec\fR(2). This
operation is permitted only if the following conditions are met:
.RS +4
.TP
.ie t \(bu
.el o
If any of the UIDs is equal to 0, P must be equal to L.
.RE
```

```
.RS +4
.TP
.ie t \(bu
.el o
If the effective UID is equal to 0, E must be equal to L.
.RE
.sp
.LP
When a process gives up privilege awareness, the following assignments take
place:
.sp
.in +2
.nf
if (euid == 0) iE = L & I
if (any uid == 0) iP = L & I
.fi
.in -2

.sp
.LP
The privileges obtained when not having a UID of \fB0\fR are the inheritable
set of the process restricted by the limit set.
.sp
.LP
Only privileges in the process's (observed) effective privilege set allow the
process to perform restricted operations. A process can use any of the
privilege manipulation functions to add or remove privileges from the privilege
sets. Privileges can be removed always. Only privileges found in the permitted
set can be added to the effective and inheritable set. The limit set cannot
grow. The inheritable set can be larger than the permitted set.
.sp
.LP
When a process performs an \fBexec\fR(2), the kernel first tries to relinquish
privilege awareness before making the following privilege set modifications:
.sp
.in +2
.nf
E' = P' = I' = L & I
L is unchanged
.fi
.in -2

.sp
.LP
If a process has not manipulated its privileges, the privilege sets effectively
remain the same, as E, P and I are already identical.
.sp
.LP
The limit set is enforced at \fBexec\fR time.
.sp
.LP
To run a non-privilege-aware application in a backward-compatible manner, a
privilege-aware application should start the non-privilege-aware application
with I=basic.
.sp
.LP
For most privileges, absence of the privilege simply results in a failure. In
some instances, the absense of a privilege can cause system calls to behave
differently. In other instances, the removal of a privilege can force a set-uid
application to seriously malfunction. Privileges of this type are considered
"unsafe". When a process is lacking any of the unsafe privileges from its limit
set, the system does not honor the set-uid bit of set-uid root applications.
The following unsafe privileges have been identified: \fBproc_setid\fR,
\fBsys_resource\fR and \fBproc_audit\fR.
.SS "Privilege Escalation"
.LP
```



```
In certain circumstances, a single privilege could lead to a process gaining
one or more additional privileges that were not explicitly granted to that
process. To prevent such an escalation of privileges, the security policy
requires explicit permission for those additional privileges.
.sp
.LP
Common examples of escalation are those mechanisms that allow modification of
system resources through "raw'' interfaces; for example, changing kernel data
structures through \fB/dev/kmem\fR or changing files through \fB/dev/dsk/*\fR.
Escalation also occurs when a process controls processes with more privileges
than the controlling process. A special case of this is manipulating or
creating objects owned by UID 0 or trying to obtain UID 0 using
\fBsetuid\fR(2). The special treatment of UID 0 is needed because the UID 0
owns all system configuration files and ordinary file protection mechanisms
allow processes with UID 0 to modify the system configuration. With appropriate
file modifications, a given process running with an effective UID of 0 can gain
all privileges.
.sp
.LP
In situations where a process might obtain UID 0, the security policy requires
additional privileges, up to the full set of privileges. Such restrictions
could be relaxed or removed at such time as additional mechanisms for
protection of system files became available. There are no such mechanisms in
the current Solaris release.
.sp
.LP
The use of UID 0 processes should be limited as much as possible. They should
be replaced with programs running under a different UID but with exactly the
privileges they need.
.sp
.LP
Daemons that never need to \fBexec\fR subprocesses should remove the
\fBPRIV_PROC_EXEC\fR privilege from their permitted and limit sets.
.SS "Assigned Privileges and Safeguards"
.LP
When privileges are assigned to a user, the system administrator could give
that user more powers than intended. The administrator should consider whether
safeguards are needed. For example, if the \fBPRIV_PROC_LOCK_MEMORY\fR
privilege is given to a user, the administrator should consider setting the
\fBproject.max-locked-memory\fR resource control as well, to prevent that user
from locking all memory.
.SS "Privilege Debugging"
.LP
When a system call fails with a permission error, it is not always immediately
obvious what caused the problem. To debug such a problem, you can use a tool
called \fBprivilege debugging\fR. When privilege debugging is enabled for a
process, the kernel reports missing privileges on the controlling terminal of
the process. (Enable debugging for a process with the \fB-D\fR option of
\fBppriv\fR(1).) Additionally, the administrator can enable system-wide
privilege debugging by setting the \fBsystem\fR(4) variable \fBpriv_debug\fR
using:
.sp
.in +2
.nf
set priv_debug = 1
.fi
.in -2

.sp
.LP
On a running system, you can use \fBmdb\fR(1) to change this variable.
.SS "Privilege Administration"
.LP
The Solaris Management Console (see \fBsmc\fR(1M)) is the preferred method of
modifying privileges for a command. Use \fBusermod\fR(1M) or \fBsmrole\fR(1M)
to assign privileges to or modify privileges for, respectively, a user or a
```

```
1313 role. Use \fBppriv\fR(1) to enumerate the privileges supported on a system and
1314 \fBtruss\fR(1) to determine which privileges a program requires.
1315 .SH SEE ALSO
1316 .LP
1317 \fBmdb\fR(1), \fBppriv\fR(1), \fBadd_drv\fR(1M), \fBifconfig\fR(1M),
1318 \fBlockd\fR(1M), \fBnfsd\fR(1M), \fBpppd\fR(1M), \fBrem_drv\fR(1M),
1319 \fBsmbd\fR(1M), \fBspppptun\fR(1M), \fBupdate_drv\fR(1M), \fBIntro\fR(2),
1320 \fBaccess\fR(2), \fBacct\fR(2), \fBacl\fR(2), \fBadjtime\fR(2), \fBaudit\fR(2),
1321 \fBauditon\fR(2), \fBchmod\fR(2), \fBchown\fR(2), \fBchroot\fR(2),
1322 \fBcreat\fR(2), \fBexec\fR(2), \fBfcntl\fR(2), \fBfork\fR(2),
1323 \fBfpathconf\fR(2), \fBgetacct\fR(2), \fBgetpflags\fR(2), \fBgetppriv\fR(2),
1324 \fBgetsid\fR(2), \fBkill\fR(2), \fBlink\fR(2), \fBmemcntl\fR(2),
1325 \fBmknod\fR(2), \fBmount\fR(2), \fBmsgctl\fR(2), \fBnice\fR(2),
1326 \fBntp_adjtime\fR(2), \fBopen\fR(2), \fBp_online\fR(2), \fBpriocntl\fR(2),
1327 \fBpriocntlset\fR(2), \fBprocessor_bind\fR(2), \fBpset_bind\fR(2),
1328 \fBpset_create\fR(2), \fBreadlink\fR(2), \fBresolvepath\fR(2), \fBrmdir\fR(2),
1329 \fBsemctl\fR(2), \fBsetauid\fR(2), \fBsetegid\fR(2), \fBseteuid\fR(2),
1330 \fBsetgid\fR(2), \fBsetgroups\fR(2), \fBsetpflags\fR(2), \fBsetppriv\fR(2),
1331 \fBsetrctl\fR(2), \fBsetregid\fR(2), \fBsetreuid\fR(2), \fBsetrlimit\fR(2),
1332 \fBsettaskid\fR(2), \fBsetuid\fR(2), \fBshmctl\fR(2), \fBshmget\fR(2),
1333 \fBshmop\fR(2), \fBsigsend\fR(2), \fBstat\fR(2), \fBstatvfs\fR(2),
1334 \fBstime\fR(2), \fBswapctl\fR(2), \fBsysinfo\fR(2), \fBuadmin\fR(2),
1335 \fBulimit\fR(2), \fBumount\fR(2), \fBunlink\fR(2), \fButime\fR(2),
1336 \fButimes\fR(2), \fBbind\fR(3SOCKET), \fBdoor_ucred\fR(3C),
1337 \fBpriv_addset\fR(3C), \fBpriv_set\fR(3C), \fBpriv_getbyname\fR(3C),
1338 \fBpriv_getbynum\fR(3C), \fBpriv_set_to_str\fR(3C), \fBpriv_str_to_set\fR(3C),
1339 \fBsocket\fR(3SOCKET), \fBt_bind\fR(3NSL), \fBtimer_create\fR(3C),
1340 \fBucred_get\fR(3C), \fBexec_attr\fR(4), \fBproc\fR(4), \fBsystem\fR(4),
1341 \fBuser_attr\fR(4), \fBxVM\fR(5), \fBddi_cred\fR(9F), \fBdrv_priv\fR(9F),
1342 \fBpriv_getbyname\fR(9F), \fBpriv_policy\fR(9F), \fBpriv_policy_choice\fR(9F),
1343 \fBpriv_policy_only\fR(9F)
1344 .sp
1345 .LP
1346 \fISystem Administration Guide: Security Services\fR
```

```
**********************************************************
    2901 Wed May 27 19:49:20 2015
new/usr/src/man/man5/security-flags.5
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
.\"
.\" This file and its contents are supplied under the terms of the
.\" Common Development and Distribution License ("CDDL"), version 1.0.
.\" You may only use this file in accordance with the terms of version
.\" 1.0 of the CDDL.
.\"
.\" A full copy of the text of the CDDL should have accompanied this
.\" source.  A copy of the CDDL is also available via the Internet at
.\" http://www.illumos.org/license/CDDL.
.\"
.TH "SECURITY-FLAGS" "5" "May 5, 2014"
.SH "NAME"
\fBsecurity-flags\fR - process security flags
.SH "DESCRIPTION"
Each process on an illumos system has an associated set of security-flags
which describe additional per-process security and exploit mitigation
features which are enabled for that process.
.P
There are two sets of these flags for each process, the effective set
(abbreviated \fIE\fR) are the set which currently apply to the process and are
immutable. The inheritable set (abbreviated \fII\fR) are the flags which will
become effective the next time the process calls one of the \fBexec(2)\fR
family of functions, and will be inherited as both the effective and
inheritable sets by any child processes. The inheritable set may be changed
at any time, subject to permissions.
.P
To change the security-flags of a process one must have both permissions
equivalent to those required to send a signal to the process and have the
\fBPRIV_PROC_SECFLAGS\fR privilege.
.P
Currently available features are:

.sp
.ne 2
.na
Address Space Layout Randomisation (ASLR)
.ad
.RS 11n
The base addresses of the stack, heap and shared library (including
\fBld.so\fR) mappings are randomised, the bases of mapped regions other than
those using \fBMAP_FIXED\fR are randomised.
.P
Currently, executable base addresses are \fInot\fR randomised, due to which
the mitigation provided by this feature is currently limited.
.P
This flag may also be enabled by the presence of the \fBDT_SUNW_ASLR\fR
dynamic tag in the \fB.dynamic\fR section of the executable file. If this
tag has a value of 1, ASLR will be enabled. If the flag has a value of
\fB0\fR ASLR will be disabled. If the tag is not present, the value of the
ASLR flag will be inherited as normal.
.RE

System default security-flags are configured via properties on the
\fBsvc:/system/process-security\fR service, which contains a boolean property
per-flag in the \fBsecflags\fR property group.  For example, to enable ASLR by
default you would execute the following commands:
.sp
.in +2
```



```
.nf
# svccfg -s svc:/system/process-security setprop secflags/aslr = true
.fi
.in -2
.sp
.P
This can be done by any user with the \fBsolaris.smf.value.process-security\fR
authorization.
.P
Since security-flags are strictly inherited, this will not take effect until
the system or zone is next booted.

.SH "SEE ALSO"
.BR psecflags (1),
.BR svccfg (1M),
.BR brk (2),
.BR exec (2),
.BR mmap (2),
.BR mmapobj (2),
.BR privileges (5),
.BR rbac (5)
#endif /* ! codereview */
```

```
**********************************************************
    2985 Wed May 27 19:49:21 2015
new/usr/src/pkg/manifests/SUNWcs.man5.inc
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
```

```
   1 #
   2 # This file and its contents are supplied under the terms of the
   3 # Common Development and Distribution License ("CDDL"), version 1.0.
   4 # You may only use this file in accordance with the terms of version
   5 # 1.0 of the CDDL.
   6 #
   7 # A full copy of the text of the CDDL should have accompanied this
   8 # source.  A copy of the CDDL is also available via the Internet
   9 # at http://www.illumos.org/license/CDDL.
  10 #

  12 #
  13 # Copyright 2011, Richard Lowe
  14 # Copyright 2012 Nexenta Systems, Inc. All rights reserved.
  15 #

  17 file path=usr/share/man/man5/Intro.5
  18 file path=usr/share/man/man5/acl.5
  19 file path=usr/share/man/man5/ad.5
  20 file path=usr/share/man/man5/ascii.5
  21 file path=usr/share/man/man5/attributes.5
  22 file path=usr/share/man/man5/device_clean.5
  23 file path=usr/share/man/man5/dhcp.5
  24 file path=usr/share/man/man5/filesystem.5
  25 file path=usr/share/man/man5/formats.5
  26 file path=usr/share/man/man5/iconv.5
  27 file path=usr/share/man/man5/iconv_unicode.5
  28 file path=usr/share/man/man5/privileges.5
  29 file path=usr/share/man/man5/rbac.5
  30 file path=usr/share/man/man5/resource_controls.5
  31 file path=usr/share/man/man5/security-flags.5
  32 #endif /* ! codereview */
  33 file path=usr/share/man/man5/smf.5
  34 file path=usr/share/man/man5/smf_bootstrap.5
  35 file path=usr/share/man/man5/smf_method.5
  36 file path=usr/share/man/man5/smf_restarter.5
  37 file path=usr/share/man/man5/smf_security.5
  38 file path=usr/share/man/man5/smf_template.5
  39 file path=usr/share/man/man5/standards.5
  40 file path=usr/share/man/man5/sticky.5
  41 file path=usr/share/man/man5/term.5
  42 link path=usr/share/man/man5/ANSI.5 target=standards.5
  43 link path=usr/share/man/man5/C++.5 target=standards.5
  44 link path=usr/share/man/man5/C.5 target=standards.5
  45 link path=usr/share/man/man5/CSI.5 target=attributes.5
  46 link path=usr/share/man/man5/ISO.5 target=standards.5
  47 link path=usr/share/man/man5/MT-Level.5 target=attributes.5
  48 link path=usr/share/man/man5/POSIX.1.5 target=standards.5
  49 link path=usr/share/man/man5/POSIX.2.5 target=standards.5
  50 link path=usr/share/man/man5/POSIX.5 target=standards.5
  51 link path=usr/share/man/man5/RBAC.5 target=rbac.5
  52 link path=usr/share/man/man5/SUS.5 target=standards.5
  53 link path=usr/share/man/man5/SUSv2.5 target=standards.5
  54 link path=usr/share/man/man5/SUSv3.5 target=standards.5
  55 link path=usr/share/man/man5/SVID.5 target=standards.5
  56 link path=usr/share/man/man5/SVID3.5 target=standards.5
  57 link path=usr/share/man/man5/XNS.5 target=standards.5
  58 link path=usr/share/man/man5/XNS4.5 target=standards.5
```



```
  59 link path=usr/share/man/man5/XNS5.5 target=standards.5
  60 link path=usr/share/man/man5/XPG.5 target=standards.5
  61 link path=usr/share/man/man5/XPG3.5 target=standards.5
  62 link path=usr/share/man/man5/XPG4.5 target=standards.5
  63 link path=usr/share/man/man5/XPG4v2.5 target=standards.5
  64 link path=usr/share/man/man5/architecture.5 target=attributes.5
  65 link path=usr/share/man/man5/availability.5 target=attributes.5
  66 link path=usr/share/man/man5/intro.5 target=Intro.5
  67 link path=usr/share/man/man5/stability.5 target=attributes.5
  68 link path=usr/share/man/man5/standard.5 target=attributes.5
```

```
**********************************************************
   87865 Wed May 27 19:49:21 2015
new/usr/src/pkg/manifests/SUNWcs.mf
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
```

```
   1 #
   2 # CDDL HEADER START
   3 #
   4 # The contents of this file are subject to the terms of the
   5 # Common Development and Distribution License (the "License").
   6 # You may not use this file except in compliance with the License.
   7 #
   8 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
   9 # or http://www.opensolaris.org/os/licensing.
  10 # See the License for the specific language governing permissions
  11 # and limitations under the License.
  12 #
  13 # When distributing Covered Code, include this CDDL HEADER in each
  14 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  15 # If applicable, add the following below this CDDL HEADER, with the
  16 # fields enclosed by brackets "[]" replaced with your own identifying
  17 # information: Portions Copyright [yyyy] [name of copyright owner]
  18 #
  19 # CDDL HEADER END
  20 #

  22 #
  23 # Copyright (c) 2010, Oracle and/or its affiliates. All rights reserved.
  24 # Copyright 2015 Nexenta Systems, Inc. All rights reserved.
  25 # Copyright (c) 2013 Gary Mills
  26 # Copyright 2014 Garrett D'Amore <garrett@damore.org>
  27 #

  29 <include SUNWcs.man1.inc>
  30 <include SUNWcs.man1m.inc>
  31 <include SUNWcs.man4.inc>
  32 <include SUNWcs.man5.inc>
  33 <include SUNWcs.man7d.inc>
  34 <include SUNWcs.man7fs.inc>
  35 set name=pkg.fmri value=pkg:/SUNWcs@$(PKGVERS)
  36 set name=pkg.description \
  37     value="core software for a specific instruction-set architecture"
  38 set name=pkg.summary value="Core Solaris"
  39 set name=info.classification value=org.opensolaris.category.2008:System/Core
  40 set name=variant.arch value=$(ARCH)
  41 dir path=dev group=sys
  42 dir path=etc group=sys
  43 dir path=etc/cron.d group=sys
  44 dir path=etc/crypto group=sys
  45 dir path=etc/crypto/certs group=sys
  46 dir path=etc/crypto/crls group=sys
  47 dir path=etc/default group=sys
  48 dir path=etc/dev group=sys
  49 dir path=etc/devices group=sys
  50 dir path=etc/dfs group=sys
  51 dir path=etc/dhcp group=sys
  52 dir path=etc/fs group=sys
  53 dir path=etc/fs/dev group=sys
  54 dir path=etc/fs/hsfs group=sys
  55 dir path=etc/fs/ufs group=sys
  56 dir path=etc/ftpd group=sys
  57 dir path=etc/inet group=sys
  58 dir path=etc/init.d group=sys
```



```
  59 dir path=etc/lib group=sys
  60 dir path=etc/logadm.d group=sys
  61 dir path=etc/mail group=mail
  62 dir path=etc/net group=sys
  63 dir path=etc/net/ticlts group=sys
  64 dir path=etc/net/ticots group=sys
  65 dir path=etc/net/ticotsord group=sys
  66 dir path=etc/opt group=sys
  67 dir path=etc/rc0.d group=sys
  68 dir path=etc/rc1.d group=sys
  69 dir path=etc/rc2.d group=sys
  70 dir path=etc/rc3.d group=sys
  71 dir path=etc/rcS.d group=sys
  72 dir path=etc/rpcsec group=sys
  73 dir path=etc/saf
  74 dir path=etc/saf/zsmon group=sys
  75 dir path=etc/sasl group=sys
  76 dir path=etc/security group=sys
  77 dir path=etc/security/audit group=sys
  78 dir path=etc/security/audit/localhost group=sys
  79 dir path=etc/security/auth_attr.d group=sys
  80 dir path=etc/security/dev group=sys
  81 dir path=etc/security/exec_attr.d group=sys
  82 dir path=etc/security/lib group=sys
  83 dir path=etc/security/prof_attr.d group=sys
  84 dir path=etc/skel group=sys
  85 dir path=etc/svc group=sys
  86 dir path=etc/svc/profile group=sys
  87 dir path=etc/svc/profile/site group=sys
  88 dir path=etc/svc/volatile group=sys
  89 dir path=etc/sysevent group=sys
  90 dir path=etc/sysevent/config group=sys
  91 dir path=etc/tm group=sys
  92 dir path=etc/user_attr.d group=sys
  93 dir path=export group=sys
  94 dir path=home group=root mode=0555
  95 dir path=lib
  96 dir path=lib/crypto
  97 dir path=lib/inet
  98 dir path=lib/svc
  99 dir path=lib/svc/bin
 100 dir path=lib/svc/capture
 101 dir path=lib/svc/manifest group=sys
 102 dir path=lib/svc/manifest/application group=sys
 103 dir path=lib/svc/manifest/application/management group=sys
 104 dir path=lib/svc/manifest/application/security group=sys
 105 dir path=lib/svc/manifest/device group=sys
 106 dir path=lib/svc/manifest/milestone group=sys
 107 dir path=lib/svc/manifest/network group=sys
 108 dir path=lib/svc/manifest/network/dns group=sys
 109 dir path=lib/svc/manifest/network/ipsec group=sys
 110 dir path=lib/svc/manifest/network/ldap group=sys
 111 dir path=lib/svc/manifest/network/routing group=sys
 112 dir path=lib/svc/manifest/network/rpc group=sys
 113 dir path=lib/svc/manifest/network/shares group=sys
 114 dir path=lib/svc/manifest/network/ssl group=sys
 115 dir path=lib/svc/manifest/platform group=sys
 116 $(sparc_ONLY)dir path=lib/svc/manifest/platform/sun4u group=sys
 117 dir path=lib/svc/manifest/site group=sys
 118 dir path=lib/svc/manifest/system group=sys
 119 dir path=lib/svc/manifest/system/device group=sys
 120 dir path=lib/svc/manifest/system/filesystem group=sys
 121 dir path=lib/svc/manifest/system/security group=sys
 122 dir path=lib/svc/manifest/system/svc group=sys
 123 dir path=lib/svc/method
 124 dir path=lib/svc/monitor
```

```
125 dir path=lib/svc/seed
126 dir path=lib/svc/share
127 dir path=mnt group=sys
128 dir path=opt group=sys
129 dir path=proc group=root mode=0555
130 dir path=root group=root mode=0700
131 dir path=sbin group=sys
132 dir path=system group=root
133 dir path=system/contract group=root mode=0555
134 dir path=system/object group=root mode=0555
135 dir path=tmp group=sys mode=1777
136 dir path=usr group=sys
137 dir path=usr/bin
138 dir path=usr/bin/$(ARCH32)
139 dir path=usr/bin/$(ARCH64)
140 dir path=usr/ccs
141 dir path=usr/ccs/bin
142 dir path=usr/demo
143 dir path=usr/games
144 dir path=usr/has
145 dir path=usr/has/bin
146 dir path=usr/has/lib
147 dir path=usr/has/man
148 dir path=usr/has/man/man1has
149 dir path=usr/kernel group=sys
150 dir path=usr/kernel/drv group=sys
151 dir path=usr/kernel/drv/$(ARCH64) group=sys
152 dir path=usr/kernel/exec group=sys
153 dir path=usr/kernel/exec/$(ARCH64) group=sys
154 dir path=usr/kernel/fs group=sys
155 dir path=usr/kernel/fs/$(ARCH64) group=sys
156 dir path=usr/kernel/pcbe group=sys
157 dir path=usr/kernel/pcbe/$(ARCH64) group=sys
158 dir path=usr/kernel/sched group=sys
159 dir path=usr/kernel/sched/$(ARCH64) group=sys
160 dir path=usr/kernel/strmod group=sys
161 dir path=usr/kernel/strmod/$(ARCH64) group=sys
162 dir path=usr/kernel/sys group=sys
163 dir path=usr/kernel/sys/$(ARCH64) group=sys
164 dir path=usr/kvm
165 dir path=usr/lib
166 dir path=usr/lib/$(ARCH64)
167 dir path=usr/lib/audit
168 dir path=usr/lib/class
169 dir path=usr/lib/class/FX
170 dir path=usr/lib/class/IA
171 dir path=usr/lib/class/RT
172 dir path=usr/lib/class/SDC
173 dir path=usr/lib/class/TS
174 dir path=usr/lib/crypto
175 dir path=usr/lib/devfsadm group=sys
176 dir path=usr/lib/devfsadm/linkmod group=sys
177 dir path=usr/lib/fs group=sys
178 dir path=usr/lib/fs/autofs group=sys
179 dir path=usr/lib/fs/autofs/$(ARCH64) group=sys
180 dir path=usr/lib/fs/cachefs group=sys
181 dir path=usr/lib/fs/ctfs group=sys
182 dir path=usr/lib/fs/dev group=sys
183 dir path=usr/lib/fs/fd group=sys
184 dir path=usr/lib/fs/hsfs group=sys
185 dir path=usr/lib/fs/lofs group=sys
186 dir path=usr/lib/fs/mntfs group=sys
187 dir path=usr/lib/fs/nfs group=sys
188 dir path=usr/lib/fs/nfs/$(ARCH64) group=sys
189 dir path=usr/lib/fs/objfs group=sys
190 dir path=usr/lib/fs/proc group=sys
```



```
191 dir path=usr/lib/fs/sharefs group=sys
192 dir path=usr/lib/fs/tmpfs group=sys
193 dir path=usr/lib/fs/ufs group=sys
194 dir path=usr/lib/help
195 dir path=usr/lib/help/auths
196 dir path=usr/lib/help/auths/locale
197 dir path=usr/lib/help/auths/locale/C
198 dir path=usr/lib/help/profiles
199 dir path=usr/lib/help/profiles/locale
200 dir path=usr/lib/help/profiles/locale/C
201 dir path=usr/lib/iconv
202 dir path=usr/lib/inet
203 dir path=usr/lib/inet/$(ARCH32)
204 dir path=usr/lib/inet/$(ARCH64)
205 dir path=usr/lib/locale
206 dir path=usr/lib/locale/C
207 dir path=usr/lib/locale/C/LC_COLLATE
208 dir path=usr/lib/locale/C/LC_CTYPE
209 dir path=usr/lib/locale/C/LC_MESSAGES
210 dir path=usr/lib/locale/C/LC_MONETARY
211 dir path=usr/lib/locale/C/LC_NUMERIC
212 dir path=usr/lib/locale/C/LC_TIME
213 dir path=usr/lib/netsvc group=sys
214 dir path=usr/lib/pci
215 dir path=usr/lib/rcm
216 dir path=usr/lib/rcm/modules
217 dir path=usr/lib/rcm/scripts
218 dir path=usr/lib/reparse
219 dir path=usr/lib/saf
220 dir path=usr/lib/secure
221 dir path=usr/lib/secure/$(ARCH64)
222 dir path=usr/lib/security
223 dir path=usr/lib/sysevent
224 dir path=usr/lib/sysevent/modules
225 dir path=usr/net group=sys
226 dir path=usr/net/nls group=sys
227 dir path=usr/net/servers group=sys
228 dir path=usr/old
229 dir path=usr/platform group=sys
230 dir path=usr/sadm
231 dir path=usr/sadm/bin
232 dir path=usr/sadm/install
233 dir path=usr/sadm/install/scripts
234 dir path=usr/sbin
235 $(i386_ONLY)dir path=usr/sbin/$(ARCH32)
236 dir path=usr/sbin/$(ARCH64)
237 dir path=usr/share
238 dir path=usr/share/doc group=other
239 dir path=usr/share/doc/ksh
240 dir path=usr/share/doc/ksh/images
241 dir path=usr/share/doc/ksh/images/callouts
242 dir path=usr/share/lib
243 dir path=usr/share/lib/mailx
244 dir path=usr/share/lib/pub
245 dir path=usr/share/lib/tabset
246 dir path=usr/share/lib/xml group=sys
247 dir path=usr/share/lib/xml/dtd group=sys
248 dir path=usr/share/lib/xml/style group=sys
249 dir path=usr/share/man
250 dir path=usr/share/man/man1
251 dir path=usr/share/man/man1m
252 dir path=usr/share/man/man4
253 dir path=usr/share/man/man5
254 dir path=usr/share/man/man7d
255 dir path=usr/share/man/man7fs
256 dir path=usr/share/src group=sys
```

```
dir path=var group=sys
dir path=var/adm group=sys mode=0775
dir path=var/adm/exacct group=adm owner=adm
dir path=var/adm/log group=adm owner=adm
dir path=var/adm/streams group=sys
dir path=var/audit group=sys
dir path=var/cores group=sys
dir path=var/cron group=sys
dir path=var/games
dir path=var/idmap group=daemon owner=daemon
dir path=var/inet group=sys
dir path=var/ld
dir path=var/ld/$(ARCH64)
dir path=var/log group=sys
dir path=var/logadm
dir path=var/mail group=mail mode=1777
dir path=var/mail/:saved group=mail mode=0775
dir path=var/news
dir path=var/opt group=sys
dir path=var/preserve mode=1777
dir path=var/run group=sys
dir path=var/saf
dir path=var/saf/zsmon group=sys
dir path=var/spool
dir path=var/spool/cron group=sys
dir path=var/spool/cron/atjobs group=sys
dir path=var/spool/cron/crontabs group=sys
dir path=var/spool/locks group=uucp owner=uucp
dir path=var/svc group=sys
dir path=var/svc/log group=sys
dir path=var/svc/manifest group=sys
dir path=var/svc/manifest/application group=sys
dir path=var/svc/manifest/application/management group=sys
dir path=var/svc/manifest/application/print group=sys
dir path=var/svc/manifest/application/security group=sys
dir path=var/svc/manifest/device group=sys
dir path=var/svc/manifest/milestone group=sys
dir path=var/svc/manifest/network group=sys
dir path=var/svc/manifest/network/dns group=sys
dir path=var/svc/manifest/network/ipsec group=sys
dir path=var/svc/manifest/network/ldap group=sys
dir path=var/svc/manifest/network/nfs group=sys
dir path=var/svc/manifest/network/nis group=sys
dir path=var/svc/manifest/network/routing group=sys
dir path=var/svc/manifest/network/rpc group=sys
dir path=var/svc/manifest/network/security group=sys
dir path=var/svc/manifest/network/shares group=sys
dir path=var/svc/manifest/network/ssl group=sys
dir path=var/svc/manifest/platform group=sys
$(sparc_ONLY)dir path=var/svc/manifest/platform/sun4u group=sys
$(sparc_ONLY)dir path=var/svc/manifest/platform/sun4v group=sys
dir path=var/svc/manifest/site group=sys
dir path=var/svc/manifest/system group=sys
dir path=var/svc/manifest/system/device group=sys
dir path=var/svc/manifest/system/filesystem group=sys
dir path=var/svc/manifest/system/security group=sys
dir path=var/svc/manifest/system/svc group=sys
dir path=var/svc/profile group=sys
dir path=var/tmp group=sys mode=1777
driver name=dump perms="dump 0660 root sys"
driver name=fssnap \
    policy="ctl read_priv_set=sys_config write_priv_set=sys_config" \
    perms="* 0640 root sys" perms="ctl 0666 root sys"
driver name=kstat perms="* 0666 root sys"
driver name=ksyms perms="* 0666 root sys"
driver name=logindmux
```



```
driver name=ptm clone_perms="ptmx 0666 root sys"
driver name=pts perms="* 0644 root sys" perms="0 0620 root tty" \
    perms="1 0620 root tty" perms="2 0620 root tty" perms="3 0620 root tty"
file path=etc/.login group=sys preserve=renamenew
file path=etc/cron.d/.proto group=sys mode=0744
file path=etc/cron.d/at.deny group=sys preserve=true
file path=etc/cron.d/cron.deny group=sys preserve=true
file path=etc/cron.d/queuedefs group=sys
file path=etc/crypto/kmf.conf group=sys preserve=true
file path=etc/crypto/pkcs11.conf group=sys preserve=true
file path=etc/datemsk group=sys mode=0444
file path=etc/default/cron group=sys preserve=true
file path=etc/default/devfsadm group=sys preserve=true
file path=etc/default/fs group=sys preserve=true
file path=etc/default/init group=sys preserve=true
file path=etc/default/keyserv group=sys preserve=true
file path=etc/default/login group=sys preserve=true
file path=etc/default/nss group=sys preserve=true
file path=etc/default/passwd group=sys preserve=true
file path=etc/default/su group=sys preserve=true
file path=etc/default/syslogd group=sys preserve=true
file path=etc/default/tar group=sys preserve=true
file path=etc/default/useradd group=sys preserve=true
file path=etc/default/utmpd group=sys preserve=true
file path=etc/dev/reserved_devnames group=sys preserve=true
file path=etc/device.tab group=root mode=0444 preserve=true
file path=etc/dfs/dfstab group=sys preserve=true
file path=etc/dfs/fstypes group=root preserve=true
file path=etc/dfs/sharetab group=root mode=0444 preserve=true
file path=etc/dgroup.tab group=sys mode=0444 preserve=true
file path=etc/dhcp/inittab group=sys preserve=true
file path=etc/dhcp/inittab6 group=sys preserve=true
file path=etc/dumpdates group=sys mode=0664 preserve=true
file path=etc/format.dat group=sys preserve=true
file path=etc/fs/dev/mount mode=0555
file path=etc/fs/hsfs/mount mode=0555
file path=etc/fs/ufs/mount mode=0555
file path=etc/ftpd/ftpusers group=sys preserve=true
file path=etc/group group=sys preserve=true
file path=etc/inet/hosts group=sys preserve=true
file path=etc/inet/inetd.conf group=sys preserve=true
file path=etc/inet/ipaddrsel.conf group=sys preserve=true
file path=etc/inet/netmasks group=sys preserve=true
file path=etc/inet/networks group=sys preserve=true
file path=etc/inet/protocols group=sys preserve=true
file path=etc/inet/services group=sys preserve=true
file path=etc/inet/wanboot.conf.sample group=sys mode=0444
file path=etc/init.d/PRESERVE group=sys mode=0744 preserve=true
file path=etc/init.d/README group=sys preserve=true
file path=etc/init.d/cachefs.daemon group=sys mode=0744 preserve=true
file path=etc/init.d/sysetup group=sys mode=0744 preserve=true
file path=etc/inittab group=sys preserve=true
file path=etc/ioctl.syscon group=sys preserve=true
file path=etc/ksh.kshrc group=sys preserve=renameold
file path=etc/logadm.conf group=sys preserve=true timestamp=19700101T000000Z
file path=etc/logindevperm group=sys preserve=true
file path=etc/magic mode=0444
file path=etc/mail/mailx.rc preserve=true
file path=etc/mailcap preserve=true
file path=etc/mime.types preserve=true
file path=etc/mnttab group=root mode=0444 preserve=true
file path=etc/motd group=sys preserve=true
file path=etc/net/ticlts/hosts group=sys
file path=etc/net/ticlts/services group=sys preserve=true
file path=etc/net/ticots/hosts group=sys
file path=etc/net/ticots/services group=sys preserve=true
```

```
389 file path=etc/net/ticotsord/hosts group=sys
390 file path=etc/net/ticotsord/services group=sys preserve=true
391 file path=etc/netconfig group=sys preserve=true
392 file path=etc/nscd.conf group=sys preserve=true
393 file path=etc/nsswitch.ad group=sys
394 file path=etc/nsswitch.conf group=sys preserve=true
395 file path=etc/nsswitch.dns group=sys
396 file path=etc/nsswitch.files group=sys
397 file path=etc/nsswitch.ldap group=sys
398 file path=etc/pam.conf group=sys preserve=true
399 file path=etc/passwd group=sys preserve=true
400 file path=etc/profile group=sys preserve=renamenew
401 file path=etc/project group=sys preserve=true
402 file path=etc/rc2.d/README group=sys
403 file path=etc/rc3.d/README group=sys
404 file path=etc/rcS.d/README group=sys
405 file path=etc/remote preserve=true
406 file path=etc/rpc group=sys preserve=true
407 file path=etc/saf/_sactab group=sys preserve=true
408 file path=etc/saf/_sysconfig group=sys preserve=true
409 file path=etc/saf/zsmon/_pmtab group=sys preserve=true
410 file path=etc/security/audit_class group=sys preserve=renamenew
411 file path=etc/security/audit_event group=sys preserve=renamenew
412 file path=etc/security/audit_warn group=sys mode=0740 preserve=renamenew
413 file path=etc/security/auth_attr group=sys preserve=true \
414     timestamp=19700101T000000Z
415 file path=etc/security/auth_attr.d/SUNWcs group=sys
416 file path=etc/security/crypt.conf group=sys preserve=renamenew
417 file path=etc/security/dev/audio mode=0400
418 file path=etc/security/dev/fd0 mode=0400
419 file path=etc/security/dev/sr0 mode=0400
420 file path=etc/security/dev/st0 mode=0400
421 file path=etc/security/dev/st1 mode=0400
422 file path=etc/security/exec_attr group=sys preserve=true \
423     timestamp=19700101T000000Z
424 file path=etc/security/exec_attr.d/SUNWcs group=sys
425 file path=etc/security/kmfpolicy.xml
426 file path=etc/security/lib/audio_clean group=sys mode=0555
427 file path=etc/security/lib/fd_clean group=sys mode=0555
428 file path=etc/security/lib/sr_clean group=sys mode=0555
429 file path=etc/security/lib/st_clean group=sys mode=0555
430 file path=etc/security/policy.conf group=sys preserve=true
431 file path=etc/security/priv_names group=sys preserve=renameold
432 file path=etc/security/prof_attr group=sys preserve=true \
433     timestamp=19700101T000000Z
434 file path=etc/security/prof_attr.d/SUNWcs group=sys
435 file path=etc/shadow group=sys mode=0400 preserve=true
436 file path=etc/skel/.profile group=other preserve=true
437 file path=etc/skel/local.cshrc group=sys preserve=true
438 file path=etc/skel/local.login group=sys preserve=true
439 file path=etc/skel/local.profile group=sys preserve=true
440 file path=etc/svc/profile/generic_limited_net.xml group=sys mode=0444
441 file path=etc/svc/profile/generic_open.xml group=sys mode=0444
442 file path=etc/svc/profile/inetd_generic.xml group=sys mode=0444
443 file path=etc/svc/profile/inetd_upgrade.xml group=sys mode=0444
444 file path=etc/svc/profile/ns_dns.xml group=sys mode=0444
445 file path=etc/svc/profile/ns_files.xml group=sys mode=0444
446 file path=etc/svc/profile/ns_ldap.xml group=sys mode=0444
447 file path=etc/svc/profile/ns_nis.xml group=sys mode=0444
448 file path=etc/svc/profile/ns_none.xml group=sys mode=0444
449 $(sparc_ONLY)file path=etc/svc/profile/platform_SUNW,SPARC-Enterprise.xml \
450     group=sys mode=0444
451 $(sparc_ONLY)file path=etc/svc/profile/platform_SUNW,Sun-Fire-15000.xml \
452     group=sys mode=0444
453 $(sparc_ONLY)file path=etc/svc/profile/platform_SUNW,Sun-Fire-880.xml \
454     group=sys mode=0444
```



```
455 $(sparc_ONLY)file path=etc/svc/profile/platform_SUNW,Sun-Fire.xml group=sys \
456     mode=0444
457 $(sparc_ONLY)file \
458     path=etc/svc/profile/platform_SUNW,Ultra-Enterprise-10000.xml group=sys \
459     mode=0444
460 $(sparc_ONLY)file \
461     path=etc/svc/profile/platform_SUNW,UltraSPARC-IIi-Netract.xml group=sys \
462     mode=0444
463 file path=etc/svc/profile/platform_none.xml group=sys mode=0444
464 $(sparc_ONLY)file path=etc/svc/profile/platform_sun4v.xml group=sys mode=0444
465 file path=etc/sysevent/config/README group=sys mode=0444
466 file path=etc/sysevent/config/SUNW,EC_dr,ESC_dr_req,sysevent.conf group=sys
467 file path=etc/syslog.conf group=sys preserve=true
468 file path=etc/ttydefs group=sys preserve=true
469 file path=etc/ttysrch group=sys preserve=true
470 file path=etc/user_attr group=sys preserve=true timestamp=19700101T000000Z
471 file path=etc/user_attr.d/SUNWcs group=sys
472 file path=etc/vfstab group=sys preserve=true
473 file path=lib/inet/in.mpathd mode=0555
474 file path=lib/inet/ipmgmtd mode=0555
475 file path=lib/inet/netcfgd mode=0555
476 file path=lib/inet/nwamd mode=0555
477 file path=lib/svc/bin/lsvcrun group=sys mode=0555
478 file path=lib/svc/bin/mfstscan group=sys mode=0555
479 file path=lib/svc/bin/restore_repository group=sys mode=0555
480 file path=lib/svc/bin/sqlite group=sys mode=0555
481 file path=lib/svc/bin/svc.configd group=sys mode=0555
482 file path=lib/svc/bin/svc.ipfd group=sys mode=0555
483 file path=lib/svc/bin/svc.startd group=sys mode=0555
484 file path=lib/svc/manifest/milestone/multi-user-server.xml group=sys mode=0444
485 file path=lib/svc/manifest/milestone/multi-user.xml group=sys mode=0444
486 file path=lib/svc/manifest/milestone/name-services.xml group=sys mode=0444
487 file path=lib/svc/manifest/milestone/network.xml group=sys mode=0444
488 file path=lib/svc/manifest/milestone/single-user.xml group=sys mode=0444
489 file path=lib/svc/manifest/milestone/sysconfig.xml group=sys mode=0444
490 file path=lib/svc/manifest/network/dlmgmt.xml group=sys mode=0444
491 file path=lib/svc/manifest/network/dns/client.xml group=sys mode=0444
492 file path=lib/svc/manifest/network/dns/install.xml group=sys mode=0444
493 file path=lib/svc/manifest/network/forwarding.xml group=sys mode=0444
494 file path=lib/svc/manifest/network/inetd-upgrade.xml group=sys mode=0444
495 file path=lib/svc/manifest/network/inetd.xml group=sys mode=0444
496 file path=lib/svc/manifest/network/ipsec/ike.xml group=sys mode=0444
497 file path=lib/svc/manifest/network/ipsec/ipsecalgs.xml group=sys mode=0444
498 file path=lib/svc/manifest/network/ipsec/manual-key.xml group=sys mode=0444
499 file path=lib/svc/manifest/network/ipsec/policy.xml group=sys mode=0444
500 file path=lib/svc/manifest/network/ldap/client.xml group=sys mode=0444
501 file path=lib/svc/manifest/network/network-initial.xml group=sys mode=0444
502 file path=lib/svc/manifest/network/network-install.xml group=sys mode=0444
503 file path=lib/svc/manifest/network/network-ipmgmt.xml group=sys mode=0444
504 file path=lib/svc/manifest/network/network-ipqos.xml group=sys mode=0444
505 file path=lib/svc/manifest/network/network-iptun.xml group=sys mode=0444
506 file path=lib/svc/manifest/network/network-location.xml group=sys mode=0444
507 file path=lib/svc/manifest/network/network-loopback.xml group=sys mode=0444
508 file path=lib/svc/manifest/network/network-netcfg.xml group=sys mode=0444
509 file path=lib/svc/manifest/network/network-netmask.xml group=sys mode=0444
510 file path=lib/svc/manifest/network/network-physical.xml group=sys mode=0444
511 file path=lib/svc/manifest/network/network-routing-setup.xml group=sys \
512     mode=0444
513 file path=lib/svc/manifest/network/network-service.xml group=sys mode=0444
514 file path=lib/svc/manifest/network/routing/legacy-routing.xml group=sys \
515     mode=0444
516 file path=lib/svc/manifest/network/rpc/bind.xml group=sys mode=0444
517 file path=lib/svc/manifest/network/rpc/keyserv.xml group=sys mode=0444
518 file path=lib/svc/manifest/network/shares/group.xml group=sys mode=0444
519 file path=lib/svc/manifest/network/shares/reparsed.xml group=sys mode=0444
520 file path=lib/svc/manifest/network/socket-filter-kssl.xml group=sys mode=0444
```

```
file path=lib/svc/manifest/network/ssl/kssl-proxy.xml group=sys mode=0444
file path=lib/svc/manifest/system/auditd.xml group=sys mode=0444
file path=lib/svc/manifest/system/auditset.xml group=sys mode=0444
file path=lib/svc/manifest/system/boot-archive-update.xml group=sys mode=0444
file path=lib/svc/manifest/system/boot-archive.xml group=sys mode=0444
file path=lib/svc/manifest/system/boot-config.xml group=sys mode=0444
file path=lib/svc/manifest/system/consadm.xml group=sys mode=0444
file path=lib/svc/manifest/system/console-login.xml group=sys mode=0444
file path=lib/svc/manifest/system/coreadm.xml group=sys mode=0444
file path=lib/svc/manifest/system/cron.xml group=sys mode=0444
file path=lib/svc/manifest/system/cryptosvc.xml group=sys mode=0444
file path=lib/svc/manifest/system/device/allocate.xml group=sys mode=0444
file path=lib/svc/manifest/system/device/devices-audio.xml group=sys mode=0444
file path=lib/svc/manifest/system/device/devices-local.xml group=sys mode=0444
file path=lib/svc/manifest/system/device/mpxio-upgrade.xml group=sys mode=0444
file path=lib/svc/manifest/system/early-manifest-import.xml group=sys \
    mode=0444
file path=lib/svc/manifest/system/extended-accounting.xml group=sys mode=0444
file path=lib/svc/manifest/system/filesystem/local-fs.xml group=sys mode=0444
file path=lib/svc/manifest/system/filesystem/minimal-fs.xml group=sys \
    mode=0444
file path=lib/svc/manifest/system/filesystem/root-fs.xml group=sys mode=0444
file path=lib/svc/manifest/system/filesystem/usr-fs.xml group=sys mode=0444
$(i386_ONLY)file path=lib/svc/manifest/system/hostid.xml group=sys mode=0444
file path=lib/svc/manifest/system/hotplug.xml group=sys mode=0444
file path=lib/svc/manifest/system/identity.xml group=sys mode=0444
file path=lib/svc/manifest/system/idmap.xml group=sys mode=0444
file path=lib/svc/manifest/system/keymap.xml group=sys mode=0444
file path=lib/svc/manifest/system/logadm-upgrade.xml group=sys mode=0444
file path=lib/svc/manifest/system/manifest-import.xml group=sys mode=0444
file path=lib/svc/manifest/system/name-service-cache.xml group=sys mode=0444
file path=lib/svc/manifest/system/pfexecd.xml group=sys mode=0444
file path=lib/svc/manifest/system/process-security.xml group=sys mode=0444
#endif /* ! codereview */
file path=lib/svc/manifest/system/rbac.xml group=sys mode=0444
file path=lib/svc/manifest/system/rmtmpfiles.xml group=sys mode=0444
file path=lib/svc/manifest/system/sac.xml group=sys mode=0444
file path=lib/svc/manifest/system/svc/global.xml group=sys mode=0444
file path=lib/svc/manifest/system/svc/restarter.xml group=sys mode=0444
file path=lib/svc/manifest/system/system-log.xml group=sys mode=0444
file path=lib/svc/manifest/system/utmp.xml group=sys mode=0444
file path=lib/svc/manifest/system/vtdaemon.xml group=sys mode=0444
file path=lib/svc/method/boot-archive mode=0555
file path=lib/svc/method/boot-archive-update mode=0555
file path=lib/svc/method/console-login mode=0555
file path=lib/svc/method/devices-audio mode=0555
file path=lib/svc/method/devices-local mode=0555
file path=lib/svc/method/dns-install mode=0555
file path=lib/svc/method/fs-local mode=0555
file path=lib/svc/method/fs-minimal mode=0555
file path=lib/svc/method/fs-root mode=0555
file path=lib/svc/method/fs-usr mode=0555
file path=lib/svc/method/identity-domain mode=0555
file path=lib/svc/method/identity-node mode=0555
file path=lib/svc/method/inetd-upgrade mode=0555
file path=lib/svc/method/keymap mode=0555
file path=lib/svc/method/ldap-client mode=0555
file path=lib/svc/method/logadm-upgrade mode=0555
file path=lib/svc/method/manifest-import mode=0555
file path=lib/svc/method/mpxio-upgrade mode=0555
file path=lib/svc/method/net-init mode=0555
file path=lib/svc/method/net-install mode=0555
file path=lib/svc/method/net-ipmgmt mode=0555
file path=lib/svc/method/net-ipqos mode=0555
file path=lib/svc/method/net-iptun mode=0555
file path=lib/svc/method/net-loc mode=0555
```



```
file path=lib/svc/method/net-loopback mode=0555
file path=lib/svc/method/net-netmask mode=0555
file path=lib/svc/method/net-nwam mode=0555
file path=lib/svc/method/net-physical mode=0555
file path=lib/svc/method/net-routing-setup mode=0555
file path=lib/svc/method/net-svc mode=0555
file path=lib/svc/method/rmtmpfiles mode=0555
file path=lib/svc/method/rpc-bind mode=0555
file path=lib/svc/method/svc-allocate mode=0555
file path=lib/svc/method/svc-auditd mode=0555
file path=lib/svc/method/svc-auditset mode=0555
file path=lib/svc/method/svc-boot-config mode=0555
file path=lib/svc/method/svc-consadm mode=0555
file path=lib/svc/method/svc-cron mode=0555
file path=lib/svc/method/svc-dlmgmtd mode=0555
file path=lib/svc/method/svc-forwarding mode=0555
$(i386_ONLY)file path=lib/svc/method/svc-hostid mode=0555
file path=lib/svc/method/svc-hotplug mode=0555
file path=lib/svc/method/svc-legacy-routing mode=0555
file path=lib/svc/method/svc-nscd mode=0555
file path=lib/svc/method/svc-rbac mode=0555
file path=lib/svc/method/svc-sockfilter mode=0555
file path=lib/svc/method/svc-utmpd mode=0555
file path=lib/svc/method/system-log mode=0555
file path=lib/svc/method/vtdaemon mode=0555
file path=lib/svc/method/yp mode=0555
# global.db is not needed in non-global zones, and it's pretty large.
file path=lib/svc/seed/global.db group=sys mode=0444 \
    variant.opensolaris.zone=global
# symmetrically, nonglobal.db is not needed in global zones.
file path=lib/svc/seed/nonglobal.db group=sys mode=0444 \
    variant.opensolaris.zone=nonglobal
file path=lib/svc/share/README mode=0444
file path=lib/svc/share/fs_include.sh mode=0444
file path=lib/svc/share/ipf_include.sh mode=0444
file path=lib/svc/share/mfsthistory mode=0444
file path=lib/svc/share/net_include.sh mode=0444
file path=lib/svc/share/routing_include.sh mode=0444
file path=lib/svc/share/smf_include.sh mode=0444
file path=root/.bashrc group=root preserve=true
file path=root/.profile group=root preserve=true
file path=sbin/autopush mode=0555
$(i386_ONLY)file path=sbin/biosdev mode=0555
file path=sbin/bootadm mode=0555
file path=sbin/cryptoadm mode=0555
file path=sbin/devprop mode=0555
file path=sbin/dhcpagent mode=0555
file path=sbin/dhcpinfo mode=0555
file path=sbin/dlmgmtd mode=0555
file path=sbin/fdisk mode=0555
file path=sbin/fiocompress mode=0555
file path=sbin/hostconfig mode=0555
file path=sbin/ifconfig mode=0555
file path=sbin/ifparse mode=0555
file path=sbin/init group=sys mode=0555
$(i386_ONLY)file path=sbin/installgrub group=sys mode=0555
file path=sbin/ipmpstat mode=0555
file path=sbin/mount mode=0555
file path=sbin/mountall group=sys mode=0555
file path=sbin/netstrategy mode=0555
file path=sbin/rc0 group=sys mode=0744
file path=sbin/rc1 group=sys mode=0744
file path=sbin/rc2 group=sys mode=0744
file path=sbin/rc3 group=sys mode=0744
file path=sbin/rcS group=sys mode=0744
file path=sbin/route mode=0555
```

```
653 file path=sbin/routeadm mode=0555
654 file path=sbin/soconfig mode=0555
655 file path=sbin/su.static group=sys mode=0555
656 file path=sbin/sulogin mode=0555
657 file path=sbin/swapadd group=sys mode=0744
658 file path=sbin/sync mode=0555
659 file path=sbin/tzreload mode=0555
660 file path=sbin/uadmin group=sys mode=0555
661 file path=sbin/umount mode=0555
662 file path=sbin/umountall group=sys mode=0555
663 file path=sbin/uname mode=0555
664 file path=sbin/zonename mode=0555
665 $(i386_ONLY)file path=usr/bin/$(ARCH32)/amt mode=0555
666 file path=usr/bin/$(ARCH32)/decrypt mode=0555
667 file path=usr/bin/$(ARCH32)/digest mode=0555
668 file path=usr/bin/$(ARCH32)/ksh93 mode=0555
669 $(i386_ONLY)file path=usr/bin/$(ARCH32)/newtask group=sys mode=4555
670 $(i386_ONLY)file path=usr/bin/$(ARCH32)/nohup mode=0555
671 $(i386_ONLY)file path=usr/bin/$(ARCH32)/prctl mode=0555
672 $(i386_ONLY)file path=usr/bin/$(ARCH32)/prstat mode=0555
673 $(i386_ONLY)file path=usr/bin/$(ARCH32)/ps mode=0555
674 file path=usr/bin/$(ARCH32)/savecore mode=0555
675 $(i386_ONLY)file path=usr/bin/$(ARCH32)/setuname mode=0555
676 $(i386_ONLY)file path=usr/bin/$(ARCH32)/uptime mode=4555
677 file path=usr/bin/$(ARCH64)/amt mode=0555
678 file path=usr/bin/$(ARCH64)/crle mode=0555
679 file path=usr/bin/$(ARCH64)/decrypt mode=0555
680 file path=usr/bin/$(ARCH64)/digest mode=0555
681 file path=usr/bin/$(ARCH64)/ksh93 mode=0555
682 file path=usr/bin/$(ARCH64)/ls mode=0555
683 file path=usr/bin/$(ARCH64)/moe mode=0555
684 file path=usr/bin/$(ARCH64)/newtask group=sys mode=4555
685 file path=usr/bin/$(ARCH64)/nohup mode=0555
686 file path=usr/bin/$(ARCH64)/prctl mode=0555
687 file path=usr/bin/$(ARCH64)/prstat mode=0555
688 file path=usr/bin/$(ARCH64)/ps mode=0555
689 file path=usr/bin/$(ARCH64)/savecore mode=0555
690 file path=usr/bin/$(ARCH64)/setuname mode=0555
691 file path=usr/bin/$(ARCH64)/uptime mode=4555
692 $(i386_ONLY)file path=usr/bin/addbadsec mode=0555
693 file path=usr/bin/alias mode=0555
694 file path=usr/bin/amt mode=0555
695 file path=usr/bin/arch mode=0555
696 file path=usr/bin/at group=sys mode=4755
697 file path=usr/bin/atq group=sys mode=4755
698 file path=usr/bin/atrm group=sys mode=4755
699 file path=usr/bin/auths mode=0555
700 file path=usr/bin/basename mode=0555
701 file path=usr/bin/busstat mode=0555
702 file path=usr/bin/captoinfo mode=0555
703 file path=usr/bin/cat mode=0555
704 file path=usr/bin/chgrp mode=0555
705 file path=usr/bin/chmod mode=0555
706 file path=usr/bin/chown mode=0555
707 file path=usr/bin/ckdate mode=0555
708 file path=usr/bin/ckgid mode=0555
709 file path=usr/bin/ckint mode=0555
710 file path=usr/bin/ckitem mode=0555
711 file path=usr/bin/ckkeywd mode=0555
712 file path=usr/bin/ckpath mode=0555
713 file path=usr/bin/ckrange mode=0555
714 file path=usr/bin/ckstr mode=0555
715 file path=usr/bin/cktime mode=0555
716 file path=usr/bin/ckuid mode=0555
717 file path=usr/bin/ckyorn mode=0555
718 file path=usr/bin/clear mode=0555
```



```
719 file path=usr/bin/coreadm mode=0555
720 file path=usr/bin/cp mode=0555
721 file path=usr/bin/cpio mode=0555
722 file path=usr/bin/crle mode=0555
723 file path=usr/bin/crontab mode=4555
724 file path=usr/bin/crypt mode=0555
725 file path=usr/bin/csh mode=0555
726 file path=usr/bin/ctrun mode=0555
727 file path=usr/bin/ctstat mode=0555
728 file path=usr/bin/ctwatch mode=0555
729 file path=usr/bin/date mode=0555
730 file path=usr/bin/dd mode=0555
731 file path=usr/bin/devattr mode=0555
732 file path=usr/bin/devfree mode=0555
733 file path=usr/bin/devreserv mode=0555
734 file path=usr/bin/dirname mode=0555
735 $(i386_ONLY)file path=usr/bin/diskscan mode=0555
736 file path=usr/bin/domainname mode=0555
737 file path=usr/bin/du mode=0555
738 file path=usr/bin/dumpcs mode=0555
739 file path=usr/bin/dumpkeys mode=0555
740 file path=usr/bin/echo mode=0555
741 file path=usr/bin/ed mode=0555
742 file path=usr/bin/egrep mode=0555
743 file path=usr/bin/eject mode=0555
744 file path=usr/bin/env mode=0555
745 file path=usr/bin/expr mode=0555
746 file path=usr/bin/false mode=0555
747 file path=usr/bin/fdetach mode=0555
748 file path=usr/bin/fdformat mode=4555
749 file path=usr/bin/fgrep mode=0555
750 file path=usr/bin/file mode=0555
751 file path=usr/bin/find mode=0555
752 file path=usr/bin/fmt mode=0555
753 file path=usr/bin/fmtmsg mode=0555
754 file path=usr/bin/fold mode=0555
755 file path=usr/bin/fsstat mode=0555
756 file path=usr/bin/geniconvtbl mode=0555
757 file path=usr/bin/getconf mode=0555
758 file path=usr/bin/getdev mode=0555
759 file path=usr/bin/getdgrp mode=0555
760 file path=usr/bin/getent mode=0555
761 file path=usr/bin/getfacl mode=0555
762 file path=usr/bin/getopt mode=0555
763 file path=usr/bin/gettext mode=0555
764 file path=usr/bin/getvol mode=0555
765 file path=usr/bin/grep mode=0555
766 file path=usr/bin/groups mode=0555
767 file path=usr/bin/head mode=0555
768 file path=usr/bin/hostid mode=0555
769 file path=usr/bin/hostname mode=0555
770 file path=usr/bin/iconv mode=0555
771 file path=usr/bin/id mode=0555
772 file path=usr/bin/infocmp mode=0555
773 file path=usr/bin/iostat mode=0555
774 file path=usr/bin/isainfo mode=0555
775 file path=usr/bin/isalist mode=0555
776 file path=usr/bin/kbd mode=0555
777 file path=usr/bin/keylogin mode=0555
778 file path=usr/bin/keylogout mode=0555
779 file path=usr/bin/kmfcfg mode=0555
780 file path=usr/bin/kvmstat mode=0555
781 file path=usr/bin/line mode=0555
782 file path=usr/bin/listdgrp mode=0555
783 file path=usr/bin/listusers mode=0555
784 file path=usr/bin/loadkeys mode=0555
```

```
785 file path=usr/bin/logger mode=0555
786 file path=usr/bin/login mode=4555
787 file path=usr/bin/logins mode=0750
788 file path=usr/bin/ls mode=0555
789 file path=usr/bin/m4 mode=0555
790 file path=usr/bin/mach mode=0555
791 file path=usr/bin/mail group=mail mode=2511
792 file path=usr/bin/mailx group=mail mode=2511
793 file path=usr/bin/makedev mode=0555
794 file path=usr/bin/mesg mode=0555
795 file path=usr/bin/mkdir mode=0555
796 file path=usr/bin/mkpwdict mode=0555
797 file path=usr/bin/mktemp mode=0555
798 file path=usr/bin/moe mode=0555
799 file path=usr/bin/more mode=0555
800 file path=usr/bin/mpstat mode=0555
801 file path=usr/bin/mt mode=0555
802 file path=usr/bin/netstat mode=0555
803 file path=usr/bin/newgrp group=sys mode=4755
804 file path=usr/bin/nice mode=0555
805 file path=usr/bin/optisa mode=0555
806 file path=usr/bin/pagesize mode=0555
807 file path=usr/bin/passwd group=sys mode=6555
808 file path=usr/bin/pathchk mode=0555
809 file path=usr/bin/pax mode=0555
810 file path=usr/bin/pfexec mode=0555
811 file path=usr/bin/pg mode=0555
812 file path=usr/bin/pgrep mode=0555
813 file path=usr/bin/pktool mode=0555
814 file path=usr/bin/pr mode=0555
815 file path=usr/bin/printf mode=0555
816 file path=usr/bin/priocntl mode=0555
817 file path=usr/bin/profiles mode=0555
818 file path=usr/bin/projects mode=0555
819 file path=usr/bin/putdev mode=0555
820 file path=usr/bin/putdgrp mode=0555
821 file path=usr/bin/pwd mode=0555
822 file path=usr/bin/renice mode=0555
823 file path=usr/bin/rm mode=0555
824 file path=usr/bin/rmdir mode=0555
825 file path=usr/bin/roles mode=0555
826 file path=usr/bin/rpcinfo mode=0555
827 file path=usr/bin/runat mode=0555
828 file path=usr/bin/script mode=0555
829 file path=usr/bin/sed mode=0555
830 file path=usr/bin/setfacl mode=0555
831 file path=usr/bin/setpgrp group=sys mode=0555
832 file path=usr/bin/settime mode=0555
833 file path=usr/bin/shcomp mode=0555
834 file path=usr/bin/strchg group=root mode=0555
835 file path=usr/bin/strconf group=root mode=0555
836 file path=usr/bin/stty mode=0555
837 file path=usr/bin/su group=sys mode=4555
838 file path=usr/bin/svcprop mode=0555
839 file path=usr/bin/svcs mode=0555
840 file path=usr/bin/tabs mode=0555
841 file path=usr/bin/tail mode=0555
842 file path=usr/bin/tic mode=0555
843 file path=usr/bin/time mode=0555
844 file path=usr/bin/tip mode=4511 owner=uucp
845 file path=usr/bin/tpmadm mode=0555
846 file path=usr/bin/tput mode=0555
847 file path=usr/bin/tr mode=0555
848 file path=usr/bin/true mode=0555
849 file path=usr/bin/tty mode=0555
850 file path=usr/bin/tzselect mode=0555
```



```
851 file path=usr/bin/userattr mode=0555
852 file path=usr/bin/vmstat mode=0555
853 file path=usr/bin/which mode=0555
854 file path=usr/bin/who mode=0555
855 file path=usr/bin/wracct mode=0555
856 file path=usr/bin/write group=tty mode=2555
857 file path=usr/bin/xargs mode=0555
858 file path=usr/bin/xstr mode=0555
859 file path=usr/has/bin/edit mode=0555
860 file path=usr/has/bin/sh mode=0555
861 file path=usr/has/man/man1has/edit.1has
862 file path=usr/has/man/man1has/ex.1has
863 file path=usr/has/man/man1has/sh.1has
864 file path=usr/has/man/man1has/vi.1has
865 file path=usr/kernel/drv/$(ARCH64)/dump group=sys
866 file path=usr/kernel/drv/$(ARCH64)/fssnap group=sys
867 file path=usr/kernel/drv/$(ARCH64)/kstat group=sys
868 file path=usr/kernel/drv/$(ARCH64)/ksyms group=sys
869 file path=usr/kernel/drv/$(ARCH64)/logindmux group=sys
870 file path=usr/kernel/drv/$(ARCH64)/ptm group=sys
871 file path=usr/kernel/drv/$(ARCH64)/pts group=sys
872 $(i386_ONLY)file path=usr/kernel/drv/dump group=sys
873 file path=usr/kernel/drv/dump.conf group=sys
874 $(i386_ONLY)file path=usr/kernel/drv/fssnap group=sys
875 file path=usr/kernel/drv/fssnap.conf group=sys
876 $(i386_ONLY)file path=usr/kernel/drv/kstat group=sys
877 file path=usr/kernel/drv/kstat.conf group=sys
878 $(i386_ONLY)file path=usr/kernel/drv/ksyms group=sys
879 file path=usr/kernel/drv/ksyms.conf group=sys
880 $(i386_ONLY)file path=usr/kernel/drv/logindmux group=sys
881 file path=usr/kernel/drv/logindmux.conf group=sys
882 $(i386_ONLY)file path=usr/kernel/drv/ptm group=sys
883 file path=usr/kernel/drv/ptm.conf group=sys
884 $(i386_ONLY)file path=usr/kernel/drv/pts group=sys
885 file path=usr/kernel/drv/pts.conf group=sys
886 file path=usr/kernel/exec/$(ARCH64)/javaexec group=sys mode=0755
887 file path=usr/kernel/exec/$(ARCH64)/shbinexec group=sys mode=0755
888 $(i386_ONLY)file path=usr/kernel/exec/javaexec group=sys mode=0755
889 $(i386_ONLY)file path=usr/kernel/exec/shbinexec group=sys mode=0755
890 file path=usr/kernel/fs/$(ARCH64)/fdfs group=sys mode=0755
891 file path=usr/kernel/fs/$(ARCH64)/pcfs group=sys mode=0755
892 $(i386_ONLY)file path=usr/kernel/fs/fdfs group=sys mode=0755
893 $(i386_ONLY)file path=usr/kernel/fs/pcfs group=sys mode=0755
894 file path=usr/kernel/sched/$(ARCH64)/FX group=sys mode=0755
895 file path=usr/kernel/sched/$(ARCH64)/FX_DPTBL group=sys mode=0755
896 file path=usr/kernel/sched/$(ARCH64)/IA group=sys mode=0755
897 file path=usr/kernel/sched/$(ARCH64)/RT group=sys mode=0755
898 file path=usr/kernel/sched/$(ARCH64)/RT_DPTBL group=sys mode=0755
899 $(i386_ONLY)file path=usr/kernel/sched/FX group=sys mode=0755
900 $(i386_ONLY)file path=usr/kernel/sched/FX_DPTBL group=sys mode=0755
901 $(i386_ONLY)file path=usr/kernel/sched/IA group=sys mode=0755
902 $(i386_ONLY)file path=usr/kernel/sched/RT group=sys mode=0755
903 $(i386_ONLY)file path=usr/kernel/sched/RT_DPTBL group=sys mode=0755
904 file path=usr/kernel/strmod/$(ARCH64)/cryptmod group=sys mode=0755
905 file path=usr/kernel/strmod/$(ARCH64)/rlmod group=sys mode=0755
906 file path=usr/kernel/strmod/$(ARCH64)/telmod group=sys mode=0755
907 $(i386_ONLY)file path=usr/kernel/strmod/cryptmod group=sys mode=0755
908 $(i386_ONLY)file path=usr/kernel/strmod/rlmod group=sys mode=0755
909 $(i386_ONLY)file path=usr/kernel/strmod/telmod group=sys mode=0755
910 file path=usr/kernel/sys/$(ARCH64)/acctctl group=sys mode=0755
911 file path=usr/kernel/sys/$(ARCH64)/exacctsys group=sys mode=0755
912 file path=usr/kernel/sys/$(ARCH64)/sysacct group=sys mode=0755
913 $(i386_ONLY)file path=usr/kernel/sys/acctctl group=sys mode=0755
914 $(i386_ONLY)file path=usr/kernel/sys/exacctsys group=sys mode=0755
915 $(i386_ONLY)file path=usr/kernel/sys/sysacct group=sys mode=0755
916 file path=usr/kvm/README group=sys
```

```
file path=usr/lib/$(ARCH64)/libshare.so.1
file path=usr/lib/audit/audit_record_attr mode=0444
file path=usr/lib/calprog mode=0555
file path=usr/lib/class/FX/FXdispadmin mode=0555
file path=usr/lib/class/FX/FXpriocntl mode=0555
file path=usr/lib/class/IA/IAdispadmin mode=0555
file path=usr/lib/class/IA/IApriocntl mode=0555
file path=usr/lib/class/RT/RTdispadmin mode=0555
file path=usr/lib/class/RT/RTpriocntl mode=0555
file path=usr/lib/class/SDC/SDCdispadmin mode=0555
file path=usr/lib/class/SDC/SDCpriocntl mode=0555
file path=usr/lib/class/TS/TSdispadmin mode=0555
file path=usr/lib/class/TS/TSpriocntl mode=0555
file path=usr/lib/devfsadm/linkmod/SUNW_audio_link.so group=sys
file path=usr/lib/devfsadm/linkmod/SUNW_cfg_link.so group=sys
file path=usr/lib/devfsadm/linkmod/SUNW_disk_link.so group=sys
file path=usr/lib/devfsadm/linkmod/SUNW_fssnap_link.so group=sys
file path=usr/lib/devfsadm/linkmod/SUNW_ieee1394_link.so group=sys
file path=usr/lib/devfsadm/linkmod/SUNW_lofi_link.so group=sys
file path=usr/lib/devfsadm/linkmod/SUNW_md_link.so group=sys
file path=usr/lib/devfsadm/linkmod/SUNW_misc_link.so group=sys
file path=usr/lib/devfsadm/linkmod/SUNW_misc_link_$(ARCH).so group=sys
file path=usr/lib/devfsadm/linkmod/SUNW_port_link.so group=sys
file path=usr/lib/devfsadm/linkmod/SUNW_ramdisk_link.so group=sys
file path=usr/lib/devfsadm/linkmod/SUNW_sgen_link.so group=sys
file path=usr/lib/devfsadm/linkmod/SUNW_smp_link.so group=sys
file path=usr/lib/devfsadm/linkmod/SUNW_tape_link.so group=sys
file path=usr/lib/devfsadm/linkmod/SUNW_usb_link.so group=sys
$(i386_ONLY)file path=usr/lib/devfsadm/linkmod/SUNW_xen_link.so group=sys
file path=usr/lib/diffh mode=0555
file path=usr/lib/expreserve mode=0555
file path=usr/lib/exrecover mode=0555
file path=usr/lib/fs/cachefs/cachefsd mode=0555
file path=usr/lib/fs/cachefs/cachefslog mode=0555
file path=usr/lib/fs/cachefs/cachefspack mode=0555
file path=usr/lib/fs/cachefs/cachefsstat mode=0555
file path=usr/lib/fs/cachefs/cachefswssize mode=0555
file path=usr/lib/fs/cachefs/cfsadmin mode=0555
file path=usr/lib/fs/cachefs/cfsfstype mode=0555
file path=usr/lib/fs/cachefs/cfstagchk mode=0555
file path=usr/lib/fs/cachefs/dfshares mode=0555
file path=usr/lib/fs/cachefs/fsck mode=0555
file path=usr/lib/fs/cachefs/mount mode=0555
file path=usr/lib/fs/cachefs/share mode=0555
file path=usr/lib/fs/cachefs/umount mode=0555
file path=usr/lib/fs/cachefs/unshare mode=0555
file path=usr/lib/fs/ctfs/mount mode=0555
file path=usr/lib/fs/fd/mount mode=0555
file path=usr/lib/fs/hsfs/fstyp.so.1 mode=0555
file path=usr/lib/fs/hsfs/labelit mode=0555
file path=usr/lib/fs/lofs/mount mode=0555
file path=usr/lib/fs/mntfs/mount mode=0555
file path=usr/lib/fs/objfs/mount mode=0555
file path=usr/lib/fs/proc/mount mode=0555
file path=usr/lib/fs/sharefs/mount mode=0555
file path=usr/lib/fs/tmpfs/mount mode=0555
file path=usr/lib/fs/ufs/clri mode=0555
file path=usr/lib/fs/ufs/df mode=0555
file path=usr/lib/fs/ufs/edquota mode=0555
file path=usr/lib/fs/ufs/ff mode=0555
file path=usr/lib/fs/ufs/fsck mode=0555
file path=usr/lib/fs/ufs/fsckall mode=0555
file path=usr/lib/fs/ufs/fsdb mode=0555
file path=usr/lib/fs/ufs/fsirand mode=0555
file path=usr/lib/fs/ufs/fssnap mode=0555
file path=usr/lib/fs/ufs/fstyp.so.1 mode=0555
```



```
file path=usr/lib/fs/ufs/labelit mode=0555
file path=usr/lib/fs/ufs/lockfs mode=0555
file path=usr/lib/fs/ufs/mkfs mode=0555
file path=usr/lib/fs/ufs/ncheck mode=0555
file path=usr/lib/fs/ufs/newfs mode=0555
file path=usr/lib/fs/ufs/quot mode=0555
file path=usr/lib/fs/ufs/quota mode=4555
file path=usr/lib/fs/ufs/quotacheck mode=0555
file path=usr/lib/fs/ufs/quotaoff mode=0555
file path=usr/lib/fs/ufs/repquota mode=0555
file path=usr/lib/fs/ufs/tunefs mode=0555
file path=usr/lib/fs/ufs/ufsdump mode=4555
file path=usr/lib/fs/ufs/ufsrestore mode=4555
file path=usr/lib/fs/ufs/volcopy mode=0555
file path=usr/lib/getoptcvt mode=0555
file path=usr/lib/help/auths/locale/C/AllSolAuthsHeader.html
file path=usr/lib/help/auths/locale/C/AuditHeader.html
file path=usr/lib/help/auths/locale/C/AuthJobsAdmin.html
file path=usr/lib/help/auths/locale/C/AuthJobsUser.html
file path=usr/lib/help/auths/locale/C/AuthProfmgrAssign.html
file path=usr/lib/help/auths/locale/C/AuthProfmgrDelegate.html
file path=usr/lib/help/auths/locale/C/AuthProfmgrExecattrWrite.html
file path=usr/lib/help/auths/locale/C/AuthProfmgrRead.html
file path=usr/lib/help/auths/locale/C/AuthProfmgrWrite.html
file path=usr/lib/help/auths/locale/C/AuthReadNDMP.html
file path=usr/lib/help/auths/locale/C/AuthReadSMB.html
file path=usr/lib/help/auths/locale/C/AuthRoleAssign.html
file path=usr/lib/help/auths/locale/C/AuthRoleDelegate.html
file path=usr/lib/help/auths/locale/C/AuthRoleWrite.html
file path=usr/lib/help/auths/locale/C/BindStates.html
file path=usr/lib/help/auths/locale/C/DevAllocHeader.html
file path=usr/lib/help/auths/locale/C/DevAllocate.html
file path=usr/lib/help/auths/locale/C/DevConfig.html
file path=usr/lib/help/auths/locale/C/DevGrant.html
file path=usr/lib/help/auths/locale/C/DevRevoke.html
file path=usr/lib/help/auths/locale/C/DhcpmgrHeader.html
file path=usr/lib/help/auths/locale/C/DhcpmgrWrite.html
file path=usr/lib/help/auths/locale/C/HotplugHeader.html
file path=usr/lib/help/auths/locale/C/HotplugModify.html
file path=usr/lib/help/auths/locale/C/IdmapRules.html
file path=usr/lib/help/auths/locale/C/JobHeader.html
file path=usr/lib/help/auths/locale/C/JobsGrant.html
file path=usr/lib/help/auths/locale/C/LinkSecurity.html
file path=usr/lib/help/auths/locale/C/LoginEnable.html
file path=usr/lib/help/auths/locale/C/LoginHeader.html
file path=usr/lib/help/auths/locale/C/LoginRemote.html
file path=usr/lib/help/auths/locale/C/NetworkAutoconfRead.html
file path=usr/lib/help/auths/locale/C/NetworkAutoconfSelect.html
file path=usr/lib/help/auths/locale/C/NetworkAutoconfWlan.html
file path=usr/lib/help/auths/locale/C/NetworkAutoconfWrite.html
file path=usr/lib/help/auths/locale/C/NetworkHeader.html
file path=usr/lib/help/auths/locale/C/NetworkILBconf.html
file path=usr/lib/help/auths/locale/C/NetworkILBenable.html
file path=usr/lib/help/auths/locale/C/NetworkInterfaceConfig.html
file path=usr/lib/help/auths/locale/C/NetworkVRRP.html
file path=usr/lib/help/auths/locale/C/PriAdmin.html
file path=usr/lib/help/auths/locale/C/ProfmgrHeader.html
file path=usr/lib/help/auths/locale/C/RoleHeader.html
file path=usr/lib/help/auths/locale/C/SmfAllocate.html
file path=usr/lib/help/auths/locale/C/SmfAutofsStates.html
file path=usr/lib/help/auths/locale/C/SmfCoreadmStates.html
file path=usr/lib/help/auths/locale/C/SmfCronStates.html
file path=usr/lib/help/auths/locale/C/SmfExAcctFlowStates.html
file path=usr/lib/help/auths/locale/C/SmfExAcctNetStates.html
file path=usr/lib/help/auths/locale/C/SmfExAcctProcessStates.html
file path=usr/lib/help/auths/locale/C/SmfExAcctTaskStates.html
```

```
1049 file path=usr/lib/help/auths/locale/C/SmfHeader.html
1050 file path=usr/lib/help/auths/locale/C/SmfILBStates.html
1051 file path=usr/lib/help/auths/locale/C/SmfIPsecStates.html
1052 file path=usr/lib/help/auths/locale/C/SmfIdmapStates.html
1053 file path=usr/lib/help/auths/locale/C/SmfInetdStates.html
1054 file path=usr/lib/help/auths/locale/C/SmfLocationStates.html
1055 file path=usr/lib/help/auths/locale/C/SmfMDNSStates.html
1056 file path=usr/lib/help/auths/locale/C/SmfManageAudit.html
1057 file path=usr/lib/help/auths/locale/C/SmfManageHeader.html
1058 file path=usr/lib/help/auths/locale/C/SmfManageHotplug.html
1059 file path=usr/lib/help/auths/locale/C/SmfManageZFSSnap.html
1060 file path=usr/lib/help/auths/locale/C/SmfModifyAppl.html
1061 file path=usr/lib/help/auths/locale/C/SmfModifyDepend.html
1062 file path=usr/lib/help/auths/locale/C/SmfModifyFramework.html
1063 file path=usr/lib/help/auths/locale/C/SmfModifyHeader.html
1064 file path=usr/lib/help/auths/locale/C/SmfModifyMethod.html
1065 file path=usr/lib/help/auths/locale/C/SmfNADDStates.html
1066 file path=usr/lib/help/auths/locale/C/SmfNDMPStates.html
1067 file path=usr/lib/help/auths/locale/C/SmfNWAMStates.html
1068 file path=usr/lib/help/auths/locale/C/SmfNscdStates.html
1069 file path=usr/lib/help/auths/locale/C/SmfPowerStates.html
1070 file path=usr/lib/help/auths/locale/C/SmfReparseStates.html
1071 file path=usr/lib/help/auths/locale/C/SmfRoutingStates.html
1072 file path=usr/lib/help/auths/locale/C/SmfSMBFSStates.html
1073 file path=usr/lib/help/auths/locale/C/SmfSMBStates.html
1074 file path=usr/lib/help/auths/locale/C/SmfSendmailStates.html
1075 file path=usr/lib/help/auths/locale/C/SmfSshStates.html
1076 file path=usr/lib/help/auths/locale/C/SmfSyslogStates.html
1077 file path=usr/lib/help/auths/locale/C/SmfVRRPStates.html
1078 file path=usr/lib/help/auths/locale/C/SmfValueAudit.html
1079 file path=usr/lib/help/auths/locale/C/SmfValueCoreadm.html
1080 file path=usr/lib/help/auths/locale/C/SmfValueExAcctFlow.html
1081 file path=usr/lib/help/auths/locale/C/SmfValueExAcctNet.html
1082 file path=usr/lib/help/auths/locale/C/SmfValueExAcctProcess.html
1083 file path=usr/lib/help/auths/locale/C/SmfValueExAcctTask.html
1084 file path=usr/lib/help/auths/locale/C/SmfValueFirewall.html
1085 file path=usr/lib/help/auths/locale/C/SmfValueHeader.html
1086 file path=usr/lib/help/auths/locale/C/SmfValueIPsec.html
1087 file path=usr/lib/help/auths/locale/C/SmfValueIdmap.html
1088 file path=usr/lib/help/auths/locale/C/SmfValueInetd.html
1089 file path=usr/lib/help/auths/locale/C/SmfValueMDNS.html
1090 file path=usr/lib/help/auths/locale/C/SmfValueNADD.html
1091 file path=usr/lib/help/auths/locale/C/SmfValueNDMP.html
1092 file path=usr/lib/help/auths/locale/C/SmfValueNWAM.html
1093 file path=usr/lib/help/auths/locale/C/SmfValueProcSec.html
1094 #endif /* ! codereview */
1095 file path=usr/lib/help/auths/locale/C/SmfValueRouting.html
1096 file path=usr/lib/help/auths/locale/C/SmfValueSMB.html
1097 file path=usr/lib/help/auths/locale/C/SmfValueVscan.html
1098 file path=usr/lib/help/auths/locale/C/SmfValueVt.html
1099 file path=usr/lib/help/auths/locale/C/SmfVscanStates.html
1100 file path=usr/lib/help/auths/locale/C/SmfVtStates.html
1101 file path=usr/lib/help/auths/locale/C/SmfWpaStates.html
1102 file path=usr/lib/help/auths/locale/C/SysCpuPowerMgmt.html
1103 file path=usr/lib/help/auths/locale/C/SysDate.html
1104 file path=usr/lib/help/auths/locale/C/SysHeader.html
1105 file path=usr/lib/help/auths/locale/C/SysMaintenance.html
1106 file path=usr/lib/help/auths/locale/C/SysPowerMgmtBrightness.html
1107 file path=usr/lib/help/auths/locale/C/SysPowerMgmtHeader.html
1108 file path=usr/lib/help/auths/locale/C/SysPowerMgmtSuspend.html
1109 file path=usr/lib/help/auths/locale/C/SysPowerMgmtSuspendtoDisk.html
1110 file path=usr/lib/help/auths/locale/C/SysPowerMgmtSuspendtoRAM.html
1111 file path=usr/lib/help/auths/locale/C/SysShutdown.html
1112 file path=usr/lib/help/auths/locale/C/SysSyseventRead.html
1113 file path=usr/lib/help/auths/locale/C/SysSyseventWrite.html
1114 file path=usr/lib/help/auths/locale/C/WifiConfig.html
```



```
1115 file path=usr/lib/help/auths/locale/C/WifiWep.html
1116 file path=usr/lib/help/auths/locale/C/ZoneCloneFrom.html
1117 file path=usr/lib/help/auths/locale/C/ZoneHeader.html
1118 file path=usr/lib/help/auths/locale/C/ZoneLogin.html
1119 file path=usr/lib/help/auths/locale/C/ZoneManage.html
1120 file path=usr/lib/help/profiles/locale/C/RtAcctadm.html
1121 file path=usr/lib/help/profiles/locale/C/RtAll.html
1122 file path=usr/lib/help/profiles/locale/C/RtAuditCfg.html
1123 file path=usr/lib/help/profiles/locale/C/RtAuditCtrl.html
1124 file path=usr/lib/help/profiles/locale/C/RtAuditReview.html
1125 file path=usr/lib/help/profiles/locale/C/RtCPUPowerManagement.html
1126 file path=usr/lib/help/profiles/locale/C/RtConsUser.html
1127 file path=usr/lib/help/profiles/locale/C/RtContractObserver.html
1128 file path=usr/lib/help/profiles/locale/C/RtCronMngmnt.html
1129 file path=usr/lib/help/profiles/locale/C/RtCryptoMngmnt.html
1130 file path=usr/lib/help/profiles/locale/C/RtDHCPMngmnt.html
1131 file path=usr/lib/help/profiles/locale/C/RtDatAdmin.html
1132 file path=usr/lib/help/profiles/locale/C/RtDefault.html
1133 file path=usr/lib/help/profiles/locale/C/RtDeviceMngmnt.html
1134 file path=usr/lib/help/profiles/locale/C/RtDeviceSecurity.html
1135 file path=usr/lib/help/profiles/locale/C/RtExAcctFlow.html
1136 file path=usr/lib/help/profiles/locale/C/RtExAcctNet.html
1137 file path=usr/lib/help/profiles/locale/C/RtExAcctProcess.html
1138 file path=usr/lib/help/profiles/locale/C/RtExAcctTask.html
1139 file path=usr/lib/help/profiles/locale/C/RtFTPMngmnt.html
1140 file path=usr/lib/help/profiles/locale/C/RtFileSysMngmnt.html
1141 file path=usr/lib/help/profiles/locale/C/RtFileSysSecurity.html
1142 file path=usr/lib/help/profiles/locale/C/RtHotplugMngmnt.html
1143 file path=usr/lib/help/profiles/locale/C/RtIPFilterMngmnt.html
1144 file path=usr/lib/help/profiles/locale/C/RtIdmapMngmnt.html
1145 file path=usr/lib/help/profiles/locale/C/RtIdmapNameRulesMngmnt.html
1146 file path=usr/lib/help/profiles/locale/C/RtInetdMngmnt.html
1147 file path=usr/lib/help/profiles/locale/C/RtKerberosClntMngmnt.html
1148 file path=usr/lib/help/profiles/locale/C/RtKerberosSrvrMngmnt.html
1149 file path=usr/lib/help/profiles/locale/C/RtLogMngmnt.html
1150 file path=usr/lib/help/profiles/locale/C/RtMailMngmnt.html
1151 file path=usr/lib/help/profiles/locale/C/RtMaintAndRepair.html
1152 file path=usr/lib/help/profiles/locale/C/RtMediaBkup.html
1153 file path=usr/lib/help/profiles/locale/C/RtMediaCtlg.html
1154 file path=usr/lib/help/profiles/locale/C/RtMediaRestore.html
1155 file path=usr/lib/help/profiles/locale/C/RtNDMPMngmnt.html
1156 file path=usr/lib/help/profiles/locale/C/RtNameServiceAdmin.html
1157 file path=usr/lib/help/profiles/locale/C/RtNameServiceSecure.html
1158 file path=usr/lib/help/profiles/locale/C/RtNetAutoconfAdmin.html
1159 file path=usr/lib/help/profiles/locale/C/RtNetAutoconfUser.html
1160 file path=usr/lib/help/profiles/locale/C/RtNetILB.html
1161 file path=usr/lib/help/profiles/locale/C/RtNetIPsec.html
1162 file path=usr/lib/help/profiles/locale/C/RtNetLinkSecure.html
1163 file path=usr/lib/help/profiles/locale/C/RtNetMngmnt.html
1164 file path=usr/lib/help/profiles/locale/C/RtNetObservability.html
1165 file path=usr/lib/help/profiles/locale/C/RtNetSecure.html
1166 file path=usr/lib/help/profiles/locale/C/RtNetVRRP.html
1167 file path=usr/lib/help/profiles/locale/C/RtNetWifiMngmnt.html
1168 file path=usr/lib/help/profiles/locale/C/RtNetWifiSecure.html
1169 file path=usr/lib/help/profiles/locale/C/RtObAccessMngmnt.html
1170 file path=usr/lib/help/profiles/locale/C/RtOperator.html
1171 file path=usr/lib/help/profiles/locale/C/RtPriAdmin.html
1172 file path=usr/lib/help/profiles/locale/C/RtPrntAdmin.html
1173 file path=usr/lib/help/profiles/locale/C/RtProcManagement.html
1174 file path=usr/lib/help/profiles/locale/C/RtReparseMngmnt.html
1175 file path=usr/lib/help/profiles/locale/C/RtReservedProfile.html
1176 file path=usr/lib/help/profiles/locale/C/RtRightsDelegate.html
1177 file path=usr/lib/help/profiles/locale/C/RtSMBFSMngmnt.html
1178 file path=usr/lib/help/profiles/locale/C/RtSMBMngmnt.html
1179 file path=usr/lib/help/profiles/locale/C/RtSoftwareInstall.html
1180 file path=usr/lib/help/profiles/locale/C/RtSysAdmin.html
```

```
1181 file path=usr/lib/help/profiles/locale/C/RtSysEvMngmnt.html
1182 file path=usr/lib/help/profiles/locale/C/RtSysPowerMgmt.html
1183 file path=usr/lib/help/profiles/locale/C/RtSysPowerMgmtBrightness.html
1184 file path=usr/lib/help/profiles/locale/C/RtSysPowerMgmtSuspend.html
1185 file path=usr/lib/help/profiles/locale/C/RtSysPowerMgmtSuspendtoDisk.html
1186 file path=usr/lib/help/profiles/locale/C/RtSysPowerMgmtSuspendtoRAM.html
1187 file path=usr/lib/help/profiles/locale/C/RtUserMngmnt.html
1188 file path=usr/lib/help/profiles/locale/C/RtUserSecurity.html
1189 file path=usr/lib/help/profiles/locale/C/RtVscanMngmnt.html
1190 file path=usr/lib/help/profiles/locale/C/RtZFSFileSysMngmnt.html
1191 file path=usr/lib/help/profiles/locale/C/RtZFSStorageMngmnt.html
1192 file path=usr/lib/help/profiles/locale/C/RtZoneMngmnt.html
1193 file path=usr/lib/help/profiles/locale/C/RtZoneSecurity.html
1194 file path=usr/lib/hotplugd mode=0555
1195 file path=usr/lib/iconv/646da.8859.t mode=0444
1196 file path=usr/lib/iconv/646de.8859.t mode=0444
1197 file path=usr/lib/iconv/646en.8859.t mode=0444
1198 file path=usr/lib/iconv/646es.8859.t mode=0444
1199 file path=usr/lib/iconv/646fr.8859.t mode=0444
1200 file path=usr/lib/iconv/646it.8859.t mode=0444
1201 file path=usr/lib/iconv/646sv.8859.t mode=0444
1202 file path=usr/lib/iconv/8859.646.t mode=0444
1203 file path=usr/lib/iconv/8859.646da.t mode=0444
1204 file path=usr/lib/iconv/8859.646de.t mode=0444
1205 file path=usr/lib/iconv/8859.646en.t mode=0444
1206 file path=usr/lib/iconv/8859.646es.t mode=0444
1207 file path=usr/lib/iconv/8859.646fr.t mode=0444
1208 file path=usr/lib/iconv/8859.646it.t mode=0444
1209 file path=usr/lib/iconv/8859.646sv.t mode=0444
1210 file path=usr/lib/iconv/iconv_data mode=0444
1211 file path=usr/lib/idmapd mode=0555
1212 file path=usr/lib/inet/$(ARCH32)/in.iked mode=0555
1213 file path=usr/lib/inet/$(ARCH64)/in.iked mode=0555
1214 file path=usr/lib/inet/certdb mode=0555
1215 file path=usr/lib/inet/certlocal mode=0555
1216 file path=usr/lib/inet/certrldb mode=0555
1217 file path=usr/lib/inet/inetd mode=0555
1218 file path=usr/lib/intrd mode=0555
1219 file path=usr/lib/isaexec mode=0555
1220 file path=usr/lib/kssladm mode=0555
1221 $(sparc_ONLY)file path=usr/lib/ld.so
1222 file path=usr/lib/libshare.so.1
1223 file path=usr/lib/makekey mode=0555
1224 file path=usr/lib/more.help
1225 file path=usr/lib/newsyslog group=sys mode=0555
1226 file path=usr/lib/passmgmt group=sys mode=0555
1227 file path=usr/lib/pci/pcidr mode=0555
1228 file path=usr/lib/pci/pcidr_plugin.so
1229 file path=usr/lib/pfexecd mode=0555
1230 file path=usr/lib/platexec mode=0555
1231 file path=usr/lib/rcm/modules/SUNW_aggr_rcm.so mode=0555
1232 file path=usr/lib/rcm/modules/SUNW_cluster_rcm.so mode=0555
1233 file path=usr/lib/rcm/modules/SUNW_dump_rcm.so mode=0555
1234 file path=usr/lib/rcm/modules/SUNW_filesys_rcm.so mode=0555
1235 file path=usr/lib/rcm/modules/SUNW_ibpart_rcm.so mode=0555
1236 file path=usr/lib/rcm/modules/SUNW_ip_anon_rcm.so mode=0555
1237 file path=usr/lib/rcm/modules/SUNW_ip_rcm.so mode=0555
1238 file path=usr/lib/rcm/modules/SUNW_mpxio_rcm.so mode=0555
1239 file path=usr/lib/rcm/modules/SUNW_network_rcm.so mode=0555
1240 file path=usr/lib/rcm/modules/SUNW_swap_rcm.so mode=0555
1241 $(sparc_ONLY)file path=usr/lib/rcm/modules/SUNW_ttymux_rcm.so mode=0555
1242 file path=usr/lib/rcm/modules/SUNW_vlan_rcm.so mode=0555
1243 file path=usr/lib/rcm/modules/SUNW_vnic_rcm.so mode=0555
1244 file path=usr/lib/rcm/rcm_daemon mode=0555
1245 file path=usr/lib/reparse/reparsed group=sys mode=0555
1246 file path=usr/lib/saf/listen group=sys mode=0755
```



```
1247 file path=usr/lib/saf/nlps_server group=sys mode=0755
1248 file path=usr/lib/saf/sac group=sys mode=0555
1249 file path=usr/lib/saf/ttymon group=sys mode=0555
1250 file path=usr/lib/sysevent/modules/datalink_mod.so
1251 file path=usr/lib/sysevent/modules/devfsadmd_mod.so
1252 file path=usr/lib/sysevent/modules/sysevent_conf_mod.so
1253 file path=usr/lib/sysevent/modules/sysevent_reg_mod.so
1254 file path=usr/lib/sysevent/syseventconfd mode=0555
1255 file path=usr/lib/sysevent/syseventd mode=0555
1256 file path=usr/lib/utmp_update mode=4555
1257 file path=usr/lib/utmpd mode=0555
1258 file path=usr/lib/vtdaemon mode=0555
1259 file path=usr/lib/vtinfo mode=0555
1260 file path=usr/lib/vtxlock mode=0555
1261 file path=usr/sadm/bin/puttext mode=0555
1262 file path=usr/sadm/install/miniroot.db group=sys mode=0444
1263 file path=usr/sadm/install/scripts/i.ipsecalgs group=sys mode=0555
1264 file path=usr/sadm/install/scripts/i.kcfconf group=sys mode=0555
1265 file path=usr/sadm/install/scripts/i.kmfconf group=sys mode=0555
1266 file path=usr/sadm/install/scripts/i.manifest group=sys mode=0555
1267 file path=usr/sadm/install/scripts/i.pkcs11conf group=sys mode=0555
1268 file path=usr/sadm/install/scripts/i.rbac group=sys mode=0555
1269 file path=usr/sadm/install/scripts/r.ipsecalgs group=sys mode=0555
1270 file path=usr/sadm/install/scripts/r.kcfconf group=sys mode=0555
1271 file path=usr/sadm/install/scripts/r.kmfconf group=sys mode=0555
1272 file path=usr/sadm/install/scripts/r.manifest group=sys mode=0555
1273 file path=usr/sadm/install/scripts/r.pkcs11conf group=sys mode=0555
1274 file path=usr/sadm/install/scripts/r.rbac group=sys mode=0555
1275 file path=usr/sadm/ugdates mode=0444
1276 $(i386_ONLY)file path=usr/sbin/$(ARCH32)/add_drv group=sys mode=0555
1277 $(i386_ONLY)file path=usr/sbin/$(ARCH32)/modinfo group=sys mode=0555
1278 $(i386_ONLY)file path=usr/sbin/$(ARCH32)/modload group=sys mode=0555
1279 $(i386_ONLY)file path=usr/sbin/$(ARCH32)/modunload group=sys mode=0555
1280 $(i386_ONLY)file path=usr/sbin/$(ARCH32)/pbind group=sys mode=0555
1281 $(i386_ONLY)file path=usr/sbin/$(ARCH32)/prtconf group=sys mode=2555
1282 $(i386_ONLY)file path=usr/sbin/$(ARCH32)/psrset group=sys mode=0555
1283 $(i386_ONLY)file path=usr/sbin/$(ARCH32)/rem_drv group=sys mode=0555
1284 $(i386_ONLY)file path=usr/sbin/$(ARCH32)/swap group=sys mode=2555
1285 $(i386_ONLY)file path=usr/sbin/$(ARCH32)/sysdef group=sys mode=2555
1286 $(i386_ONLY)file path=usr/sbin/$(ARCH32)/update_drv group=sys mode=0555
1287 $(i386_ONLY)file path=usr/sbin/$(ARCH32)/whodo mode=4555
1288 file path=usr/sbin/$(ARCH64)/add_drv group=sys mode=0555
1289 file path=usr/sbin/$(ARCH64)/modinfo group=sys mode=0555
1290 file path=usr/sbin/$(ARCH64)/modload group=sys mode=0555
1291 file path=usr/sbin/$(ARCH64)/modunload group=sys mode=0555
1292 file path=usr/sbin/$(ARCH64)/pbind group=sys mode=0555
1293 file path=usr/sbin/$(ARCH64)/prtconf group=sys mode=2555
1294 file path=usr/sbin/$(ARCH64)/psrset group=sys mode=0555
1295 file path=usr/sbin/$(ARCH64)/rem_drv group=sys mode=0555
1296 file path=usr/sbin/$(ARCH64)/swap group=sys mode=2555
1297 file path=usr/sbin/$(ARCH64)/sysdef group=sys mode=2555
1298 file path=usr/sbin/$(ARCH64)/update_drv group=sys mode=0555
1299 file path=usr/sbin/$(ARCH64)/whodo mode=4555
1300 file path=usr/sbin/6to4relay mode=0555
1301 file path=usr/sbin/acctadm mode=0555
1302 file path=usr/sbin/allocate mode=4555
1303 file path=usr/sbin/arp mode=0555
1304 file path=usr/sbin/audit mode=0555
1305 file path=usr/sbin/auditconfig mode=0555
1306 file path=usr/sbin/auditd mode=0555
1307 file path=usr/sbin/auditrecord mode=0555
1308 file path=usr/sbin/auditreduce mode=0555
1309 file path=usr/sbin/auditstat mode=0555
1310 file path=usr/sbin/cfgadm mode=0555
1311 file path=usr/sbin/chroot mode=0555
1312 file path=usr/sbin/clear_locks mode=0555
```

```
1313 file path=usr/sbin/clinfo mode=0555
1314 file path=usr/sbin/clri mode=0555
1315 file path=usr/sbin/consadm group=sys mode=0555
1316 file path=usr/sbin/cron group=sys mode=0555
1317 file path=usr/sbin/devfsadm group=sys mode=0755
1318 file path=usr/sbin/devinfo mode=0555
1319 file path=usr/sbin/df mode=0555
1320 file path=usr/sbin/dfmounts mode=0555
1321 file path=usr/sbin/dispadmin mode=0555
1322 file path=usr/sbin/dminfo mode=0555
1323 file path=usr/sbin/dumpadm mode=0555
1324 file path=usr/sbin/eeprom group=sys mode=2555
1325 file path=usr/sbin/ff mode=0555
1326 file path=usr/sbin/fmthard group=sys mode=0555
1327 file path=usr/sbin/format mode=0555
1328 file path=usr/sbin/fsck mode=0555
1329 file path=usr/sbin/fstyp group=sys mode=0555
1330 file path=usr/sbin/fuser mode=0555
1331 file path=usr/sbin/getdevpolicy group=sys mode=0555
1332 file path=usr/sbin/getmajor group=sys mode=0755
1333 file path=usr/sbin/groupadd group=sys mode=0555
1334 file path=usr/sbin/groupdel group=sys mode=0555
1335 file path=usr/sbin/groupmod group=sys mode=0555
1336 file path=usr/sbin/grpck mode=0555
1337 file path=usr/sbin/halt mode=0755
1338 file path=usr/sbin/hotplug mode=0555
1339 file path=usr/sbin/idmap mode=0555
1340 file path=usr/sbin/if_mpadm mode=0555
1341 file path=usr/sbin/ikeadm mode=0555
1342 file path=usr/sbin/ikecert mode=0555
1343 file path=usr/sbin/inetadm mode=0555
1344 file path=usr/sbin/inetconv mode=0555
1345 file path=usr/sbin/install mode=0555
1346 file path=usr/sbin/installboot group=sys mode=0555
1347 file path=usr/sbin/ipaddrsel mode=0555
1348 file path=usr/sbin/ipsecalgs mode=0555
1349 file path=usr/sbin/ipsecconf mode=0555
1350 file path=usr/sbin/ipseckey mode=0555
1351 file path=usr/sbin/keyserv group=sys mode=0555
1352 file path=usr/sbin/killall mode=0555
1353 file path=usr/sbin/ksslcfg mode=0555
1354 file path=usr/sbin/link mode=0555
1355 file path=usr/sbin/locator mode=0555
1356 file path=usr/sbin/lofiadm mode=0555
1357 file path=usr/sbin/logadm mode=0555
1358 file path=usr/sbin/makedbm mode=0555
1359 file path=usr/sbin/mkdevalloc mode=0555
1360 file path=usr/sbin/mkfile mode=0555
1361 file path=usr/sbin/mknod mode=0555
1362 file path=usr/sbin/mountall group=sys mode=0555
1363 file path=usr/sbin/msgid mode=0555
1364 file path=usr/sbin/mvdir mode=0555
1365 file path=usr/sbin/ndd mode=0555
1366 file path=usr/sbin/nlsadmin group=adm mode=0755
1367 file path=usr/sbin/nscd mode=0555
1368 file path=usr/sbin/nwamadm mode=0555
1369 file path=usr/sbin/nwamcfg mode=0555
1370 file path=usr/sbin/pmadm group=sys mode=0555
1371 file path=usr/sbin/praudit mode=0555
1372 $(i386_ONLY)file path=usr/sbin/prtdiag group=sys mode=2755
1373 file path=usr/sbin/prtvtoc group=sys mode=0555
1374 file path=usr/sbin/psradm group=sys mode=0555
1375 file path=usr/sbin/psrinfo group=sys mode=0555
1376 file path=usr/sbin/pwck mode=0555
1377 file path=usr/sbin/pwconv group=sys mode=0555
1378 file path=usr/sbin/raidctl mode=0555
```



```
1379 file path=usr/sbin/ramdiskadm mode=0555
1380 file path=usr/sbin/rctladm mode=0555
1381 file path=usr/sbin/root_archive group=sys mode=0555
1382 file path=usr/sbin/rpcbind mode=0555
1383 $(i386_ONLY)file path=usr/sbin/rtc mode=0555
1384 file path=usr/sbin/sacadm group=sys mode=4755
1385 file path=usr/sbin/setmnt mode=0555
1386 file path=usr/sbin/shareall mode=0555
1387 file path=usr/sbin/sharectl mode=0555
1388 file path=usr/sbin/sharemgr mode=0555
1389 file path=usr/sbin/shutdown group=sys mode=0755
1390 file path=usr/sbin/smbios mode=0555
1391 file path=usr/sbin/stmsboot mode=0555
1392 file path=usr/sbin/strace group=sys mode=0555
1393 file path=usr/sbin/strclean group=sys mode=0555
1394 file path=usr/sbin/strerr group=sys mode=0555
1395 file path=usr/sbin/sttydefs group=sys mode=0755
1396 file path=usr/sbin/svcadm mode=0555
1397 file path=usr/sbin/svccfg mode=0555
1398 file path=usr/sbin/syncinit mode=0555
1399 file path=usr/sbin/syncloop mode=0555
1400 file path=usr/sbin/syncstat mode=0555
1401 file path=usr/sbin/syseventadm group=sys mode=0555
1402 file path=usr/sbin/syslogd group=sys mode=0555
1403 file path=usr/sbin/tar mode=0555
1404 file path=usr/sbin/traceroute mode=4555
1405 file path=usr/sbin/trapstat mode=0555
1406 file path=usr/sbin/ttyadm group=sys mode=0755
1407 $(i386_ONLY)file path=usr/sbin/ucodeadm mode=0555
1408 file path=usr/sbin/umountall group=sys mode=0555
1409 file path=usr/sbin/unlink mode=0555
1410 file path=usr/sbin/unshareall mode=0555
1411 file path=usr/sbin/useradd group=sys mode=0555
1412 file path=usr/sbin/userdel group=sys mode=0555
1413 file path=usr/sbin/usermod group=sys mode=0555
1414 $(sparc_ONLY)file path=usr/sbin/virtinfo mode=0555
1415 file path=usr/sbin/volcopy mode=0555
1416 file path=usr/sbin/wall group=tty mode=2555
1417 file path=usr/sbin/zdump mode=0555
1418 file path=usr/sbin/zic mode=0555
1419 file path=usr/share/doc/ksh/COMPATIBILITY
1420 file path=usr/share/doc/ksh/DESIGN
1421 file path=usr/share/doc/ksh/OBSOLETE
1422 file path=usr/share/doc/ksh/README
1423 file path=usr/share/doc/ksh/RELEASE
1424 file path=usr/share/doc/ksh/TYPES
1425 file path=usr/share/doc/ksh/images/callouts/1.png
1426 file path=usr/share/doc/ksh/images/callouts/10.png
1427 file path=usr/share/doc/ksh/images/callouts/2.png
1428 file path=usr/share/doc/ksh/images/callouts/3.png
1429 file path=usr/share/doc/ksh/images/callouts/4.png
1430 file path=usr/share/doc/ksh/images/callouts/5.png
1431 file path=usr/share/doc/ksh/images/callouts/6.png
1432 file path=usr/share/doc/ksh/images/callouts/7.png
1433 file path=usr/share/doc/ksh/images/callouts/8.png
1434 file path=usr/share/doc/ksh/images/callouts/9.png
1435 file path=usr/share/doc/ksh/images/tag_bourne.png
1436 file path=usr/share/doc/ksh/images/tag_i18n.png
1437 file path=usr/share/doc/ksh/images/tag_ksh.png
1438 file path=usr/share/doc/ksh/images/tag_ksh88.png
1439 file path=usr/share/doc/ksh/images/tag_ksh93.png
1440 file path=usr/share/doc/ksh/images/tag_l10n.png
1441 file path=usr/share/doc/ksh/images/tag_perf.png
1442 file path=usr/share/doc/ksh/shell_styleguide.docbook
1443 file path=usr/share/lib/mailx/mailx.help
1444 file path=usr/share/lib/mailx/mailx.help.~
```

```
file path=usr/share/lib/tabset/3101
file path=usr/share/lib/tabset/beehive
file path=usr/share/lib/tabset/hds
file path=usr/share/lib/tabset/hds3
file path=usr/share/lib/tabset/std
file path=usr/share/lib/tabset/stdcrt
file path=usr/share/lib/tabset/teleray
file path=usr/share/lib/tabset/vt100
file path=usr/share/lib/tabset/wyse-adds
file path=usr/share/lib/tabset/xerox1720
file path=usr/share/lib/termcap
file path=usr/share/lib/unittab
file path=usr/share/lib/xml/dtd/adt_record.dtd.1
file path=usr/share/lib/xml/dtd/kmfpolicy.dtd
file path=usr/share/lib/xml/dtd/service_bundle.dtd.1 group=sys
file path=usr/share/lib/xml/style/adt_record.xsl.1
file path=var/adm/aculog mode=0600 owner=uucp preserve=true
file path=var/adm/spellhist mode=0666 preserve=true
file path=var/adm/utmpx preserve=true
file path=var/adm/wtmpx group=adm owner=adm preserve=true
file path=var/log/authlog group=sys mode=0600 preserve=true
file path=var/log/syslog group=sys preserve=true
file path=var/saf/zsmon/log group=sys preserve=true
file path=var/spool/cron/crontabs/adm group=sys mode=0600 preserve=true
file path=var/spool/cron/crontabs/root group=sys mode=0600 preserve=true
hardlink path=etc/rc2.d/S20sysetup target=../../etc/init.d/sysetup
hardlink path=etc/rc2.d/S73cachefs.daemon \
    target=../../etc/init.d/cachefs.daemon
hardlink path=etc/rc2.d/S89PRESERVE target=../../etc/init.d/PRESERVE
$(sparc_ONLY)hardlink path=etc/svc/profile/platform_SUNW,Sun-Fire-V890.xml \
    target=./platform_SUNW,Sun-Fire-880.xml
$(sparc_ONLY)hardlink \
    path=etc/svc/profile/platform_SUNW,UltraSPARC-IIe-NetraCT-40.xml \
    target=./platform_SUNW,UltraSPARC-IIi-Netract.xml
$(sparc_ONLY)hardlink \
    path=etc/svc/profile/platform_SUNW,UltraSPARC-IIe-NetraCT-60.xml \
    target=./platform_SUNW,UltraSPARC-IIi-Netract.xml
hardlink path=sbin/rc5 target=../sbin/rc0
hardlink path=sbin/rc6 target=../sbin/rc0
hardlink path=usr/bin/$(ARCH32)/encrypt target=decrypt
hardlink path=usr/bin/$(ARCH32)/ksh target=ksh93
hardlink path=usr/bin/$(ARCH32)/mac target=digest
hardlink path=usr/bin/$(ARCH32)/rksh target=ksh93
hardlink path=usr/bin/$(ARCH32)/rksh93 target=ksh93
$(i386_ONLY)hardlink path=usr/bin/$(ARCH32)/w target=uptime
hardlink path=usr/bin/$(ARCH64)/encrypt target=decrypt
hardlink path=usr/bin/$(ARCH64)/ksh target=ksh93
hardlink path=usr/bin/$(ARCH64)/mac target=digest
hardlink path=usr/bin/$(ARCH64)/rksh target=ksh93
hardlink path=usr/bin/$(ARCH64)/rksh93 target=ksh93
hardlink path=usr/bin/$(ARCH64)/w target=uptime
hardlink path=usr/bin/bg target=../../usr/bin/alias
hardlink path=usr/bin/cd target=../../usr/bin/alias
hardlink path=usr/bin/cksum target=../../usr/bin/alias
hardlink path=usr/bin/cmp target=../../usr/bin/alias
hardlink path=usr/bin/comm target=../../usr/bin/alias
hardlink path=usr/bin/command target=../../usr/bin/alias
hardlink path=usr/bin/cut target=../../usr/bin/alias
hardlink path=usr/bin/decrypt target=../../usr/lib/isaexec
hardlink path=usr/bin/digest target=../../usr/lib/isaexec
hardlink path=usr/bin/dispgid target=../../usr/bin/ckgid
hardlink path=usr/bin/dispuid target=../../usr/bin/ckuid
hardlink path=usr/bin/edit target=../has/bin/edit
hardlink path=usr/bin/encrypt target=../../usr/lib/isaexec
hardlink path=usr/bin/fc target=../../usr/bin/alias
hardlink path=usr/bin/fg target=../../usr/bin/alias
```



```
hardlink path=usr/bin/getopts target=../../usr/bin/alias
hardlink path=usr/bin/hash target=../../usr/bin/alias
hardlink path=usr/bin/jobs target=../../usr/bin/alias
hardlink path=usr/bin/join target=../../usr/bin/alias
hardlink path=usr/bin/kill target=../../usr/bin/alias
hardlink path=usr/bin/ksh target=../../usr/lib/isaexec
hardlink path=usr/bin/ksh93 target=../../usr/lib/isaexec
hardlink path=usr/bin/ln target=../../usr/bin/cp
hardlink path=usr/bin/logname target=../../usr/bin/alias
hardlink path=usr/bin/mac target=../../usr/lib/isaexec
hardlink path=usr/bin/mv target=../../usr/bin/cp
hardlink path=usr/bin/newtask target=../../usr/lib/isaexec
hardlink path=usr/bin/nohup target=../../usr/lib/isaexec
hardlink path=usr/bin/page target=../../usr/bin/more
hardlink path=usr/bin/paste target=../../usr/bin/alias
hardlink path=usr/bin/pfbash target=../../usr/bin/pfexec
hardlink path=usr/bin/pfcsh target=../../usr/bin/pfexec
hardlink path=usr/bin/pfksh target=../../usr/bin/pfexec
hardlink path=usr/bin/pfksh93 target=../../usr/bin/pfexec
hardlink path=usr/bin/pfrksh target=../../usr/bin/pfexec
hardlink path=usr/bin/pfrksh93 target=../../usr/bin/pfexec
hardlink path=usr/bin/pfsh target=../../usr/bin/pfexec
hardlink path=usr/bin/pftcsh target=../../usr/bin/pfexec
hardlink path=usr/bin/pfzsh target=../../usr/bin/pfexec
hardlink path=usr/bin/pkill target=../../usr/bin/pgrep
hardlink path=usr/bin/prctl target=../../usr/lib/isaexec
hardlink path=usr/bin/print target=../../usr/bin/alias
hardlink path=usr/bin/prstat target=../../usr/lib/isaexec
hardlink path=usr/bin/ps target=../../usr/lib/isaexec
hardlink path=usr/bin/read target=../../usr/bin/alias
hardlink path=usr/bin/red target=../../usr/bin/ed
hardlink path=usr/bin/rev target=../../usr/bin/alias
hardlink path=usr/bin/rksh target=../../usr/lib/isaexec
hardlink path=usr/bin/rksh93 target=../../usr/lib/isaexec
hardlink path=usr/bin/savecore target=../../usr/lib/isaexec
hardlink path=usr/bin/setuname target=../../usr/lib/isaexec
hardlink path=usr/bin/sleep target=../../usr/bin/alias
hardlink path=usr/bin/sum target=../../usr/bin/alias
hardlink path=usr/bin/tee target=../../usr/bin/alias
hardlink path=usr/bin/test target=../../usr/bin/alias
hardlink path=usr/bin/touch target=../../usr/bin/settime
hardlink path=usr/bin/type target=../../usr/bin/alias
hardlink path=usr/bin/ulimit target=../../usr/bin/alias
hardlink path=usr/bin/umask target=../../usr/bin/alias
hardlink path=usr/bin/unalias target=../../usr/bin/alias
hardlink path=usr/bin/uniq target=../../usr/bin/alias
hardlink path=usr/bin/uptime target=../../usr/lib/isaexec
hardlink path=usr/bin/vedit target=../has/bin/edit
hardlink path=usr/bin/w target=../../usr/lib/isaexec
hardlink path=usr/bin/wait target=../../usr/bin/alias
hardlink path=usr/bin/wc target=../../usr/bin/alias
hardlink path=usr/has/bin/ex target=edit
hardlink path=usr/has/bin/pfsh target=../../bin/pfexec
hardlink path=usr/has/bin/vedit target=edit
hardlink path=usr/has/bin/vi target=edit
hardlink path=usr/has/bin/view target=edit
hardlink path=usr/lib/fs/hsfs/fstyp target=../../../sbin/fstyp
hardlink path=usr/lib/fs/ufs/dcopy target=../../../../usr/lib/fs/ufs/clri
hardlink path=usr/lib/fs/ufs/fstyp target=../../../sbin/fstyp
hardlink path=usr/lib/fs/ufs/quotaon \
    target=../../../../usr/lib/fs/ufs/quotaoff
hardlink path=usr/lib/inet/in.iked target=../../../usr/lib/isaexec
hardlink path=usr/sadm/bin/dispgid target=../../../usr/bin/ckgid
hardlink path=usr/sadm/bin/dispuid target=../../../usr/bin/ckuid
hardlink path=usr/sadm/bin/errange target=../../../usr/bin/ckrange
hardlink path=usr/sadm/bin/errdate target=../../../usr/bin/ckdate
```

```
1577 hardlink path=usr/sadm/bin/errgid target=../../../usr/bin/ckgid
1578 hardlink path=usr/sadm/bin/errint target=../../../usr/bin/ckint
1579 hardlink path=usr/sadm/bin/erritem target=../../../usr/bin/ckitem
1580 hardlink path=usr/sadm/bin/errpath target=../../../usr/bin/ckpath
1581 hardlink path=usr/sadm/bin/errstr target=../../../usr/bin/ckstr
1582 hardlink path=usr/sadm/bin/errtime target=../../../usr/bin/cktime
1583 hardlink path=usr/sadm/bin/erruid target=../../../usr/bin/ckuid
1584 hardlink path=usr/sadm/bin/erryorn target=../../../usr/bin/ckyorn
1585 hardlink path=usr/sadm/bin/helpdate target=../../../usr/bin/ckdate
1586 hardlink path=usr/sadm/bin/helpgid target=../../../usr/bin/ckgid
1587 hardlink path=usr/sadm/bin/helpint target=../../../usr/bin/ckint
1588 hardlink path=usr/sadm/bin/helpitem target=../../../usr/bin/ckitem
1589 hardlink path=usr/sadm/bin/helppath target=../../../usr/bin/ckpath
1590 hardlink path=usr/sadm/bin/helprange target=../../../usr/bin/ckrange
1591 hardlink path=usr/sadm/bin/helpstr target=../../../usr/bin/ckstr
1592 hardlink path=usr/sadm/bin/helptime target=../../../usr/bin/cktime
1593 hardlink path=usr/sadm/bin/helpuid target=../../../usr/bin/ckuid
1594 hardlink path=usr/sadm/bin/helpyorn target=../../../usr/bin/ckyorn
1595 hardlink path=usr/sadm/bin/valdate target=../../../usr/bin/ckdate
1596 hardlink path=usr/sadm/bin/valgid target=../../../usr/bin/ckgid
1597 hardlink path=usr/sadm/bin/valint target=../../../usr/bin/ckint
1598 hardlink path=usr/sadm/bin/valpath target=../../../usr/bin/ckpath
1599 hardlink path=usr/sadm/bin/valrange target=../../../usr/bin/ckrange
1600 hardlink path=usr/sadm/bin/valstr target=../../../usr/bin/ckstr
1601 hardlink path=usr/sadm/bin/valtime target=../../../usr/bin/cktime
1602 hardlink path=usr/sadm/bin/valuid target=../../../usr/bin/ckuid
1603 hardlink path=usr/sadm/bin/valyorn target=../../../usr/bin/ckyorn
1604 hardlink path=usr/sbin/add_drv target=../../usr/lib/isaexec
1605 hardlink path=usr/sbin/audlinks target=./devfsadm
1606 hardlink path=usr/sbin/consadmd target=../../usr/sbin/consadm
1607 hardlink path=usr/sbin/deallocate target=../../usr/sbin/allocate
1608 hardlink path=usr/sbin/devlinks target=./devfsadm
1609 hardlink path=usr/sbin/dfshares target=../../usr/sbin/dfmounts
1610 hardlink path=usr/sbin/disks target=./devfsadm
1611 hardlink path=usr/sbin/drvconfig target=./devfsadm
1612 hardlink path=usr/sbin/list_devices target=../../usr/sbin/allocate
1613 hardlink path=usr/sbin/mkdevmaps target=../../usr/sbin/mkdevalloc
1614 hardlink path=usr/sbin/modinfo target=../../usr/lib/isaexec
1615 hardlink path=usr/sbin/modload target=../../usr/lib/isaexec
1616 hardlink path=usr/sbin/modunload target=../../usr/lib/isaexec
1617 hardlink path=usr/sbin/pbind target=../../usr/lib/isaexec
1618 hardlink path=usr/sbin/ports target=./devfsadm
1619 hardlink path=usr/sbin/poweroff target=./halt
1620 hardlink path=usr/sbin/prtconf target=../../usr/lib/isaexec
1621 $(sparc_ONLY)hardlink path=usr/sbin/prtdiag target=../../usr/lib/platexec
1622 hardlink path=usr/sbin/psrset target=../../usr/lib/isaexec
1623 hardlink path=usr/sbin/reboot target=./halt
1624 hardlink path=usr/sbin/rem_drv target=../../usr/lib/isaexec
1625 hardlink path=usr/sbin/roleadd target=../../usr/sbin/useradd
1626 hardlink path=usr/sbin/roledel target=../../usr/sbin/userdel
1627 hardlink path=usr/sbin/rolemod target=../../usr/sbin/usermod
1628 hardlink path=usr/sbin/share target=../../usr/sbin/sharemgr
1629 hardlink path=usr/sbin/swap target=../../usr/lib/isaexec
1630 hardlink path=usr/sbin/sysdef target=../../usr/lib/isaexec
1631 hardlink path=usr/sbin/tapes target=./devfsadm
1632 hardlink path=usr/sbin/unshare target=../../usr/sbin/sharemgr
1633 hardlink path=usr/sbin/update_drv target=../../usr/lib/isaexec
1634 hardlink path=usr/sbin/whodo target=../../usr/lib/isaexec
1635 legacy pkg=SUNWcsr \
1636     desc="core software for a specific instruction-set architecture" \
1637     name="Core Solaris, (Root)"
1638 legacy pkg=SUNWcsu \
1639     desc="core software for a specific instruction-set architecture" \
1640     name="Core Solaris, (Usr)"
1641 legacy pkg=SUNWftpr desc="FTP Server Configuration Files" \
1642     name="FTP Server, (Root)"
```



```
1643 license cr_Sun license=cr_Sun
1644 license lic_CDDL license=lic_CDDL
1645 license usr/src/cmd/cmd-inet/sbin/ifparse/THIRDPARTYLICENSE \
1646     license=usr/src/cmd/cmd-inet/sbin/ifparse/THIRDPARTYLICENSE
1647 license usr/src/cmd/cmd-inet/usr.lib/in.mpathd/THIRDPARTYLICENSE \
1648     license=usr/src/cmd/cmd-inet/usr.lib/in.mpathd/THIRDPARTYLICENSE
1649 license usr/src/cmd/cmd-inet/usr.sbin/THIRDPARTYLICENSE.arp \
1650     license=usr/src/cmd/cmd-inet/usr.sbin/THIRDPARTYLICENSE.arp
1651 license usr/src/cmd/cmd-inet/usr.sbin/THIRDPARTYLICENSE.route \
1652     license=usr/src/cmd/cmd-inet/usr.sbin/THIRDPARTYLICENSE.route
1653 license usr/src/cmd/cmd-inet/usr.sbin/ifconfig/THIRDPARTYLICENSE \
1654     license=usr/src/cmd/cmd-inet/usr.sbin/ifconfig/THIRDPARTYLICENSE
1655 license usr/src/cmd/cmd-inet/usr.sbin/traceroute/THIRDPARTYLICENSE \
1656     license=usr/src/cmd/cmd-inet/usr.sbin/traceroute/THIRDPARTYLICENSE
1657 license usr/src/cmd/cron/THIRDPARTYLICENSE \
1658     license=usr/src/cmd/cron/THIRDPARTYLICENSE
1659 license usr/src/cmd/csh/THIRDPARTYLICENSE \
1660     license=usr/src/cmd/csh/THIRDPARTYLICENSE
1661 license usr/src/cmd/eeprom/THIRDPARTYLICENSE \
1662     license=usr/src/cmd/eeprom/THIRDPARTYLICENSE
1663 license usr/src/cmd/fs.d/ufs/THIRDPARTYLICENSE \
1664     license=usr/src/cmd/fs.d/ufs/THIRDPARTYLICENSE
1665 license usr/src/cmd/mt/THIRDPARTYLICENSE \
1666     license=usr/src/cmd/mt/THIRDPARTYLICENSE
1667 license usr/src/cmd/script/THIRDPARTYLICENSE \
1668     license=usr/src/cmd/script/THIRDPARTYLICENSE
1669 license usr/src/cmd/sed/THIRDPARTYLICENSE \
1670     license=usr/src/cmd/sed/THIRDPARTYLICENSE
1671 license usr/src/cmd/stat/vmstat/THIRDPARTYLICENSE \
1672     license=usr/src/cmd/stat/vmstat/THIRDPARTYLICENSE
1673 license usr/src/cmd/tail/THIRDPARTYLICENSE \
1674     license=usr/src/cmd/tail/THIRDPARTYLICENSE
1675 license usr/src/cmd/tip/THIRDPARTYLICENSE \
1676     license=usr/src/cmd/tip/THIRDPARTYLICENSE
1677 license usr/src/cmd/tr/THIRDPARTYLICENSE \
1678     license=usr/src/cmd/tr/THIRDPARTYLICENSE
1679 license usr/src/cmd/vi/THIRDPARTYLICENSE \
1680     license=usr/src/cmd/vi/THIRDPARTYLICENSE
1681 license usr/src/cmd/which/THIRDPARTYLICENSE \
1682     license=usr/src/cmd/which/THIRDPARTYLICENSE
1683 license usr/src/cmd/xstr/THIRDPARTYLICENSE \
1684     license=usr/src/cmd/xstr/THIRDPARTYLICENSE
1685 license usr/src/common/bzip2/LICENSE license=usr/src/common/bzip2/LICENSE
1686 link path=bin target=./usr/bin
1687 link path=etc/TIMEZONE target=./default/init
1688 link path=etc/autopush target=../sbin/autopush
1689 link path=etc/cfgadm target=../usr/sbin/cfgadm
1690 link path=etc/clri target=../usr/sbin/clri
1691 link path=etc/cron target=../usr/sbin/cron
1692 link path=etc/dcopy target=../usr/sbin/dcopy
1693 link path=etc/ff target=../usr/sbin/ff
1694 link path=etc/fmthard target=../usr/sbin/fmthard
1695 link path=etc/format target=../usr/sbin/format
1696 link path=etc/fsck target=../usr/sbin/fsck
1697 link path=etc/fsdb target=../usr/sbin/fsdb
1698 link path=etc/fstyp target=../usr/sbin/fstyp
1699 link path=etc/ftpusers target=./ftpd/ftpusers
1700 link path=etc/getty target=../usr/lib/saf/ttymon
1701 link path=etc/grpck target=../usr/sbin/grpck
1702 link path=etc/halt target=../usr/sbin/halt
1703 link path=etc/hosts target=./inet/hosts
1704 link path=etc/inet/ipnodes target=./hosts
1705 link path=etc/inetd.conf target=./inet/inetd.conf
1706 link path=etc/init target=../sbin/init
1707 link path=etc/install target=../usr/sbin/install
1708 link path=etc/killall target=../usr/sbin/killall
```

```
1709 link path=etc/labelit target=../usr/sbin/labelit
1710 link path=etc/lib/ld.so.1 target=../../lib/ld.so.1
1711 link path=etc/lib/libdl.so.1 target=../../lib/libdl.so.1
1712 link path=etc/lib/nss_files.so.1 target=../../lib/nss_files.so.1
1713 link path=etc/log target=../var/adm/log
1714 link path=etc/mkfs target=../usr/sbin/mkfs
1715 link path=etc/mknod target=../usr/sbin/mknod
1716 link path=etc/mount target=../sbin/mount
1717 link path=etc/mountall target=../sbin/mountall
1718 link path=etc/ncheck target=../usr/sbin/ncheck
1719 link path=etc/netmasks target=./inet/netmasks
1720 link path=etc/networks target=./inet/networks
1721 link path=etc/protocols target=./inet/protocols
1722 link path=etc/prtconf target=../usr/sbin/prtconf
1723 link path=etc/prtvtoc target=../usr/sbin/prtvtoc
1724 link path=etc/rc0 target=../sbin/rc0
1725 link path=etc/rc1 target=../sbin/rc1
1726 link path=etc/rc2 target=../sbin/rc2
1727 link path=etc/rc3 target=../sbin/rc3
1728 link path=etc/rc5 target=../sbin/rc5
1729 link path=etc/rc6 target=../sbin/rc6
1730 link path=etc/rcS target=../sbin/rcS
1731 link path=etc/reboot target=../usr/sbin/halt
1732 link path=etc/security/audit/localhost/files target=../../../../var/audit
1733 link path=etc/services target=./inet/services
1734 link path=etc/setmnt target=../usr/sbin/setmnt
1735 link path=etc/shutdown target=../usr/sbin/shutdown
1736 link path=etc/sulogin target=../sbin/sulogin
1737 link path=etc/swap target=../usr/sbin/swap
1738 link path=etc/swapadd target=../sbin/swapadd
1739 link path=etc/sysdef target=../usr/sbin/sysdef
1740 link path=etc/tar target=../usr/sbin/tar
1741 link path=etc/telinit target=../sbin/init
1742 link path=etc/uadmin target=../sbin/uadmin
1743 link path=etc/umount target=../sbin/umount
1744 link path=etc/umountall target=../sbin/umountall
1745 link path=etc/utmpx target=../var/adm/utmpx
1746 link path=etc/volcopy target=../usr/sbin/volcopy
1747 link path=etc/wall target=../usr/sbin/wall
1748 link path=etc/whodo target=../usr/sbin/whodo
1749 link path=etc/wtmpx target=../var/adm/wtmpx
1750 link path=sbin/in.mpathd target=../lib/inet/in.mpathd
1751 link path=sbin/jsh target=../usr/bin/ksh93
1752 link path=sbin/pfsh target=../usr/bin/pfexec
1753 link path=sbin/sh target=../usr/bin/$(ARCH32)/ksh93
1754 link path=sbin/su target=../usr/bin/su
1755 link path=usr/adm target=../var/adm
1756 link path=usr/bin/cachefspack target=../lib/fs/cachefs/cachefspack
1757 link path=usr/bin/cachefsstat target=../lib/fs/cachefs/cachefsstat
1758 link path=usr/bin/df target=../sbin/df
1759 link path=usr/bin/jsh target=ksh93
1760 link path=usr/bin/pwconv target=../sbin/pwconv
1761 link path=usr/bin/rmail target=./mail
1762 link path=usr/bin/sh target=$(ARCH32)/ksh93
1763 link path=usr/bin/strclean target=../sbin/strclean
1764 link path=usr/bin/strerr target=../sbin/strerr
1765 link path=usr/bin/sync target=../../sbin/sync
1766 link path=usr/bin/tar target=../sbin/tar
1767 link path=usr/bin/uname target=../../sbin/uname
1768 link path=usr/ccs/bin/m4 target=../../bin/m4
1769 link path=usr/has/bin/jsh target=sh
1770 link path=usr/has/lib/rsh target=../bin/sh
1771 link path=usr/lib/$(ARCH64)/ld.so.1 target=../../../lib/$(ARCH64)/ld.so.1
1772 link path=usr/lib/cron target=../../etc/cron.d
1773 link path=usr/lib/devfsadm/devfsadmd target=../../sbin/devfsadm
1774 link path=usr/lib/embedded_su target=../bin/su
```



```
1775 link path=usr/lib/fs/dev/mount target=../../../../etc/fs/dev/mount
1776 link path=usr/lib/fs/hsfs/mount target=../../../../etc/fs/hsfs/mount
1777 link path=usr/lib/fs/ufs/mount target=../../../../etc/fs/ufs/mount
1778 link path=usr/lib/inet/in.mpathd target=../../../lib/inet/in.mpathd
1779 link path=usr/lib/ld.so.1 target=../../lib/ld.so.1
1780 link path=usr/lib/locale/POSIX target=./C
1781 link path=usr/lib/rsh target=../bin/ksh93
1782 link path=usr/lib/secure/32 target=.
1783 link path=usr/lib/secure/64 target=$(ARCH64)
1784 link path=usr/mail target=../var/mail
1785 link path=usr/net/nls/listen target=../../lib/saf/listen
1786 link path=usr/net/nls/nlps_server target=../../lib/saf/nlps_server
1787 link path=usr/news target=../var/news
1788 link path=usr/preserve target=../var/preserve
1789 link path=usr/pub target=./share/lib/pub
1790 link path=usr/sbin/autopush target=../../sbin/autopush
1791 link path=usr/sbin/bootadm target=../../sbin/bootadm
1792 link path=usr/sbin/cachefslog target=../lib/fs/cachefs/cachefslog
1793 link path=usr/sbin/cachefswssize target=../lib/fs/cachefs/cachefswssize
1794 link path=usr/sbin/cfsadmin target=../lib/fs/cachefs/cfsadmin
1795 link path=usr/sbin/cryptoadm target=../../sbin/cryptoadm
1796 link path=usr/sbin/dcopy target=./clri
1797 link path=usr/sbin/devnm target=./df
1798 link path=usr/sbin/dladm target=../../sbin/dladm
1799 link path=usr/sbin/dlstat target=../../sbin/dlstat
1800 link path=usr/sbin/edquota target=../lib/fs/ufs/edquota
1801 link path=usr/sbin/fdisk target=../../sbin/fdisk
1802 link path=usr/sbin/fiocompress target=../../sbin/fiocompress
1803 link path=usr/sbin/flowadm target=../../sbin/flowadm
1804 link path=usr/sbin/flowstat target=../../sbin/flowstat
1805 link path=usr/sbin/fsdb target=./clri
1806 link path=usr/sbin/fsirand target=../lib/fs/ufs/fsirand
1807 link path=usr/sbin/fssnap target=./clri
1808 link path=usr/sbin/hostconfig target=../../sbin/hostconfig
1809 link path=usr/sbin/ifconfig target=../../sbin/ifconfig
1810 link path=usr/sbin/inetd target=../lib/inet/inetd
1811 link path=usr/sbin/init target=../../sbin/init
1812 $(i386_ONLY)link path=usr/sbin/installgrub target=../../sbin/installgrub
1813 link path=usr/sbin/ipadm target=../../sbin/ipadm
1814 link path=usr/sbin/ipmpstat target=../../sbin/ipmpstat
1815 link path=usr/sbin/labelit target=./clri
1816 link path=usr/sbin/lockfs target=../lib/fs/ufs/lockfs
1817 link path=usr/sbin/mkfs target=./clri
1818 link path=usr/sbin/mount target=../../sbin/mount
1819 link path=usr/sbin/ncheck target=./ff
1820 link path=usr/sbin/newfs target=../lib/fs/ufs/newfs
1821 link path=usr/sbin/quot target=../lib/fs/ufs/quot
1822 link path=usr/sbin/quota target=../lib/fs/ufs/quota
1823 link path=usr/sbin/quotacheck target=../lib/fs/ufs/quotacheck
1824 link path=usr/sbin/quotaoff target=../lib/fs/ufs/quotaoff
1825 link path=usr/sbin/quotaon target=../lib/fs/ufs/quotaon
1826 link path=usr/sbin/repquota target=../lib/fs/ufs/repquota
1827 link path=usr/sbin/route target=../../sbin/route
1828 link path=usr/sbin/routeadm target=../../sbin/routeadm
1829 link path=usr/sbin/sync target=../../sbin/sync
1830 link path=usr/sbin/tunefs target=../lib/fs/ufs/tunefs
1831 link path=usr/sbin/tzreload target=../../sbin/tzreload
1832 link path=usr/sbin/uadmin target=../../sbin/uadmin
1833 link path=usr/sbin/ufsdump target=../lib/fs/ufs/ufsdump
1834 link path=usr/sbin/ufsrestore target=../lib/fs/ufs/ufsrestore
1835 link path=usr/sbin/umount target=../../sbin/umount
1836 link path=usr/spool target=../var/spool
1837 link path=usr/src target=./share/src
1838 link path=usr/tmp target=../var/tmp
1839 link path=var/ld/32 target=.
1840 link path=var/ld/64 target=$(ARCH64)
```

```
1841 #
1842 # The bootadm binary needs the etc/release file.
1843 #
1844 depend fmri=release/name type=require
1845 #
1846 # intrd and others use the illumos-defaulted perl interpreter
1847 #
1848 depend fmri=runtime/perl$(PERL_PKGVERS) type=require
1849 #
1850 # intrd uses sun-solaris Perl modules
1851 #
1852 depend fmri=runtime/perl$(PERL_PKGVERS)/module/sun-solaris type=require
1853 #
1854 # The loadkeys binary needs the keytables.
1855 #
1856 depend fmri=system/data/keyboard/keytables type=require
1857 #
1858 # Depend on terminfo data.
1859 #
1860 depend fmri=system/data/terminfo type=require
1861 #
1862 # Depend on zoneinfo data.
1863 #
1864 depend fmri=system/data/zoneinfo type=require
1865 #
1866 # The mailx binary calls /usr/lib/sendmail provided by mailwrapper
1867 #
1868 depend fmri=system/network/mailwrapper type=require
```

```
**********************************************************
   17590 Wed May 27 19:49:22 2015
new/usr/src/pkg/manifests/consolidation-osnet-osnet-message-files.mf
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
```

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#

# Copyright (c) 2010, Oracle and/or its affiliates. All rights reserved.
# Copyright 2014 Garrett D'Amore <garrett@damore.org>

set name=pkg.fmri \
    value=pkg:/consolidation/osnet/osnet-message-files@$(PKGVERS)
set name=pkg.description \
    value="localizable message files for the OS-Networking consolidation"
set name=pkg.summary value="Localizable ON message files"
set name=info.classification \
    value=org.opensolaris.category.2008:Development/System

#
# This package should not have automated dependencies generated because
# it provides messages only.
#
set name=org.opensolaris.nodepend value=true
set name=variant.arch value=$(ARCH)
dir path=usr group=sys
dir path=usr/lib
dir path=usr/lib/help
dir path=usr/lib/help/auths
dir path=usr/lib/help/auths/locale
dir path=usr/lib/help/profiles
dir path=usr/lib/help/profiles/locale
dir path=usr/lib/locale
dir path=usr/lib/locale/C
dir path=usr/lib/locale/C/LC_MESSAGES
dir path=usr/lib/locale/C/LC_TIME
dir path=usr/share
dir path=usr/share/lib
dir path=usr/share/lib/locale
dir path=usr/share/lib/locale/com
dir path=usr/share/lib/locale/com/sun
dir path=usr/share/lib/locale/com/sun/admin
dir path=usr/share/lib/locale/com/sun/admin/pm
dir path=usr/share/lib/locale/com/sun/admin/pm/client
dir path=usr/share/lib/locale/com/sun/slp
```



```
file path=usr/lib/help/auths/locale/AllSolAuthsHeader.html
file path=usr/lib/help/auths/locale/AuditHeader.html
file path=usr/lib/help/auths/locale/AuthJobsAdmin.html
file path=usr/lib/help/auths/locale/AuthJobsUser.html
file path=usr/lib/help/auths/locale/AuthProfmgrAssign.html
file path=usr/lib/help/auths/locale/AuthProfmgrDelegate.html
file path=usr/lib/help/auths/locale/AuthProfmgrExecattrWrite.html
file path=usr/lib/help/auths/locale/AuthProfmgrRead.html
file path=usr/lib/help/auths/locale/AuthProfmgrWrite.html
file path=usr/lib/help/auths/locale/AuthReadNDMP.html
file path=usr/lib/help/auths/locale/AuthReadSMB.html
file path=usr/lib/help/auths/locale/AuthRoleAssign.html
file path=usr/lib/help/auths/locale/AuthRoleDelegate.html
file path=usr/lib/help/auths/locale/AuthRoleWrite.html
file path=usr/lib/help/auths/locale/BindStates.html
file path=usr/lib/help/auths/locale/DevAllocHeader.html
file path=usr/lib/help/auths/locale/DevAllocate.html
file path=usr/lib/help/auths/locale/DevCDRW.html
file path=usr/lib/help/auths/locale/DevConfig.html
file path=usr/lib/help/auths/locale/DevGrant.html
file path=usr/lib/help/auths/locale/DevRevoke.html
file path=usr/lib/help/auths/locale/DhcpmgrHeader.html
file path=usr/lib/help/auths/locale/DhcpmgrWrite.html
file path=usr/lib/help/auths/locale/FileChown.html
file path=usr/lib/help/auths/locale/FileHeader.html
file path=usr/lib/help/auths/locale/FileOwner.html
file path=usr/lib/help/auths/locale/HotplugHeader.html
file path=usr/lib/help/auths/locale/HotplugModify.html
file path=usr/lib/help/auths/locale/IdmapRules.html
file path=usr/lib/help/auths/locale/JobHeader.html
file path=usr/lib/help/auths/locale/JobsGrant.html
file path=usr/lib/help/auths/locale/LabelFileDowngrade.html
file path=usr/lib/help/auths/locale/LabelFileUpgrade.html
file path=usr/lib/help/auths/locale/LabelHeader.html
file path=usr/lib/help/auths/locale/LabelPrint.html
file path=usr/lib/help/auths/locale/LabelRange.html
file path=usr/lib/help/auths/locale/LabelServer.html
file path=usr/lib/help/auths/locale/LabelWinDowngrade.html
file path=usr/lib/help/auths/locale/LabelWinNoView.html
file path=usr/lib/help/auths/locale/LabelWinUpgrade.html
file path=usr/lib/help/auths/locale/LinkSecurity.html
file path=usr/lib/help/auths/locale/LoginEnable.html
file path=usr/lib/help/auths/locale/LoginHeader.html
file path=usr/lib/help/auths/locale/LoginRemote.html
file path=usr/lib/help/auths/locale/MailHeader.html
file path=usr/lib/help/auths/locale/MailQueue.html
file path=usr/lib/help/auths/locale/NetworkAutoconfRead.html
file path=usr/lib/help/auths/locale/NetworkAutoconfSelect.html
file path=usr/lib/help/auths/locale/NetworkAutoconfWlan.html
file path=usr/lib/help/auths/locale/NetworkAutoconfWrite.html
file path=usr/lib/help/auths/locale/NetworkHeader.html
file path=usr/lib/help/auths/locale/NetworkILBconf.html
file path=usr/lib/help/auths/locale/NetworkILBenable.html
file path=usr/lib/help/auths/locale/NetworkInterfaceConfig.html
file path=usr/lib/help/auths/locale/NetworkVRRP.html
file path=usr/lib/help/auths/locale/PriAdmin.html
file path=usr/lib/help/auths/locale/PrintAdmin.html
file path=usr/lib/help/auths/locale/PrintCancel.html
file path=usr/lib/help/auths/locale/PrintHeader.html
file path=usr/lib/help/auths/locale/PrintList.html
file path=usr/lib/help/auths/locale/PrintNoBanner.html
file path=usr/lib/help/auths/locale/PrintPs.html
file path=usr/lib/help/auths/locale/PrintUnlabeled.html
file path=usr/lib/help/auths/locale/ProfmgrHeader.html
file path=usr/lib/help/auths/locale/RoleHeader.html
file path=usr/lib/help/auths/locale/SmfAllocate.html
```

```
125 file path=usr/lib/help/auths/locale/SmfAutofsStates.html
126 file path=usr/lib/help/auths/locale/SmfCoreadmStates.html
127 file path=usr/lib/help/auths/locale/SmfCronStates.html
128 file path=usr/lib/help/auths/locale/SmfExAcctFlowStates.html
129 file path=usr/lib/help/auths/locale/SmfExAcctNetStates.html
130 file path=usr/lib/help/auths/locale/SmfExAcctProcessStates.html
131 file path=usr/lib/help/auths/locale/SmfExAcctTaskStates.html
132 file path=usr/lib/help/auths/locale/SmfHeader.html
133 file path=usr/lib/help/auths/locale/SmfILBStates.html
134 file path=usr/lib/help/auths/locale/SmfIPsecStates.html
135 file path=usr/lib/help/auths/locale/SmfIdmapStates.html
136 file path=usr/lib/help/auths/locale/SmfInetdStates.html
137 file path=usr/lib/help/auths/locale/SmfLocationStates.html
138 file path=usr/lib/help/auths/locale/SmfMDNSStates.html
139 file path=usr/lib/help/auths/locale/SmfManageAudit.html
140 file path=usr/lib/help/auths/locale/SmfManageHeader.html
141 file path=usr/lib/help/auths/locale/SmfManageHotplug.html
142 file path=usr/lib/help/auths/locale/SmfManageZFSSnap.html
143 file path=usr/lib/help/auths/locale/SmfModifyAppl.html
144 file path=usr/lib/help/auths/locale/SmfModifyDepend.html
145 file path=usr/lib/help/auths/locale/SmfModifyFramework.html
146 file path=usr/lib/help/auths/locale/SmfModifyHeader.html
147 file path=usr/lib/help/auths/locale/SmfModifyMethod.html
148 file path=usr/lib/help/auths/locale/SmfNADDStates.html
149 file path=usr/lib/help/auths/locale/SmfNDMPStates.html
150 file path=usr/lib/help/auths/locale/SmfNWAMStates.html
151 file path=usr/lib/help/auths/locale/SmfNscdStates.html
152 file path=usr/lib/help/auths/locale/SmfPowerStates.html
153 file path=usr/lib/help/auths/locale/SmfReparseStates.html
154 file path=usr/lib/help/auths/locale/SmfRoutingStates.html
155 file path=usr/lib/help/auths/locale/SmfSMBFSStates.html
156 file path=usr/lib/help/auths/locale/SmfSMBStates.html
157 file path=usr/lib/help/auths/locale/SmfSendmailStates.html
158 file path=usr/lib/help/auths/locale/SmfSshStates.html
159 file path=usr/lib/help/auths/locale/SmfSyslogStates.html
160 file path=usr/lib/help/auths/locale/SmfVRRPStates.html
161 file path=usr/lib/help/auths/locale/SmfValueAudit.html
162 file path=usr/lib/help/auths/locale/SmfValueCoreadm.html
163 file path=usr/lib/help/auths/locale/SmfValueExAcctFlow.html
164 file path=usr/lib/help/auths/locale/SmfValueExAcctNet.html
165 file path=usr/lib/help/auths/locale/SmfValueExAcctProcess.html
166 file path=usr/lib/help/auths/locale/SmfValueExAcctTask.html
167 file path=usr/lib/help/auths/locale/SmfValueFirewall.html
168 file path=usr/lib/help/auths/locale/SmfValueHeader.html
169 file path=usr/lib/help/auths/locale/SmfValueIPsec.html
170 file path=usr/lib/help/auths/locale/SmfValueIdmap.html
171 file path=usr/lib/help/auths/locale/SmfValueInetd.html
172 file path=usr/lib/help/auths/locale/SmfValueMDNS.html
173 file path=usr/lib/help/auths/locale/SmfValueNADD.html
174 file path=usr/lib/help/auths/locale/SmfValueNDMP.html
175 file path=usr/lib/help/auths/locale/SmfValueNWAM.html
176 file path=usr/lib/help/auths/locale/SmfValueProcSec.html
177 #endif /* ! codereview */
178 file path=usr/lib/help/auths/locale/SmfValueRouting.html
179 file path=usr/lib/help/auths/locale/SmfValueSMB.html
180 file path=usr/lib/help/auths/locale/SmfValueVscan.html
181 file path=usr/lib/help/auths/locale/SmfValueVt.html
182 file path=usr/lib/help/auths/locale/SmfVscanStates.html
183 file path=usr/lib/help/auths/locale/SmfVtStates.html
184 file path=usr/lib/help/auths/locale/SmfWpaStates.html
185 file path=usr/lib/help/auths/locale/SysCpuPowerMgmt.html
186 file path=usr/lib/help/auths/locale/SysDate.html
187 file path=usr/lib/help/auths/locale/SysHeader.html
188 file path=usr/lib/help/auths/locale/SysMaintenance.html
189 file path=usr/lib/help/auths/locale/SysPowerMgmtBrightness.html
190 file path=usr/lib/help/auths/locale/SysPowerMgmtHeader.html
```



```
191 file path=usr/lib/help/auths/locale/SysPowerMgmtSuspend.html
192 file path=usr/lib/help/auths/locale/SysPowerMgmtSuspendtoDisk.html
193 file path=usr/lib/help/auths/locale/SysPowerMgmtSuspendtoRAM.html
194 file path=usr/lib/help/auths/locale/SysShutdown.html
195 file path=usr/lib/help/auths/locale/SysSyseventRead.html
196 file path=usr/lib/help/auths/locale/SysSyseventWrite.html
197 file path=usr/lib/help/auths/locale/TNDaemon.html
198 file path=usr/lib/help/auths/locale/TNctl.html
199 file path=usr/lib/help/auths/locale/ValueTND.html
200 file path=usr/lib/help/auths/locale/WifiConfig.html
201 file path=usr/lib/help/auths/locale/WifiWep.html
202 file path=usr/lib/help/auths/locale/ZoneCloneFrom.html
203 file path=usr/lib/help/auths/locale/ZoneHeader.html
204 file path=usr/lib/help/auths/locale/ZoneLogin.html
205 file path=usr/lib/help/auths/locale/ZoneManage.html
206 file path=usr/lib/help/profiles/locale/RtAcctadm.html
207 file path=usr/lib/help/profiles/locale/RtAll.html
208 file path=usr/lib/help/profiles/locale/RtAuditCfg.html
209 file path=usr/lib/help/profiles/locale/RtAuditCtrl.html
210 file path=usr/lib/help/profiles/locale/RtAuditReview.html
211 file path=usr/lib/help/profiles/locale/RtCPUPowerManagement.html
212 file path=usr/lib/help/profiles/locale/RtConsUser.html
213 file path=usr/lib/help/profiles/locale/RtContractObserver.html
214 file path=usr/lib/help/profiles/locale/RtCronMngmnt.html
215 file path=usr/lib/help/profiles/locale/RtCryptoMngmnt.html
216 file path=usr/lib/help/profiles/locale/RtDHCPMngmnt.html
217 file path=usr/lib/help/profiles/locale/RtDatAdmin.html
218 file path=usr/lib/help/profiles/locale/RtDefault.html
219 file path=usr/lib/help/profiles/locale/RtDeviceMngmnt.html
220 file path=usr/lib/help/profiles/locale/RtDeviceSecurity.html
221 file path=usr/lib/help/profiles/locale/RtExAcctFlow.html
222 file path=usr/lib/help/profiles/locale/RtExAcctNet.html
223 file path=usr/lib/help/profiles/locale/RtExAcctProcess.html
224 file path=usr/lib/help/profiles/locale/RtExAcctTask.html
225 file path=usr/lib/help/profiles/locale/RtFTPMngmnt.html
226 file path=usr/lib/help/profiles/locale/RtFileSysMngmnt.html
227 file path=usr/lib/help/profiles/locale/RtFileSysSecurity.html
228 file path=usr/lib/help/profiles/locale/RtHotplugMngmnt.html
229 file path=usr/lib/help/profiles/locale/RtIPFilterMngmnt.html
230 file path=usr/lib/help/profiles/locale/RtIdmapMngmnt.html
231 file path=usr/lib/help/profiles/locale/RtIdmapNameRulesMngmnt.html
232 file path=usr/lib/help/profiles/locale/RtInetdMngmnt.html
233 file path=usr/lib/help/profiles/locale/RtInfoSec.html
234 file path=usr/lib/help/profiles/locale/RtKerberosClntMngmnt.html
235 file path=usr/lib/help/profiles/locale/RtKerberosSrvrMngmnt.html
236 file path=usr/lib/help/profiles/locale/RtLogMngmnt.html
237 file path=usr/lib/help/profiles/locale/RtMailMngmnt.html
238 file path=usr/lib/help/profiles/locale/RtMaintAndRepair.html
239 file path=usr/lib/help/profiles/locale/RtMediaBkup.html
240 file path=usr/lib/help/profiles/locale/RtMediaCtlg.html
241 file path=usr/lib/help/profiles/locale/RtMediaRestore.html
242 file path=usr/lib/help/profiles/locale/RtNDMPMngmnt.html
243 file path=usr/lib/help/profiles/locale/RtNameServiceAdmin.html
244 file path=usr/lib/help/profiles/locale/RtNameServiceSecure.html
245 file path=usr/lib/help/profiles/locale/RtNetAutoconfAdmin.html
246 file path=usr/lib/help/profiles/locale/RtNetAutoconfUser.html
247 file path=usr/lib/help/profiles/locale/RtNetILB.html
248 file path=usr/lib/help/profiles/locale/RtNetIPsec.html
249 file path=usr/lib/help/profiles/locale/RtNetLinkSecure.html
250 file path=usr/lib/help/profiles/locale/RtNetMngmnt.html
251 file path=usr/lib/help/profiles/locale/RtNetObservability.html
252 file path=usr/lib/help/profiles/locale/RtNetSecure.html
253 file path=usr/lib/help/profiles/locale/RtNetVRRP.html
254 file path=usr/lib/help/profiles/locale/RtNetWifiMngmnt.html
255 file path=usr/lib/help/profiles/locale/RtNetWifiSecure.html
256 file path=usr/lib/help/profiles/locale/RtObAccessMngmnt.html
```

```
file path=usr/lib/help/profiles/locale/RtObjectLabelMngmnt.html
file path=usr/lib/help/profiles/locale/RtOperator.html
file path=usr/lib/help/profiles/locale/RtOutsideAccred.html
file path=usr/lib/help/profiles/locale/RtPriAdmin.html
file path=usr/lib/help/profiles/locale/RtPrntAdmin.html
file path=usr/lib/help/profiles/locale/RtProcManagement.html
file path=usr/lib/help/profiles/locale/RtReparseMngmnt.html
file path=usr/lib/help/profiles/locale/RtReservedProfile.html
file path=usr/lib/help/profiles/locale/RtRightsDelegate.html
file path=usr/lib/help/profiles/locale/RtSMBFSMngmnt.html
file path=usr/lib/help/profiles/locale/RtSMBMngmnt.html
file path=usr/lib/help/profiles/locale/RtSoftwareInstall.html
file path=usr/lib/help/profiles/locale/RtSysAdmin.html
file path=usr/lib/help/profiles/locale/RtSysEvMngmnt.html
file path=usr/lib/help/profiles/locale/RtSysPowerMgmt.html
file path=usr/lib/help/profiles/locale/RtSysPowerMgmtBrightness.html
file path=usr/lib/help/profiles/locale/RtSysPowerMgmtSuspend.html
file path=usr/lib/help/profiles/locale/RtSysPowerMgmtSuspendtoDisk.html
file path=usr/lib/help/profiles/locale/RtSysPowerMgmtSuspendtoRAM.html
file path=usr/lib/help/profiles/locale/RtUserMngmnt.html
file path=usr/lib/help/profiles/locale/RtUserSecurity.html
file path=usr/lib/help/profiles/locale/RtVscanMngmnt.html
file path=usr/lib/help/profiles/locale/RtZFSFileSysMngmnt.html
file path=usr/lib/help/profiles/locale/RtZFSStorageMngmnt.html
file path=usr/lib/help/profiles/locale/RtZoneMngmnt.html
file path=usr/lib/help/profiles/locale/RtZoneSecurity.html
file path=usr/lib/locale/C/LC_MESSAGES/AMD.po
file path=usr/lib/locale/C/LC_MESSAGES/DISK.po
file path=usr/lib/locale/C/LC_MESSAGES/FMD.po
file path=usr/lib/locale/C/LC_MESSAGES/FMNOTIFY.po
file path=usr/lib/locale/C/LC_MESSAGES/GMCA.po
file path=usr/lib/locale/C/LC_MESSAGES/INTEL.po
file path=usr/lib/locale/C/LC_MESSAGES/NXGE.po
file path=usr/lib/locale/C/LC_MESSAGES/PCI.po
file path=usr/lib/locale/C/LC_MESSAGES/PCIEX.po
file path=usr/lib/locale/C/LC_MESSAGES/SCA1000.po
file path=usr/lib/locale/C/LC_MESSAGES/SCA500.po
file path=usr/lib/locale/C/LC_MESSAGES/SCF.po
file path=usr/lib/locale/C/LC_MESSAGES/SENSOR.po
file path=usr/lib/locale/C/LC_MESSAGES/SMF.po
file path=usr/lib/locale/C/LC_MESSAGES/STORAGE.po
file path=usr/lib/locale/C/LC_MESSAGES/SUN4.po
file path=usr/lib/locale/C/LC_MESSAGES/SUN4U.po
file path=usr/lib/locale/C/LC_MESSAGES/SUN4V.po
file path=usr/lib/locale/C/LC_MESSAGES/SUNOS.po
file path=usr/lib/locale/C/LC_MESSAGES/SUNW_OST_ADMIN.po group=sys
file path=usr/lib/locale/C/LC_MESSAGES/SUNW_OST_LINFO group=sys
file path=usr/lib/locale/C/LC_MESSAGES/SUNW_OST_NETRPC.po group=sys
file path=usr/lib/locale/C/LC_MESSAGES/SUNW_OST_OSCMD.po group=sys
file path=usr/lib/locale/C/LC_MESSAGES/SUNW_OST_OSLIB.po group=sys
file path=usr/lib/locale/C/LC_MESSAGES/SUNW_OST_SGS.po group=sys
file path=usr/lib/locale/C/LC_MESSAGES/SUNW_OST_SYSOSPAM.po group=sys
file path=usr/lib/locale/C/LC_MESSAGES/SUNW_OST_UCBCMD.po group=sys
file path=usr/lib/locale/C/LC_MESSAGES/SUNW_OST_ZONEINFO.po group=sys
file path=usr/lib/locale/C/LC_MESSAGES/ZFS.po
file path=usr/lib/locale/C/LC_MESSAGES/libast group=sys
file path=usr/lib/locale/C/LC_MESSAGES/libcmd group=sys
file path=usr/lib/locale/C/LC_MESSAGES/libdll group=sys
file path=usr/lib/locale/C/LC_MESSAGES/libshell group=sys
file path=usr/lib/locale/C/LC_MESSAGES/libsum group=sys
file path=usr/lib/locale/C/LC_MESSAGES/magic group=sys
file path=usr/lib/locale/C/LC_MESSAGES/mailx.help group=sys
file path=usr/lib/locale/C/LC_MESSAGES/more.help group=sys
file path=usr/lib/locale/C/LC_MESSAGES/priv_names group=sys
file path=usr/lib/locale/C/LC_MESSAGES/uxlibc.src group=sys
file path=usr/lib/locale/C/LC_TIME/SUNW_OST_OSCMD.po group=sys
```



```
file path=usr/lib/locale/C/LC_TIME/SUNW_OST_OSLIB.po group=sys
file path=usr/share/lib/locale/com/sun/admin/pm/client/pmHelpResources.java \
    group=lp
file path=usr/share/lib/locale/com/sun/admin/pm/client/pmResources.java \
    group=lp
file path=usr/share/lib/locale/com/sun/slp/ClientLib_en.properties group=sys
file path=usr/share/lib/locale/com/sun/slp/Server_en.properties group=sys
legacy pkg=SUNW0on arch=all \
    desc="localizable message files for the OS-Networking consolidation" \
    name="Localizable ON message files" version=11.11,REV=2009.11.10
license cr_Sun license=cr_Sun
license lic_CDDL license=lic_CDDL
```

```
**********************************************************
   12099 Wed May 27 19:49:22 2015
new/usr/src/pkg/manifests/system-extended-system-utilities.mf
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#

#
# Copyright (c) 2010, Oracle and/or its affiliates. All rights reserved.
# Copyright 2012 Nexenta Systems, Inc. All rights reserved.
#

set name=pkg.fmri value=pkg:/system/extended-system-utilities@$(PKGVERS)
set name=pkg.description \
    value="additional UNIX system utilities, including awk, bc, cal, compress, d
set name=pkg.summary value="Extended System Utilities"
set name=info.classification value=org.opensolaris.category.2008:System/Core
set name=variant.arch value=$(ARCH)
dir path=usr group=sys
dir path=usr/bin
$(i386_ONLY)dir path=usr/bin/$(ARCH32)
dir path=usr/bin/$(ARCH64)
dir path=usr/lib
dir path=usr/lib/$(ARCH64)
dir path=usr/lib/adb group=sys
dir path=usr/lib/adb/$(ARCH64) group=sys
dir path=usr/lib/fs group=sys
dir path=usr/lib/fs/pcfs group=sys
dir path=usr/lib/spell
dir path=usr/proc
dir path=usr/proc/bin
dir path=usr/sbin
dir path=usr/sbin/$(ARCH64)
dir path=usr/share
dir path=usr/share/lib
dir path=usr/share/lib/dict
dir path=usr/share/man/man1
dir path=usr/share/man/man1m
$(i386_ONLY)file path=usr/bin/$(ARCH32)/pargs mode=0555
$(i386_ONLY)file path=usr/bin/$(ARCH32)/pcred mode=0555
$(i386_ONLY)file path=usr/bin/$(ARCH32)/pfiles mode=0555
$(i386_ONLY)file path=usr/bin/$(ARCH32)/pflags mode=0555
$(i386_ONLY)file path=usr/bin/$(ARCH32)/pldd mode=0555
$(i386_ONLY)file path=usr/bin/$(ARCH32)/plgrp mode=0555
```



```
$(i386_ONLY)file path=usr/bin/$(ARCH32)/pmadvise mode=0555
$(i386_ONLY)file path=usr/bin/$(ARCH32)/pmap mode=0555
$(i386_ONLY)file path=usr/bin/$(ARCH32)/ppgsz mode=0555
$(i386_ONLY)file path=usr/bin/$(ARCH32)/ppriv mode=0555
$(i386_ONLY)file path=usr/bin/$(ARCH32)/preap mode=0555
$(i386_ONLY)file path=usr/bin/$(ARCH32)/prun mode=0555
$(i386_ONLY)file path=usr/bin/$(ARCH32)/psecflags mode=0555
#endif /* ! codereview */
$(i386_ONLY)file path=usr/bin/$(ARCH32)/psig mode=0555
$(i386_ONLY)file path=usr/bin/$(ARCH32)/pstack mode=0555
$(i386_ONLY)file path=usr/bin/$(ARCH32)/pstop mode=0555
$(i386_ONLY)file path=usr/bin/$(ARCH32)/ptime mode=0555
$(i386_ONLY)file path=usr/bin/$(ARCH32)/ptree mode=0555
$(i386_ONLY)file path=usr/bin/$(ARCH32)/pwait mode=0555
$(i386_ONLY)file path=usr/bin/$(ARCH32)/pwdx mode=0555
$(i386_ONLY)file path=usr/bin/$(ARCH32)/sort mode=0555
file path=usr/bin/$(ARCH64)/pargs mode=0555
file path=usr/bin/$(ARCH64)/pcred mode=0555
file path=usr/bin/$(ARCH64)/pfiles mode=0555
file path=usr/bin/$(ARCH64)/pflags mode=0555
file path=usr/bin/$(ARCH64)/pldd mode=0555
file path=usr/bin/$(ARCH64)/plgrp mode=0555
file path=usr/bin/$(ARCH64)/pmadvise mode=0555
file path=usr/bin/$(ARCH64)/pmap mode=0555
file path=usr/bin/$(ARCH64)/ppgsz mode=0555
file path=usr/bin/$(ARCH64)/ppriv mode=0555
file path=usr/bin/$(ARCH64)/preap mode=0555
file path=usr/bin/$(ARCH64)/prun mode=0555
file path=usr/bin/$(ARCH64)/psecflags mode=0555
#endif /* ! codereview */
file path=usr/bin/$(ARCH64)/psig mode=0555
file path=usr/bin/$(ARCH64)/pstack mode=0555
file path=usr/bin/$(ARCH64)/pstop mode=0555
file path=usr/bin/$(ARCH64)/ptime mode=0555
file path=usr/bin/$(ARCH64)/ptree mode=0555
file path=usr/bin/$(ARCH64)/pwait mode=0555
file path=usr/bin/$(ARCH64)/pwdx mode=0555
file path=usr/bin/$(ARCH64)/sort mode=0555
file path=usr/bin/asa mode=0555
file path=usr/bin/awk mode=0555
file path=usr/bin/banner mode=0555
file path=usr/bin/batch mode=0555
file path=usr/bin/bc mode=0555
file path=usr/bin/bdiff mode=0755
file path=usr/bin/cal mode=0555
file path=usr/bin/calendar mode=0555
file path=usr/bin/col mode=0555
file path=usr/bin/compress mode=0555
file path=usr/bin/csplit mode=0555
file path=usr/bin/dc mode=0555
file path=usr/bin/diff mode=0555
file path=usr/bin/diff3 mode=0555
file path=usr/bin/dircmp mode=0555
file path=usr/bin/dos2unix mode=0555
file path=usr/bin/expand mode=0555
file path=usr/bin/factor mode=0555
file path=usr/bin/kstat mode=0555
file path=usr/bin/last mode=0555
file path=usr/bin/lastcomm mode=0555
file path=usr/bin/lgrpinfo mode=0555
file path=usr/bin/look mode=0755
file path=usr/bin/mkfifo mode=0555
file path=usr/bin/nawk mode=0555
file path=usr/bin/newform mode=0555
file path=usr/bin/news mode=0555
file path=usr/bin/nl mode=0555
```

```
125 file path=usr/bin/pack mode=0555
126 file path=usr/bin/pginfo mode=0555
127 file path=usr/bin/pgstat mode=0555
128 file path=usr/bin/sdiff mode=0555
129 file path=usr/bin/spell mode=0555
130 file path=usr/bin/split mode=0555
131 file path=usr/bin/tcopy mode=0555
132 file path=usr/bin/unexpand mode=0555
133 file path=usr/bin/units mode=0555
134 file path=usr/bin/unix2dos mode=0555
135 file path=usr/bin/unpack mode=0555
136 file path=usr/bin/uudecode group=uucp mode=0555
137 file path=usr/bin/uuencode group=uucp mode=0555
138 file path=usr/bin/yes mode=0555
139 file path=usr/lib/$(ARCH64)/madv.so.1
140 file path=usr/lib/$(ARCH64)/mpss.so.1
141 file path=usr/lib/adb/$(ARCH64)/adbsub.o group=sys
142 file path=usr/lib/adb/adbgen group=sys mode=0755
143 file path=usr/lib/adb/adbgen1 group=sys mode=0755
144 file path=usr/lib/adb/adbgen3 group=sys mode=0755
145 file path=usr/lib/adb/adbgen4 group=sys mode=0755
146 file path=usr/lib/adb/adbsub.o group=sys
147 file path=usr/lib/diff3prog mode=0555
148 file path=usr/lib/fs/pcfs/fsck mode=0555
149 file path=usr/lib/fs/pcfs/fstyp.so.1 mode=0555
150 file path=usr/lib/fs/pcfs/mkfs mode=0555
151 file path=usr/lib/fs/pcfs/mount mode=0555
152 file path=usr/lib/madv.so.1
153 file path=usr/lib/mpss.so.1
154 file path=usr/lib/spell/compress mode=0555
155 file path=usr/lib/spell/hashcheck mode=0555
156 file path=usr/lib/spell/hashmake mode=0555
157 file path=usr/lib/spell/hlista
158 file path=usr/lib/spell/hlistb
159 file path=usr/lib/spell/hstop
160 file path=usr/lib/spell/spellin mode=0555
161 file path=usr/lib/spell/spellprog mode=0555
162 file path=usr/sbin/dmesg mode=0555
163 file path=usr/sbin/projadd group=sys mode=0555
164 file path=usr/sbin/projdel group=sys mode=0555
165 file path=usr/sbin/projmod group=sys mode=0555
166 file path=usr/share/lib/dict/words
167 file path=usr/share/man/man1/asa.1
168 file path=usr/share/man/man1/awk.1
169 file path=usr/share/man/man1/banner.1
170 file path=usr/share/man/man1/bc.1
171 file path=usr/share/man/man1/bdiff.1
172 file path=usr/share/man/man1/cal.1
173 file path=usr/share/man/man1/calendar.1
174 file path=usr/share/man/man1/col.1
175 file path=usr/share/man/man1/compress.1
176 file path=usr/share/man/man1/csplit.1
177 file path=usr/share/man/man1/dc.1
178 file path=usr/share/man/man1/diff.1
179 file path=usr/share/man/man1/diff3.1
180 file path=usr/share/man/man1/dircmp.1
181 file path=usr/share/man/man1/dos2unix.1
182 file path=usr/share/man/man1/expand.1
183 file path=usr/share/man/man1/factor.1
184 file path=usr/share/man/man1/last.1
185 file path=usr/share/man/man1/lastcomm.1
186 file path=usr/share/man/man1/lgrpinfo.1
187 file path=usr/share/man/man1/look.1
188 file path=usr/share/man/man1/madv.so.1.1
189 file path=usr/share/man/man1/mpss.so.1.1
190 file path=usr/share/man/man1/nawk.1
```



```
191 file path=usr/share/man/man1/newform.1
192 file path=usr/share/man/man1/news.1
193 file path=usr/share/man/man1/nl.1
194 file path=usr/share/man/man1/pack.1
195 file path=usr/share/man/man1/pargs.1
196 file path=usr/share/man/man1/plgrp.1
197 file path=usr/share/man/man1/pmadvise.1
198 file path=usr/share/man/man1/pmap.1
199 file path=usr/share/man/man1/ppgsz.1
200 file path=usr/share/man/man1/ppriv.1
201 file path=usr/share/man/man1/preap.1
202 file path=usr/share/man/man1/psecflags.1
203 #endif /* ! codereview */
204 file path=usr/share/man/man1/ptree.1
205 file path=usr/share/man/man1/sdiff.1
206 file path=usr/share/man/man1/sort.1
207 file path=usr/share/man/man1/spell.1
208 file path=usr/share/man/man1/split.1
209 file path=usr/share/man/man1/tcopy.1
210 file path=usr/share/man/man1/units.1
211 file path=usr/share/man/man1/unix2dos.1
212 file path=usr/share/man/man1/yes.1
213 file path=usr/share/man/man1m/adbgen.1m
214 file path=usr/share/man/man1m/dmesg.1m
215 file path=usr/share/man/man1m/fsck_pcfs.1m
216 file path=usr/share/man/man1m/kstat.1m
217 file path=usr/share/man/man1m/mkfifo.1m
218 file path=usr/share/man/man1m/mkfs_pcfs.1m
219 file path=usr/share/man/man1m/mount_pcfs.1m
220 file path=usr/share/man/man1m/projadd.1m
221 file path=usr/share/man/man1m/projdel.1m
222 file path=usr/share/man/man1m/projmod.1m
223 hardlink path=usr/bin/oawk target=../../usr/bin/awk
224 hardlink path=usr/bin/pargs target=../../usr/lib/isaexec
225 hardlink path=usr/bin/pcred target=../../usr/lib/isaexec
226 hardlink path=usr/bin/pfiles target=../../usr/lib/isaexec
227 hardlink path=usr/bin/pflags target=../../usr/lib/isaexec
228 hardlink path=usr/bin/pldd target=../../usr/lib/isaexec
229 hardlink path=usr/bin/plgrp target=../../usr/lib/isaexec
230 hardlink path=usr/bin/pmadvise target=../../usr/lib/isaexec
231 hardlink path=usr/bin/pmap target=../../usr/lib/isaexec
232 hardlink path=usr/bin/ppgsz target=../../usr/lib/isaexec
233 hardlink path=usr/bin/ppriv target=../../usr/lib/isaexec
234 hardlink path=usr/bin/preap target=../../usr/lib/isaexec
235 hardlink path=usr/bin/prun target=../../usr/lib/isaexec
236 hardlink path=usr/bin/psecflags target=../../usr/lib/isaexec
237 #endif /* ! codereview */
238 hardlink path=usr/bin/psig target=../../usr/lib/isaexec
239 hardlink path=usr/bin/pstack target=../../usr/lib/isaexec
240 hardlink path=usr/bin/pstop target=../../usr/lib/isaexec
241 hardlink path=usr/bin/ptime target=../../usr/lib/isaexec
242 hardlink path=usr/bin/ptree target=../../usr/lib/isaexec
243 hardlink path=usr/bin/pwait target=../../usr/lib/isaexec
244 hardlink path=usr/bin/pwdx target=../../usr/lib/isaexec
245 hardlink path=usr/bin/sort target=../../usr/lib/isaexec
246 hardlink path=usr/bin/uncompress target=../../usr/bin/compress
247 hardlink path=usr/bin/zcat target=../../usr/bin/compress
248 hardlink path=usr/lib/fs/pcfs/fstyp target=../../../sbin/fstyp
249 legacy pkg=SUNWesu \
250     desc="additional UNIX system utilities, including awk, bc, cal, compress, di
251     name="Extended System Utilities"
252 license cr_Sun license=cr_Sun
253 license lic_CDDL license=lic_CDDL
254 license usr/src/cmd/compress/THIRDPARTYLICENSE \
255     license=usr/src/cmd/compress/THIRDPARTYLICENSE
256 license usr/src/cmd/lastcomm/THIRDPARTYLICENSE \
```

```
257     license=usr/src/cmd/lastcomm/THIRDPARTYLICENSE
258 license usr/src/cmd/look/THIRDPARTYLICENSE \
259     license=usr/src/cmd/look/THIRDPARTYLICENSE
260 license usr/src/cmd/units/THIRDPARTYLICENSE \
261     license=usr/src/cmd/units/THIRDPARTYLICENSE
262 link path=usr/bin/dmesg target=../sbin/dmesg
263 link path=usr/bin/pcat target=./unpack
264 link path=usr/bin/strace target=../sbin/strace
265 link path=usr/dict target=./share/lib/dict
266 link path=usr/proc/bin/pcred target=../../bin/pcred
267 link path=usr/proc/bin/pfiles target=../../bin/pfiles
268 link path=usr/proc/bin/pflags target=../../bin/pflags
269 link path=usr/proc/bin/pldd target=../../bin/pldd
270 link path=usr/proc/bin/pmap target=../../bin/pmap
271 link path=usr/proc/bin/prun target=../../bin/prun
272 link path=usr/proc/bin/psig target=../../bin/psig
273 link path=usr/proc/bin/pstack target=../../bin/pstack
274 link path=usr/proc/bin/pstop target=../../bin/pstop
275 link path=usr/proc/bin/ptime target=../../bin/ptime
276 link path=usr/proc/bin/ptree target=../../bin/ptree
277 link path=usr/proc/bin/pwait target=../../bin/pwait
278 link path=usr/proc/bin/pwdx target=../../bin/pwdx
279 link path=usr/share/man/man1/hashcheck.1 target=spell.1
280 link path=usr/share/man/man1/hashmake.1 target=spell.1
281 link path=usr/share/man/man1/pcat.1 target=pack.1
282 link path=usr/share/man/man1/spellin.1 target=spell.1
283 link path=usr/share/man/man1/uncompress.1 target=compress.1
284 link path=usr/share/man/man1/unexpand.1 target=expand.1
285 link path=usr/share/man/man1/unpack.1 target=pack.1
286 link path=usr/share/man/man1/zcat.1 target=compress.1
287 depend fmri=runtime/perl$(PERL_PKGVERS) type=require
```

```
**********************************************************
   90099 Wed May 27 19:49:23 2015
new/usr/src/pkg/manifests/system-header.mf
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
```

```
   1 #
   2 # CDDL HEADER START
   3 #
   4 # The contents of this file are subject to the terms of the
   5 # Common Development and Distribution License (the "License").
   6 # You may not use this file except in compliance with the License.
   7 #
   8 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
   9 # or http://www.opensolaris.org/os/licensing.
  10 # See the License for the specific language governing permissions
  11 # and limitations under the License.
  12 #
  13 # When distributing Covered Code, include this CDDL HEADER in each
  14 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  15 # If applicable, add the following below this CDDL HEADER, with the
  16 # fields enclosed by brackets "[]" replaced with your own identifying
  17 # information: Portions Copyright [yyyy] [name of copyright owner]
  18 #
  19 # CDDL HEADER END
  20 #

  22 #
  23 # Copyright (c) 2010, Oracle and/or its affiliates. All rights reserved.
  24 # Copyright (c) 2012 by Delphix. All rights reserved.
  25 # Copyright 2012 Nexenta Systems, Inc. All rights reserved.
  26 # Copyright 2014 Garrett D'Amore <garrett@damore.org>
  27 #

  29 set name=pkg.fmri value=pkg:/system/header@$(PKGVERS)
  30 set name=pkg.description \
  31     value="SunOS C/C++ header files for general development of software"
  32 set name=pkg.summary value="SunOS Header Files"
  33 set name=info.classification value=org.opensolaris.category.2008:System/Core
  34 set name=variant.arch value=$(ARCH)
  35 dir path=usr group=sys
  36 dir path=usr/include
  37 $(i386_ONLY)dir path=usr/include/$(ARCH64)
  38 $(i386_ONLY)dir path=usr/include/$(ARCH64)/sys
  39 dir path=usr/include/arpa
  40 dir path=usr/include/asm
  41 dir path=usr/include/ast
  42 dir path=usr/include/bsm
  43 dir path=usr/include/dat
  44 dir path=usr/include/des
  45 dir path=usr/include/gssapi
  46 dir path=usr/include/hal
  47 $(i386_ONLY)dir path=usr/include/ia32
  48 $(i386_ONLY)dir path=usr/include/ia32/sys
  49 dir path=usr/include/inet
  50 dir path=usr/include/inet/kssl
  51 dir path=usr/include/ipp
  52 dir path=usr/include/ipp/ipgpc
  53 dir path=usr/include/iso
  54 dir path=usr/include/kerberosv5
  55 dir path=usr/include/libpolkit
  56 dir path=usr/include/net
  57 dir path=usr/include/netinet
  58 dir path=usr/include/nfs
```



```
  59 dir path=usr/include/protocols
  60 dir path=usr/include/rpc
  61 dir path=usr/include/rpcsvc
  62 dir path=usr/include/sasl
  63 dir path=usr/include/scsi
  64 dir path=usr/include/scsi/plugins
  65 dir path=usr/include/scsi/plugins/ses
  66 dir path=usr/include/scsi/plugins/ses/framework
  67 dir path=usr/include/scsi/plugins/ses/vendor
  68 dir path=usr/include/scsi/plugins/smp
  69 dir path=usr/include/scsi/plugins/smp/engine
  70 dir path=usr/include/scsi/plugins/smp/framework
  71 dir path=usr/include/security
  72 dir path=usr/include/sharefs
  73 dir path=usr/include/sys
  74 dir path=usr/include/sys/av
  75 dir path=usr/include/sys/contract
  76 dir path=usr/include/sys/crypto
  77 dir path=usr/include/sys/dktp
  78 dir path=usr/include/sys/fc4
  79 dir path=usr/include/sys/fm
  80 dir path=usr/include/sys/fm/cpu
  81 dir path=usr/include/sys/fm/fs
  82 dir path=usr/include/sys/fm/io
  83 $(sparc_ONLY)dir path=usr/include/sys/fpu
  84 dir path=usr/include/sys/fs
  85 dir path=usr/include/sys/hotplug
  86 dir path=usr/include/sys/hotplug/pci
  87 dir path=usr/include/sys/ib
  88 dir path=usr/include/sys/ib/adapters
  89 dir path=usr/include/sys/ib/adapters/hermon
  90 dir path=usr/include/sys/ib/adapters/tavor
  91 dir path=usr/include/sys/ib/clients
  92 dir path=usr/include/sys/ib/clients/ibd
  93 dir path=usr/include/sys/ib/clients/of
  94 dir path=usr/include/sys/ib/clients/of/rdma
  95 dir path=usr/include/sys/ib/clients/of/sol_ofs
  96 dir path=usr/include/sys/ib/clients/of/sol_ucma
  97 dir path=usr/include/sys/ib/clients/of/sol_umad
  98 dir path=usr/include/sys/ib/clients/of/sol_uverbs
  99 dir path=usr/include/sys/ib/ibnex
 100 dir path=usr/include/sys/ib/ibtl
 101 dir path=usr/include/sys/ib/ibtl/impl
 102 dir path=usr/include/sys/ib/mgt
 103 dir path=usr/include/sys/ib/mgt/ibmf
 104 dir path=usr/include/sys/iso
 105 dir path=usr/include/sys/lvm
 106 dir path=usr/include/sys/proc
 107 dir path=usr/include/sys/rsm
 108 $(i386_ONLY)dir path=usr/include/sys/sata group=sys
 109 dir path=usr/include/sys/scsi
 110 dir path=usr/include/sys/scsi/adapters
 111 dir path=usr/include/sys/scsi/conf
 112 dir path=usr/include/sys/scsi/generic
 113 dir path=usr/include/sys/scsi/impl
 114 dir path=usr/include/sys/scsi/targets
 115 dir path=usr/include/sys/sysevent
 116 dir path=usr/include/sys/tsol
 117 dir path=usr/include/tsol
 118 dir path=usr/include/uuid
 119 $(sparc_ONLY)dir path=usr/include/v7
 120 $(sparc_ONLY)dir path=usr/include/v7/sys
 121 $(sparc_ONLY)dir path=usr/include/v9
 122 $(sparc_ONLY)dir path=usr/include/v9/sys
 123 dir path=usr/include/vm
 124 dir path=usr/platform group=sys
```

```
125 $(sparc_ONLY)dir path=usr/platform/SUNW,A70 group=sys
126 $(sparc_ONLY)dir path=usr/platform/SUNW,Netra-CP2300 group=sys
127 $(sparc_ONLY)dir path=usr/platform/SUNW,Netra-CP2300/include
128 $(sparc_ONLY)dir path=usr/platform/SUNW,Netra-CP3010 group=sys
129 $(sparc_ONLY)dir path=usr/platform/SUNW,Netra-CP3010/include
130 $(sparc_ONLY)dir path=usr/platform/SUNW,Netra-T12 group=sys
131 $(sparc_ONLY)dir path=usr/platform/SUNW,Netra-T4 group=sys
132 $(sparc_ONLY)dir path=usr/platform/SUNW,SPARC-Enterprise group=sys
133 $(sparc_ONLY)dir path=usr/platform/SUNW,Serverblade1 group=sys
134 $(sparc_ONLY)dir path=usr/platform/SUNW,Sun-Blade-100 group=sys
135 $(sparc_ONLY)dir path=usr/platform/SUNW,Sun-Blade-1000 group=sys
136 $(sparc_ONLY)dir path=usr/platform/SUNW,Sun-Blade-1500 group=sys
137 $(sparc_ONLY)dir path=usr/platform/SUNW,Sun-Blade-2500 group=sys
138 $(sparc_ONLY)dir path=usr/platform/SUNW,Sun-Fire group=sys
139 $(sparc_ONLY)dir path=usr/platform/SUNW,Sun-Fire-15000 group=sys
140 $(sparc_ONLY)dir path=usr/platform/SUNW,Sun-Fire-280R group=sys
141 $(sparc_ONLY)dir path=usr/platform/SUNW,Sun-Fire-480R group=sys
142 $(sparc_ONLY)dir path=usr/platform/SUNW,Sun-Fire-880 group=sys
143 $(sparc_ONLY)dir path=usr/platform/SUNW,Sun-Fire-V215 group=sys
144 $(sparc_ONLY)dir path=usr/platform/SUNW,Sun-Fire-V240 group=sys
145 $(sparc_ONLY)dir path=usr/platform/SUNW,Sun-Fire-V250 group=sys
146 $(sparc_ONLY)dir path=usr/platform/SUNW,Sun-Fire-V440 group=sys
147 $(sparc_ONLY)dir path=usr/platform/SUNW,Sun-Fire-V445 group=sys
148 $(sparc_ONLY)dir path=usr/platform/SUNW,Sun-Fire-V490 group=sys
149 $(sparc_ONLY)dir path=usr/platform/SUNW,Sun-Fire-V890 group=sys
150 $(sparc_ONLY)dir path=usr/platform/SUNW,Ultra-2 group=sys
151 $(sparc_ONLY)dir path=usr/platform/SUNW,Ultra-250 group=sys
152 $(sparc_ONLY)dir path=usr/platform/SUNW,Ultra-4 group=sys
153 $(sparc_ONLY)dir path=usr/platform/SUNW,Ultra-Enterprise group=sys
154 $(sparc_ONLY)dir path=usr/platform/SUNW,Ultra-Enterprise-10000 group=sys
155 $(sparc_ONLY)dir path=usr/platform/SUNW,UltraSPARC-IIe-NetraCT-40 group=sys
156 $(sparc_ONLY)dir path=usr/platform/SUNW,UltraSPARC-IIe-NetraCT-60 group=sys
157 $(sparc_ONLY)dir path=usr/platform/SUNW,UltraSPARC-IIi-Netract group=sys
158 $(i386_ONLY)dir path=usr/platform/i86pc group=sys
159 $(i386_ONLY)dir path=usr/platform/i86pc/include
160 $(i386_ONLY)dir path=usr/platform/i86pc/include/sys
161 $(i386_ONLY)dir path=usr/platform/i86pc/include/vm
162 $(i386_ONLY)dir path=usr/platform/i86xpv group=sys
163 $(i386_ONLY)dir path=usr/platform/i86xpv/include
164 $(i386_ONLY)dir path=usr/platform/i86xpv/include/sys
165 $(i386_ONLY)dir path=usr/platform/i86xpv/include/vm
166 $(sparc_ONLY)dir path=usr/platform/sun4u group=sys
167 $(sparc_ONLY)dir path=usr/platform/sun4u/include
168 $(sparc_ONLY)dir path=usr/platform/sun4u/include/sys
169 $(sparc_ONLY)dir path=usr/platform/sun4u/include/sys/i2c
170 $(sparc_ONLY)dir path=usr/platform/sun4u/include/sys/i2c/clients
171 $(sparc_ONLY)dir path=usr/platform/sun4u/include/sys/i2c/misc
172 $(sparc_ONLY)dir path=usr/platform/sun4u/include/vm
173 $(sparc_ONLY)dir path=usr/platform/sun4v group=sys
174 $(sparc_ONLY)dir path=usr/platform/sun4v/include
175 $(sparc_ONLY)dir path=usr/platform/sun4v/include/sys
176 $(sparc_ONLY)dir path=usr/platform/sun4v/include/vm
177 dir path=usr/share
178 dir path=usr/share/man
179 dir path=usr/share/man/man3head
180 dir path=usr/share/man/man4
181 dir path=usr/share/man/man5
182 dir path=usr/share/man/man7i
183 dir path=usr/share/src group=sys
184 dir path=usr/share/src/uts
185 $(i386_ONLY)dir path=usr/share/src/uts/i86pc
186 $(i386_ONLY)dir path=usr/share/src/uts/i86xpv
187 $(sparc_ONLY)dir path=usr/share/src/uts/sun4u
188 $(sparc_ONLY)dir path=usr/share/src/uts/sun4v
189 dir path=usr/xpg4
190 dir path=usr/xpg4/include
```



```
191 $(i386_ONLY)file path=usr/include/$(ARCH64)/sys/kdi_regs.h
192 $(i386_ONLY)file path=usr/include/$(ARCH64)/sys/privmregs.h
193 $(i386_ONLY)file path=usr/include/$(ARCH64)/sys/privregs.h
194 file path=usr/include/aio.h
195 file path=usr/include/alloca.h
196 file path=usr/include/apptrace.h
197 file path=usr/include/apptrace_impl.h
198 file path=usr/include/ar.h
199 file path=usr/include/archives.h
200 file path=usr/include/arpa/ftp.h
201 file path=usr/include/arpa/inet.h
202 file path=usr/include/arpa/nameser.h
203 file path=usr/include/arpa/nameser_compat.h
204 file path=usr/include/arpa/telnet.h
205 file path=usr/include/arpa/tftp.h
206 $(i386_ONLY)file path=usr/include/asm/atomic.h
207 $(i386_ONLY)file path=usr/include/asm/bitmap.h
208 $(i386_ONLY)file path=usr/include/asm/byteorder.h
209 $(i386_ONLY)file path=usr/include/asm/clock.h
210 $(i386_ONLY)file path=usr/include/asm/cpu.h
211 $(i386_ONLY)file path=usr/include/asm/cpuvar.h
212 $(sparc_ONLY)file path=usr/include/asm/flush.h
213 $(i386_ONLY)file path=usr/include/asm/htable.h
214 $(i386_ONLY)file path=usr/include/asm/mmu.h
215 file path=usr/include/asm/sunddi.h
216 file path=usr/include/asm/thread.h
217 file path=usr/include/assert.h
218 file path=usr/include/ast/align.h
219 file path=usr/include/ast/ast.h
220 file path=usr/include/ast/ast_botch.h
221 file path=usr/include/ast/ast_ccode.h
222 file path=usr/include/ast/ast_common.h
223 file path=usr/include/ast/ast_dir.h
224 file path=usr/include/ast/ast_dirent.h
225 file path=usr/include/ast/ast_fcntl.h
226 file path=usr/include/ast/ast_float.h
227 file path=usr/include/ast/ast_fs.h
228 file path=usr/include/ast/ast_getopt.h
229 file path=usr/include/ast/ast_iconv.h
230 file path=usr/include/ast/ast_lib.h
231 file path=usr/include/ast/ast_limits.h
232 file path=usr/include/ast/ast_map.h
233 file path=usr/include/ast/ast_mmap.h
234 file path=usr/include/ast/ast_mode.h
235 file path=usr/include/ast/ast_namval.h
236 file path=usr/include/ast/ast_ndbm.h
237 file path=usr/include/ast/ast_nl_types.h
238 file path=usr/include/ast/ast_param.h
239 file path=usr/include/ast/ast_standards.h
240 file path=usr/include/ast/ast_std.h
241 file path=usr/include/ast/ast_stdio.h
242 file path=usr/include/ast/ast_sys.h
243 file path=usr/include/ast/ast_time.h
244 file path=usr/include/ast/ast_tty.h
245 file path=usr/include/ast/ast_version.h
246 file path=usr/include/ast/ast_vfork.h
247 file path=usr/include/ast/ast_wait.h
248 file path=usr/include/ast/ast_wchar.h
249 file path=usr/include/ast/ast_windows.h
250 file path=usr/include/ast/bytesex.h
251 file path=usr/include/ast/ccode.h
252 file path=usr/include/ast/cdt.h
253 file path=usr/include/ast/cmd.h
254 file path=usr/include/ast/cmdext.h
255 file path=usr/include/ast/debug.h
256 file path=usr/include/ast/dirent.h
```

```
257 file path=usr/include/ast/dlldefs.h
258 file path=usr/include/ast/dt.h
259 file path=usr/include/ast/endian.h
260 file path=usr/include/ast/error.h
261 file path=usr/include/ast/find.h
262 file path=usr/include/ast/fnmatch.h
263 file path=usr/include/ast/fnv.h
264 file path=usr/include/ast/fs3d.h
265 file path=usr/include/ast/fts.h
266 file path=usr/include/ast/ftw.h
267 file path=usr/include/ast/ftwalk.h
268 file path=usr/include/ast/getopt.h
269 file path=usr/include/ast/glob.h
270 file path=usr/include/ast/hash.h
271 file path=usr/include/ast/hashkey.h
272 file path=usr/include/ast/hashpart.h
273 file path=usr/include/ast/history.h
274 file path=usr/include/ast/iconv.h
275 file path=usr/include/ast/ip6.h
276 file path=usr/include/ast/lc.h
277 file path=usr/include/ast/ls.h
278 file path=usr/include/ast/magic.h
279 file path=usr/include/ast/magicid.h
280 file path=usr/include/ast/mc.h
281 file path=usr/include/ast/mime.h
282 file path=usr/include/ast/mnt.h
283 file path=usr/include/ast/modecanon.h
284 file path=usr/include/ast/modex.h
285 file path=usr/include/ast/namval.h
286 file path=usr/include/ast/nl_types.h
287 file path=usr/include/ast/nval.h
288 file path=usr/include/ast/option.h
289 file path=usr/include/ast/preroot.h
290 file path=usr/include/ast/proc.h
291 file path=usr/include/ast/prototyped.h
292 file path=usr/include/ast/re_comp.h
293 file path=usr/include/ast/recfmt.h
294 file path=usr/include/ast/regex.h
295 file path=usr/include/ast/regexp.h
296 file path=usr/include/ast/sfdisc.h
297 file path=usr/include/ast/sfio.h
298 file path=usr/include/ast/sfio_s.h
299 file path=usr/include/ast/sfio_t.h
300 file path=usr/include/ast/shcmd.h
301 file path=usr/include/ast/shell.h
302 file path=usr/include/ast/sig.h
303 file path=usr/include/ast/stack.h
304 file path=usr/include/ast/stak.h
305 file path=usr/include/ast/stdio.h
306 file path=usr/include/ast/stk.h
307 file path=usr/include/ast/sum.h
308 file path=usr/include/ast/swap.h
309 file path=usr/include/ast/tar.h
310 file path=usr/include/ast/times.h
311 file path=usr/include/ast/tm.h
312 file path=usr/include/ast/tmx.h
313 file path=usr/include/ast/tok.h
314 file path=usr/include/ast/tv.h
315 file path=usr/include/ast/usage.h
316 file path=usr/include/ast/vdb.h
317 file path=usr/include/ast/vecargs.h
318 file path=usr/include/ast/vmalloc.h
319 file path=usr/include/ast/wait.h
320 file path=usr/include/ast/wchar.h
321 file path=usr/include/ast/wordexp.h
322 file path=usr/include/atomic.h
```



```
323 file path=usr/include/attr.h
324 file path=usr/include/auth_attr.h
325 file path=usr/include/bsm/adt.h
326 file path=usr/include/bsm/adt_event.h
327 file path=usr/include/bsm/audit.h
328 file path=usr/include/bsm/audit_kernel.h
329 file path=usr/include/bsm/audit_kevents.h
330 file path=usr/include/bsm/audit_record.h
331 file path=usr/include/bsm/audit_uevents.h
332 file path=usr/include/bsm/devices.h
333 file path=usr/include/bsm/libbsm.h
334 file path=usr/include/config_admin.h
335 file path=usr/include/cpio.h
336 file path=usr/include/crypt.h
337 file path=usr/include/cryptoutil.h
338 file path=usr/include/ctype.h
339 file path=usr/include/curses.h
340 file path=usr/include/dat/dat.h
341 file path=usr/include/dat/dat_error.h
342 file path=usr/include/dat/dat_platform_specific.h
343 file path=usr/include/dat/dat_redirection.h
344 file path=usr/include/dat/dat_registry.h
345 file path=usr/include/dat/dat_vendor_specific.h
346 file path=usr/include/dat/udat.h
347 file path=usr/include/dat/udat_config.h
348 file path=usr/include/dat/udat_redirection.h
349 file path=usr/include/dat/udat_vendor_specific.h
350 file path=usr/include/deflt.h
351 file path=usr/include/des/des.h
352 file path=usr/include/des/desdata.h
353 file path=usr/include/des/softdes.h
354 file path=usr/include/device_info.h
355 file path=usr/include/devid.h
356 file path=usr/include/devmgmt.h
357 file path=usr/include/devpoll.h
358 file path=usr/include/dial.h
359 file path=usr/include/dirent.h
360 file path=usr/include/dlfcn.h
361 file path=usr/include/door.h
362 file path=usr/include/elf.h
363 file path=usr/include/err.h
364 file path=usr/include/errno.h
365 file path=usr/include/eti.h
366 file path=usr/include/euc.h
367 file path=usr/include/exacct.h
368 file path=usr/include/exacct_impl.h
369 file path=usr/include/exec_attr.h
370 file path=usr/include/execinfo.h
371 file path=usr/include/fatal.h
372 file path=usr/include/fcntl.h
373 file path=usr/include/float.h
374 file path=usr/include/fmtmsg.h
375 file path=usr/include/fnmatch.h
376 file path=usr/include/form.h
377 file path=usr/include/ftw.h
378 file path=usr/include/gelf.h
379 file path=usr/include/getopt.h
380 file path=usr/include/getwidth.h
381 file path=usr/include/glob.h
382 file path=usr/include/grp.h
383 file path=usr/include/gssapi/gssapi.h
384 file path=usr/include/gssapi/gssapi_ext.h
385 file path=usr/include/hal/libhal-storage.h
386 file path=usr/include/hal/libhal.h
387 $(i386_ONLY)file path=usr/include/ia32/sys/asm_linkage.h
388 $(i386_ONLY)file path=usr/include/ia32/sys/kdi_regs.h
```

```
389 $(i386_ONLY)file path=usr/include/ia32/sys/machtypes.h
390 $(i386_ONLY)file path=usr/include/ia32/sys/privmregs.h
391 $(i386_ONLY)file path=usr/include/ia32/sys/privregs.h
392 $(i386_ONLY)file path=usr/include/ia32/sys/psw.h
393 $(i386_ONLY)file path=usr/include/ia32/sys/pte.h
394 $(i386_ONLY)file path=usr/include/ia32/sys/reg.h
395 $(i386_ONLY)file path=usr/include/ia32/sys/stack.h
396 $(i386_ONLY)file path=usr/include/ia32/sys/trap.h
397 $(i386_ONLY)file path=usr/include/ia32/sys/traptrace.h
398 file path=usr/include/iconv.h
399 file path=usr/include/idmap.h
400 file path=usr/include/ieeefp.h
401 file path=usr/include/ifaddrs.h
402 file path=usr/include/inet/arp.h
403 file path=usr/include/inet/common.h
404 file path=usr/include/inet/ip.h
405 file path=usr/include/inet/ip6.h
406 file path=usr/include/inet/ip6_asp.h
407 file path=usr/include/inet/ip_arp.h
408 file path=usr/include/inet/ip_ftable.h
409 file path=usr/include/inet/ip_if.h
410 file path=usr/include/inet/ip_ire.h
411 file path=usr/include/inet/ip_multi.h
412 file path=usr/include/inet/ip_netinfo.h
413 file path=usr/include/inet/ip_rts.h
414 file path=usr/include/inet/ip_stack.h
415 file path=usr/include/inet/ipclassifier.h
416 file path=usr/include/inet/ipdrop.h
417 file path=usr/include/inet/ipnet.h
418 file path=usr/include/inet/ipp_common.h
419 file path=usr/include/inet/kssl/ksslapi.h
420 file path=usr/include/inet/led.h
421 file path=usr/include/inet/mi.h
422 file path=usr/include/inet/mib2.h
423 file path=usr/include/inet/nd.h
424 file path=usr/include/inet/optcom.h
425 file path=usr/include/inet/sctp_itf.h
426 file path=usr/include/inet/snmpcom.h
427 file path=usr/include/inet/tcp.h
428 file path=usr/include/inet/tcp_sack.h
429 file path=usr/include/inet/tcp_stack.h
430 file path=usr/include/inet/tcp_stats.h
431 file path=usr/include/inet/tunables.h
432 file path=usr/include/inet/wifi_ioctl.h
433 file path=usr/include/inttypes.h
434 file path=usr/include/ipmp.h
435 file path=usr/include/ipmp_admin.h
436 file path=usr/include/ipmp_mpathd.h
437 file path=usr/include/ipmp_query.h
438 file path=usr/include/ipp/ipgpc/ipgpc.h
439 file path=usr/include/ipp/ipp.h
440 file path=usr/include/ipp/ipp_config.h
441 file path=usr/include/ipp/ipp_impl.h
442 file path=usr/include/ipp/ippctl.h
443 file path=usr/include/iso/ctype_iso.h
444 file path=usr/include/iso/limits_iso.h
445 file path=usr/include/iso/locale_iso.h
446 file path=usr/include/iso/setjmp_iso.h
447 file path=usr/include/iso/signal_iso.h
448 file path=usr/include/iso/stdarg_c99.h
449 file path=usr/include/iso/stdarg_iso.h
450 file path=usr/include/iso/stddef_iso.h
451 file path=usr/include/iso/stdio_c99.h
452 file path=usr/include/iso/stdio_iso.h
453 file path=usr/include/iso/stdlib_c99.h
454 file path=usr/include/iso/stdlib_iso.h
```



```
455 file path=usr/include/iso/string_iso.h
456 file path=usr/include/iso/time_iso.h
457 file path=usr/include/iso/wchar_c99.h
458 file path=usr/include/iso/wchar_iso.h
459 file path=usr/include/iso/wctype_iso.h
460 file path=usr/include/iso646.h
461 file path=usr/include/kerberosv5/com_err.h
462 file path=usr/include/kerberosv5/krb5.h
463 file path=usr/include/kerberosv5/mit-sipb-copyright.h
464 file path=usr/include/kerberosv5/mit_copyright.h
465 file path=usr/include/klpd.h
466 file path=usr/include/kmfapi.h
467 file path=usr/include/kmftypes.h
468 file path=usr/include/kstat.h
469 file path=usr/include/kvm.h
470 file path=usr/include/langinfo.h
471 file path=usr/include/lastlog.h
472 file path=usr/include/lber.h
473 file path=usr/include/ldap.h
474 file path=usr/include/libcontract.h
475 file path=usr/include/libctf.h
476 file path=usr/include/libdevice.h
477 file path=usr/include/libdevinfo.h
478 file path=usr/include/libdladm.h
479 file path=usr/include/libdlbridge.h
480 file path=usr/include/libdlib.h
481 file path=usr/include/libdllink.h
482 file path=usr/include/libdlpi.h
483 file path=usr/include/libdlvlan.h
484 file path=usr/include/libelf.h
485 $(i386_ONLY)file path=usr/include/libfdisk.h
486 file path=usr/include/libfstyp.h
487 file path=usr/include/libfstyp_module.h
488 file path=usr/include/libgen.h
489 file path=usr/include/libgrubmgmt.h
490 file path=usr/include/libintl.h
491 file path=usr/include/libipmi.h
492 file path=usr/include/libipp.h
493 file path=usr/include/libnvpair.h
494 file path=usr/include/libnwam.h
495 file path=usr/include/libpolkit/libpolkit.h
496 file path=usr/include/librcm.h
497 file path=usr/include/libscf.h
498 file path=usr/include/libscf_priv.h
499 file path=usr/include/libshare.h
500 file path=usr/include/libsvm.h
501 file path=usr/include/libsysevent.h
502 file path=usr/include/libsysevent_impl.h
503 file path=usr/include/libtsnet.h
504 $(sparc_ONLY)file path=usr/include/libv12n.h
505 file path=usr/include/libw.h
506 file path=usr/include/libzfs.h
507 file path=usr/include/libzfs_core.h
508 file path=usr/include/libzoneinfo.h
509 file path=usr/include/limits.h
510 file path=usr/include/linenum.h
511 file path=usr/include/link.h
512 file path=usr/include/listen.h
513 file path=usr/include/locale.h
514 file path=usr/include/macros.h
515 file path=usr/include/maillock.h
516 file path=usr/include/malloc.h
517 file path=usr/include/md4.h
518 file path=usr/include/md5.h
519 file path=usr/include/mdiox.h
520 file path=usr/include/mdmn_changelog.h
```

```
file path=usr/include/memory.h
file path=usr/include/menu.h
file path=usr/include/meta.h
file path=usr/include/meta_basic.h
file path=usr/include/meta_runtime.h
file path=usr/include/metacl.h
file path=usr/include/metad.h
file path=usr/include/metadyn.h
file path=usr/include/metamed.h
file path=usr/include/metamhd.h
file path=usr/include/mhdx.h
file path=usr/include/mon.h
file path=usr/include/monetary.h
file path=usr/include/mp.h
file path=usr/include/mqueue.h
file path=usr/include/mtmalloc.h
file path=usr/include/nan.h
file path=usr/include/ndbm.h
file path=usr/include/ndpd.h
file path=usr/include/net/af.h
file path=usr/include/net/bridge.h
file path=usr/include/net/if.h
file path=usr/include/net/if_arp.h
file path=usr/include/net/if_dl.h
file path=usr/include/net/if_types.h
file path=usr/include/net/pfkeyv2.h
file path=usr/include/net/pfpolicy.h
file path=usr/include/net/ppp-comp.h
file path=usr/include/net/ppp_defs.h
file path=usr/include/net/pppio.h
file path=usr/include/net/radix.h
file path=usr/include/net/route.h
file path=usr/include/net/trill.h
file path=usr/include/net/vjcompress.h
file path=usr/include/netconfig.h
file path=usr/include/netdb.h
file path=usr/include/netdir.h
file path=usr/include/netinet/arp.h
file path=usr/include/netinet/dhcp.h
file path=usr/include/netinet/dhcp6.h
file path=usr/include/netinet/icmp6.h
file path=usr/include/netinet/icmp_var.h
file path=usr/include/netinet/if_ether.h
file path=usr/include/netinet/igmp.h
file path=usr/include/netinet/igmp_var.h
file path=usr/include/netinet/in.h
file path=usr/include/netinet/in_pcb.h
file path=usr/include/netinet/in_systm.h
file path=usr/include/netinet/in_var.h
file path=usr/include/netinet/ip.h
file path=usr/include/netinet/ip6.h
file path=usr/include/netinet/ip_icmp.h
file path=usr/include/netinet/ip_mroute.h
file path=usr/include/netinet/ip_var.h
file path=usr/include/netinet/pim.h
file path=usr/include/netinet/sctp.h
file path=usr/include/netinet/tcp.h
file path=usr/include/netinet/tcp_debug.h
file path=usr/include/netinet/tcp_fsm.h
file path=usr/include/netinet/tcp_seq.h
file path=usr/include/netinet/tcp_timer.h
file path=usr/include/netinet/tcp_var.h
file path=usr/include/netinet/tcpip.h
file path=usr/include/netinet/udp.h
file path=usr/include/netinet/udp_var.h
file path=usr/include/netinet/vrrp.h
```



```
file path=usr/include/nfs/auth.h
file path=usr/include/nfs/export.h
file path=usr/include/nfs/lm.h
file path=usr/include/nfs/mapid.h
file path=usr/include/nfs/mount.h
file path=usr/include/nfs/nfs.h
file path=usr/include/nfs/nfs4.h
file path=usr/include/nfs/nfs4_attr.h
file path=usr/include/nfs/nfs4_clnt.h
file path=usr/include/nfs/nfs4_db_impl.h
file path=usr/include/nfs/nfs4_idmap_impl.h
file path=usr/include/nfs/nfs4_kprot.h
file path=usr/include/nfs/nfs_acl.h
file path=usr/include/nfs/nfs_clnt.h
file path=usr/include/nfs/nfs_cmd.h
file path=usr/include/nfs/nfs_log.h
file path=usr/include/nfs/nfs_sec.h
file path=usr/include/nfs/nfsid_map.h
file path=usr/include/nfs/nfssys.h
file path=usr/include/nfs/rnode.h
file path=usr/include/nfs/rnode4.h
file path=usr/include/nl_types.h
file path=usr/include/nlist.h
file path=usr/include/note.h
file path=usr/include/nss_common.h
file path=usr/include/nss_dbdefs.h
file path=usr/include/nss_netdir.h
file path=usr/include/nsswitch.h
file path=usr/include/panel.h
file path=usr/include/paths.h
file path=usr/include/pcsample.h
file path=usr/include/pfmt.h
file path=usr/include/pkgdev.h
file path=usr/include/pkginfo.h
file path=usr/include/pkglocs.h
file path=usr/include/pkgstrct.h
file path=usr/include/pkgtrans.h
file path=usr/include/poll.h
file path=usr/include/port.h
file path=usr/include/priv.h
file path=usr/include/proc_service.h
file path=usr/include/procfs.h
file path=usr/include/prof.h
file path=usr/include/prof_attr.h
file path=usr/include/project.h
file path=usr/include/protocols/dumprestore.h
file path=usr/include/protocols/routed.h
file path=usr/include/protocols/rwhod.h
file path=usr/include/protocols/timed.h
file path=usr/include/pthread.h
file path=usr/include/pw.h
file path=usr/include/pwd.h
file path=usr/include/rcm_module.h
file path=usr/include/rctl.h
file path=usr/include/re_comp.h
file path=usr/include/regex.h
file path=usr/include/regexp.h
file path=usr/include/regexpr.h
file path=usr/include/resolv.h
file path=usr/include/rje.h
file path=usr/include/rp_plugin.h
file path=usr/include/rpc/auth.h
file path=usr/include/rpc/auth_des.h
file path=usr/include/rpc/auth_sys.h
file path=usr/include/rpc/auth_unix.h
file path=usr/include/rpc/bootparam.h
```

```
653 file path=usr/include/rpc/clnt.h
654 file path=usr/include/rpc/clnt_soc.h
655 file path=usr/include/rpc/clnt_stat.h
656 file path=usr/include/rpc/des_crypt.h
657 $(sparc_ONLY)file path=usr/include/rpc/ib.h
658 file path=usr/include/rpc/key_prot.h
659 file path=usr/include/rpc/nettype.h
660 file path=usr/include/rpc/pmap_clnt.h
661 file path=usr/include/rpc/pmap_prot.h
662 file path=usr/include/rpc/pmap_prot.x
663 file path=usr/include/rpc/pmap_rmt.h
664 file path=usr/include/rpc/raw.h
665 file path=usr/include/rpc/rpc.h
666 file path=usr/include/rpc/rpc_com.h
667 file path=usr/include/rpc/rpc_msg.h
668 file path=usr/include/rpc/rpc_rdma.h
669 file path=usr/include/rpc/rpc_sztypes.h
670 file path=usr/include/rpc/rpcb_clnt.h
671 file path=usr/include/rpc/rpcb_prot.h
672 file path=usr/include/rpc/rpcb_prot.x
673 file path=usr/include/rpc/rpcent.h
674 file path=usr/include/rpc/rpcsec_gss.h
675 file path=usr/include/rpc/rpcsys.h
676 file path=usr/include/rpc/svc.h
677 file path=usr/include/rpc/svc_auth.h
678 file path=usr/include/rpc/svc_mt.h
679 file path=usr/include/rpc/svc_soc.h
680 file path=usr/include/rpc/types.h
681 file path=usr/include/rpc/xdr.h
682 file path=usr/include/rpcsvc/autofs_prot.h
683 file path=usr/include/rpcsvc/autofs_prot.x
684 file path=usr/include/rpcsvc/bootparam.h
685 file path=usr/include/rpcsvc/bootparam_prot.h
686 file path=usr/include/rpcsvc/bootparam_prot.x
687 file path=usr/include/rpcsvc/dbm.h
688 file path=usr/include/rpcsvc/key_prot.x
689 file path=usr/include/rpcsvc/mount.h
690 file path=usr/include/rpcsvc/mount.x
691 file path=usr/include/rpcsvc/nfs4_prot.h
692 file path=usr/include/rpcsvc/nfs4_prot.x
693 file path=usr/include/rpcsvc/nfs_acl.h
694 file path=usr/include/rpcsvc/nfs_acl.x
695 file path=usr/include/rpcsvc/nfs_prot.h
696 file path=usr/include/rpcsvc/nfs_prot.x
697 file path=usr/include/rpcsvc/nis.h
698 file path=usr/include/rpcsvc/nis.x
699 file path=usr/include/rpcsvc/nis_db.h
700 file path=usr/include/rpcsvc/nis_object.x
701 file path=usr/include/rpcsvc/nislib.h
702 file path=usr/include/rpcsvc/nlm_prot.h
703 file path=usr/include/rpcsvc/nlm_prot.x
704 file path=usr/include/rpcsvc/nsm_addr.h
705 file path=usr/include/rpcsvc/nsm_addr.x
706 file path=usr/include/rpcsvc/rex.h
707 file path=usr/include/rpcsvc/rex.x
708 file path=usr/include/rpcsvc/rpc_sztypes.h
709 file path=usr/include/rpcsvc/rpc_sztypes.x
710 file path=usr/include/rpcsvc/rquota.h
711 file path=usr/include/rpcsvc/rquota.x
712 file path=usr/include/rpcsvc/rstat.h
713 file path=usr/include/rpcsvc/rstat.x
714 file path=usr/include/rpcsvc/rusers.h
715 file path=usr/include/rpcsvc/rusers.x
716 file path=usr/include/rpcsvc/rwall.h
717 file path=usr/include/rpcsvc/rwall.x
718 file path=usr/include/rpcsvc/sm_inter.h
```



```
719 file path=usr/include/rpcsvc/sm_inter.x
720 file path=usr/include/rpcsvc/spray.h
721 file path=usr/include/rpcsvc/spray.x
722 file path=usr/include/rpcsvc/ufs_prot.h
723 file path=usr/include/rpcsvc/ufs_prot.x
724 file path=usr/include/rpcsvc/yp.x
725 file path=usr/include/rpcsvc/yp_prot.h
726 file path=usr/include/rpcsvc/ypclnt.h
727 file path=usr/include/rpcsvc/yppasswd.h
728 file path=usr/include/rpcsvc/ypupd.h
729 file path=usr/include/rsmapi.h
730 file path=usr/include/rtld_db.h
731 file path=usr/include/sac.h
732 file path=usr/include/sasl/prop.h
733 file path=usr/include/sasl/sasl.h
734 file path=usr/include/sasl/saslplug.h
735 file path=usr/include/sasl/saslutil.h
736 file path=usr/include/sched.h
737 file path=usr/include/schedctl.h
738 file path=usr/include/scsi/libscsi.h
739 file path=usr/include/scsi/libses.h
740 file path=usr/include/scsi/libses_plugin.h
741 file path=usr/include/scsi/libsmp.h
742 file path=usr/include/scsi/libsmp_plugin.h
743 file path=usr/include/scsi/plugins/ses/framework/libses.h
744 file path=usr/include/scsi/plugins/ses/framework/ses2.h
745 file path=usr/include/scsi/plugins/ses/framework/ses2_impl.h
746 file path=usr/include/scsi/plugins/ses/vendor/sun.h
747 file path=usr/include/sdp.h
748 file path=usr/include/search.h
749 file path=usr/include/secdb.h
750 file path=usr/include/security/auditd.h
751 file path=usr/include/security/cryptoki.h
752 file path=usr/include/security/pam_appl.h
753 file path=usr/include/security/pam_modules.h
754 file path=usr/include/security/pkcs11.h
755 file path=usr/include/security/pkcs11f.h
756 file path=usr/include/security/pkcs11t.h
757 file path=usr/include/semaphore.h
758 file path=usr/include/setjmp.h
759 file path=usr/include/sgtty.h
760 file path=usr/include/sha1.h
761 file path=usr/include/sha2.h
762 file path=usr/include/shadow.h
763 file path=usr/include/sharefs/share.h
764 file path=usr/include/sharefs/sharefs.h
765 file path=usr/include/sharefs/sharetab.h
766 file path=usr/include/siginfo.h
767 file path=usr/include/signal.h
768 file path=usr/include/sip.h
769 file path=usr/include/smbios.h
770 file path=usr/include/spawn.h
771 $(i386_ONLY)file path=usr/include/stack_unwind.h
772 file path=usr/include/stdarg.h
773 file path=usr/include/stdbool.h
774 file path=usr/include/stddef.h
775 file path=usr/include/stdint.h
776 file path=usr/include/stdio.h
777 file path=usr/include/stdio_ext.h
778 file path=usr/include/stdio_impl.h
779 file path=usr/include/stdio_tag.h
780 file path=usr/include/stdlib.h
781 file path=usr/include/storclass.h
782 file path=usr/include/string.h
783 file path=usr/include/strings.h
784 file path=usr/include/stropts.h
```

```
file path=usr/include/syms.h
file path=usr/include/synch.h
file path=usr/include/sys/acct.h
file path=usr/include/sys/acctctl.h
file path=usr/include/sys/acl.h
file path=usr/include/sys/acl_impl.h
file path=usr/include/sys/acpi_drv.h
file path=usr/include/sys/aio.h
file path=usr/include/sys/aio_impl.h
file path=usr/include/sys/aio_req.h
file path=usr/include/sys/aiocb.h
file path=usr/include/sys/archsystm.h
file path=usr/include/sys/ascii.h
file path=usr/include/sys/asm_linkage.h
file path=usr/include/sys/asynch.h
file path=usr/include/sys/atomic.h
file path=usr/include/sys/attr.h
file path=usr/include/sys/autoconf.h
file path=usr/include/sys/auxv.h
file path=usr/include/sys/auxv_386.h
file path=usr/include/sys/auxv_SPARC.h
file path=usr/include/sys/av/iec61883.h
file path=usr/include/sys/avintr.h
file path=usr/include/sys/avl.h
file path=usr/include/sys/avl_impl.h
file path=usr/include/sys/bitmap.h
file path=usr/include/sys/bitset.h
file path=usr/include/sys/bl.h
file path=usr/include/sys/blkdev.h
file path=usr/include/sys/bofi.h
file path=usr/include/sys/bofi_impl.h
file path=usr/include/sys/bootconf.h
$(i386_ONLY)file path=usr/include/sys/bootregs.h
file path=usr/include/sys/bootstat.h
$(i386_ONLY)file path=usr/include/sys/bootsvcs.h
file path=usr/include/sys/bpp_io.h
file path=usr/include/sys/brand.h
file path=usr/include/sys/buf.h
file path=usr/include/sys/bufmod.h
file path=usr/include/sys/bustypes.h
file path=usr/include/sys/byteorder.h
file path=usr/include/sys/callb.h
file path=usr/include/sys/callo.h
file path=usr/include/sys/cap_util.h
file path=usr/include/sys/ccompile.h
file path=usr/include/sys/cdio.h
file path=usr/include/sys/cis.h
file path=usr/include/sys/cis_handlers.h
file path=usr/include/sys/cis_protos.h
file path=usr/include/sys/cladm.h
file path=usr/include/sys/class.h
file path=usr/include/sys/clconf.h
file path=usr/include/sys/cmlb.h
file path=usr/include/sys/cmn_err.h
$(sparc_ONLY)file path=usr/include/sys/cmpregs.h
file path=usr/include/sys/compress.h
file path=usr/include/sys/condvar.h
file path=usr/include/sys/condvar_impl.h
file path=usr/include/sys/conf.h
file path=usr/include/sys/consdev.h
file path=usr/include/sys/console.h
file path=usr/include/sys/consplat.h
file path=usr/include/sys/contract.h
file path=usr/include/sys/contract/device.h
file path=usr/include/sys/contract/device_impl.h
file path=usr/include/sys/contract/process.h
```



```
file path=usr/include/sys/contract/process_impl.h
file path=usr/include/sys/contract_impl.h
$(i386_ONLY)file path=usr/include/sys/controlregs.h
file path=usr/include/sys/copyops.h
file path=usr/include/sys/core.h
file path=usr/include/sys/corectl.h
file path=usr/include/sys/cpc_impl.h
file path=usr/include/sys/cpc_pcbe.h
file path=usr/include/sys/cpr.h
file path=usr/include/sys/cpu.h
file path=usr/include/sys/cpucaps.h
file path=usr/include/sys/cpucaps_impl.h
file path=usr/include/sys/cpupart.h
file path=usr/include/sys/cpuvar.h
file path=usr/include/sys/crc32.h
file path=usr/include/sys/cred.h
file path=usr/include/sys/cred_impl.h
file path=usr/include/sys/crtctl.h
file path=usr/include/sys/crypto/api.h
file path=usr/include/sys/crypto/common.h
file path=usr/include/sys/crypto/ioctl.h
file path=usr/include/sys/crypto/ioctladmin.h
file path=usr/include/sys/crypto/spi.h
file path=usr/include/sys/cs.h
file path=usr/include/sys/cs_priv.h
file path=usr/include/sys/cs_strings.h
file path=usr/include/sys/cs_stubs.h
file path=usr/include/sys/cs_types.h
file path=usr/include/sys/csiioctl.h
file path=usr/include/sys/ctf.h
file path=usr/include/sys/ctf_api.h
file path=usr/include/sys/ctfs.h
file path=usr/include/sys/ctfs_impl.h
file path=usr/include/sys/ctype.h
file path=usr/include/sys/cyclic.h
file path=usr/include/sys/cyclic_impl.h
file path=usr/include/sys/dacf.h
file path=usr/include/sys/dacf_impl.h
file path=usr/include/sys/damap.h
file path=usr/include/sys/damap_impl.h
file path=usr/include/sys/dc_ki.h
file path=usr/include/sys/ddi.h
file path=usr/include/sys/ddi_hp.h
file path=usr/include/sys/ddi_hp_impl.h
file path=usr/include/sys/ddi_impldefs.h
file path=usr/include/sys/ddi_implfuncs.h
file path=usr/include/sys/ddi_intr.h
file path=usr/include/sys/ddi_intr_impl.h
file path=usr/include/sys/ddi_isa.h
file path=usr/include/sys/ddi_obsolete.h
file path=usr/include/sys/ddi_periodic.h
file path=usr/include/sys/ddidevmap.h
file path=usr/include/sys/ddidmareq.h
file path=usr/include/sys/ddifm.h
file path=usr/include/sys/ddifm_impl.h
file path=usr/include/sys/ddimapreq.h
file path=usr/include/sys/ddipropdefs.h
file path=usr/include/sys/dditypes.h
file path=usr/include/sys/debug.h
$(i386_ONLY)file path=usr/include/sys/debugreg.h
file path=usr/include/sys/des.h
file path=usr/include/sys/devcache.h
file path=usr/include/sys/devcache_impl.h
file path=usr/include/sys/devctl.h
file path=usr/include/sys/devfm.h
file path=usr/include/sys/devid_cache.h
```

```
 917 file path=usr/include/sys/devinfo_impl.h
 918 file path=usr/include/sys/devops.h
 919 file path=usr/include/sys/devpolicy.h
 920 file path=usr/include/sys/devpoll.h
 921 file path=usr/include/sys/dirent.h
 922 file path=usr/include/sys/disp.h
 923 file path=usr/include/sys/dkbad.h
 924 file path=usr/include/sys/dkio.h
 925 file path=usr/include/sys/dklabel.h
 926 $(sparc_ONLY)file path=usr/include/sys/dkmpio.h
 927 $(i386_ONLY)file path=usr/include/sys/dktp/altsctr.h
 928 $(i386_ONLY)file path=usr/include/sys/dktp/cmpkt.h
 929 file path=usr/include/sys/dktp/dadkio.h
 930 file path=usr/include/sys/dktp/fdisk.h
 931 file path=usr/include/sys/dl.h
 932 file path=usr/include/sys/dld.h
 933 file path=usr/include/sys/dlpi.h
 934 file path=usr/include/sys/dls_mgmt.h
 935 $(i386_ONLY)file path=usr/include/sys/dma_engine.h
 936 file path=usr/include/sys/dma_i8237A.h
 937 file path=usr/include/sys/dnlc.h
 938 file path=usr/include/sys/door.h
 939 file path=usr/include/sys/door_data.h
 940 file path=usr/include/sys/door_impl.h
 941 file path=usr/include/sys/dumphdr.h
 942 file path=usr/include/sys/ecppio.h
 943 file path=usr/include/sys/ecppreg.h
 944 file path=usr/include/sys/ecppsys.h
 945 file path=usr/include/sys/ecppvar.h
 946 file path=usr/include/sys/efi_partition.h
 947 file path=usr/include/sys/elf.h
 948 file path=usr/include/sys/elf_386.h
 949 file path=usr/include/sys/elf_SPARC.h
 950 file path=usr/include/sys/elf_amd64.h
 951 file path=usr/include/sys/elf_notes.h
 952 file path=usr/include/sys/elftypes.h
 953 file path=usr/include/sys/epm.h
 954 file path=usr/include/sys/errno.h
 955 file path=usr/include/sys/errorq.h
 956 file path=usr/include/sys/errorq_impl.h
 957 file path=usr/include/sys/esunddi.h
 958 file path=usr/include/sys/ethernet.h
 959 file path=usr/include/sys/euc.h
 960 file path=usr/include/sys/eucioctl.h
 961 file path=usr/include/sys/exacct.h
 962 file path=usr/include/sys/exacct_catalog.h
 963 file path=usr/include/sys/exacct_impl.h
 964 file path=usr/include/sys/exec.h
 965 file path=usr/include/sys/exechdr.h
 966 file path=usr/include/sys/fault.h
 967 file path=usr/include/sys/fbio.h
 968 file path=usr/include/sys/fbuf.h
 969 file path=usr/include/sys/fc4/fc.h
 970 file path=usr/include/sys/fc4/fc_transport.h
 971 file path=usr/include/sys/fc4/fcal.h
 972 file path=usr/include/sys/fc4/fcal_linkapp.h
 973 file path=usr/include/sys/fc4/fcal_transport.h
 974 file path=usr/include/sys/fc4/fcio.h
 975 file path=usr/include/sys/fc4/fcp.h
 976 file path=usr/include/sys/fc4/linkapp.h
 977 file path=usr/include/sys/fcntl.h
 978 file path=usr/include/sys/fdbuffer.h
 979 file path=usr/include/sys/fdio.h
 980 $(sparc_ONLY)file path=usr/include/sys/fdreg.h
 981 $(sparc_ONLY)file path=usr/include/sys/fdvar.h
 982 file path=usr/include/sys/feature_tests.h
```



```
 983 file path=usr/include/sys/fem.h
 984 file path=usr/include/sys/file.h
 985 file path=usr/include/sys/filio.h
 986 file path=usr/include/sys/flock.h
 987 file path=usr/include/sys/flock_impl.h
 988 $(sparc_ONLY)file path=usr/include/sys/fm/cpu/SPARC64-VI.h
 989 $(sparc_ONLY)file path=usr/include/sys/fm/cpu/UltraSPARC-II.h
 990 $(sparc_ONLY)file path=usr/include/sys/fm/cpu/UltraSPARC-III.h
 991 $(sparc_ONLY)file path=usr/include/sys/fm/cpu/UltraSPARC-T1.h
 992 file path=usr/include/sys/fm/fs/zfs.h
 993 file path=usr/include/sys/fm/io/ddi.h
 994 file path=usr/include/sys/fm/io/disk.h
 995 file path=usr/include/sys/fm/io/opl_mc_fm.h
 996 file path=usr/include/sys/fm/io/pci.h
 997 file path=usr/include/sys/fm/io/scsi.h
 998 file path=usr/include/sys/fm/io/sun4upci.h
 999 file path=usr/include/sys/fm/protocol.h
1000 file path=usr/include/sys/fm/util.h
1001 file path=usr/include/sys/fork.h
1002 $(i386_ONLY)file path=usr/include/sys/fp.h
1003 $(sparc_ONLY)file path=usr/include/sys/fpu/fpu_simulator.h
1004 $(sparc_ONLY)file path=usr/include/sys/fpu/fpusystm.h
1005 $(sparc_ONLY)file path=usr/include/sys/fpu/globals.h
1006 $(sparc_ONLY)file path=usr/include/sys/fpu/ieee.h
1007 file path=usr/include/sys/frame.h
1008 file path=usr/include/sys/fs/autofs.h
1009 file path=usr/include/sys/fs/cachefs_dir.h
1010 file path=usr/include/sys/fs/cachefs_dlog.h
1011 file path=usr/include/sys/fs/cachefs_filegrp.h
1012 file path=usr/include/sys/fs/cachefs_fs.h
1013 file path=usr/include/sys/fs/cachefs_fscache.h
1014 file path=usr/include/sys/fs/cachefs_ioctl.h
1015 file path=usr/include/sys/fs/cachefs_log.h
1016 file path=usr/include/sys/fs/decomp.h
1017 file path=usr/include/sys/fs/dv_node.h
1018 file path=usr/include/sys/fs/fifonode.h
1019 file path=usr/include/sys/fs/hsfs_isospec.h
1020 file path=usr/include/sys/fs/hsfs_node.h
1021 file path=usr/include/sys/fs/hsfs_rrip.h
1022 file path=usr/include/sys/fs/hsfs_spec.h
1023 file path=usr/include/sys/fs/hsfs_susp.h
1024 file path=usr/include/sys/fs/lofs_info.h
1025 file path=usr/include/sys/fs/lofs_node.h
1026 file path=usr/include/sys/fs/mntdata.h
1027 file path=usr/include/sys/fs/namenode.h
1028 file path=usr/include/sys/fs/pc_dir.h
1029 file path=usr/include/sys/fs/pc_fs.h
1030 file path=usr/include/sys/fs/pc_label.h
1031 file path=usr/include/sys/fs/pc_node.h
1032 file path=usr/include/sys/fs/pxfs_ki.h
1033 file path=usr/include/sys/fs/sdev_impl.h
1034 file path=usr/include/sys/fs/snode.h
1035 file path=usr/include/sys/fs/swapnode.h
1036 file path=usr/include/sys/fs/tmp.h
1037 file path=usr/include/sys/fs/tmpnode.h
1038 file path=usr/include/sys/fs/udf_inode.h
1039 file path=usr/include/sys/fs/udf_volume.h
1040 file path=usr/include/sys/fs/ufs_acl.h
1041 file path=usr/include/sys/fs/ufs_bio.h
1042 file path=usr/include/sys/fs/ufs_filio.h
1043 file path=usr/include/sys/fs/ufs_fs.h
1044 file path=usr/include/sys/fs/ufs_fsdir.h
1045 file path=usr/include/sys/fs/ufs_inode.h
1046 file path=usr/include/sys/fs/ufs_lockfs.h
1047 file path=usr/include/sys/fs/ufs_log.h
1048 file path=usr/include/sys/fs/ufs_mount.h
```

```
1049 file path=usr/include/sys/fs/ufs_panic.h
1050 file path=usr/include/sys/fs/ufs_prot.h
1051 file path=usr/include/sys/fs/ufs_quota.h
1052 file path=usr/include/sys/fs/ufs_snap.h
1053 file path=usr/include/sys/fs/ufs_trans.h
1054 file path=usr/include/sys/fs/zfs.h
1055 file path=usr/include/sys/fs_reparse.h
1056 file path=usr/include/sys/fs_subr.h
1057 file path=usr/include/sys/fsid.h
1058 $(sparc_ONLY)file path=usr/include/sys/fsr.h
1059 file path=usr/include/sys/fss.h
1060 file path=usr/include/sys/fssnap.h
1061 file path=usr/include/sys/fssnap_if.h
1062 file path=usr/include/sys/fsspriocntl.h
1063 file path=usr/include/sys/fstyp.h
1064 file path=usr/include/sys/ftrace.h
1065 file path=usr/include/sys/fx.h
1066 file path=usr/include/sys/fxpriocntl.h
1067 file path=usr/include/sys/gfs.h
1068 file path=usr/include/sys/gld.h
1069 file path=usr/include/sys/gldpriv.h
1070 file path=usr/include/sys/group.h
1071 file path=usr/include/sys/hdio.h
1072 file path=usr/include/sys/hook.h
1073 file path=usr/include/sys/hook_event.h
1074 file path=usr/include/sys/hook_impl.h
1075 file path=usr/include/sys/hotplug/hpcsvc.h
1076 file path=usr/include/sys/hotplug/hpctrl.h
1077 file path=usr/include/sys/hotplug/pci/pcicfg.h
1078 file path=usr/include/sys/hotplug/pci/pcihp.h
1079 file path=usr/include/sys/hwconf.h
1080 $(i386_ONLY)file path=usr/include/sys/hypervisor.h
1081 $(i386_ONLY)file path=usr/include/sys/i8272A.h
1082 file path=usr/include/sys/ia.h
1083 file path=usr/include/sys/iapriocntl.h
1084 file path=usr/include/sys/ib/adapters/hermon/hermon_ioctl.h
1085 file path=usr/include/sys/ib/adapters/mlnx_umap.h
1086 file path=usr/include/sys/ib/adapters/tavor/tavor_ioctl.h
1087 file path=usr/include/sys/ib/clients/ibd/ibd.h
1088 file path=usr/include/sys/ib/clients/of/ofa_solaris.h
1089 file path=usr/include/sys/ib/clients/of/ofed_kernel.h
1090 file path=usr/include/sys/ib/clients/of/rdma/ib_addr.h
1091 file path=usr/include/sys/ib/clients/of/rdma/ib_user_mad.h
1092 file path=usr/include/sys/ib/clients/of/rdma/ib_user_sa.h
1093 file path=usr/include/sys/ib/clients/of/rdma/ib_user_verbs.h
1094 file path=usr/include/sys/ib/clients/of/rdma/ib_verbs.h
1095 file path=usr/include/sys/ib/clients/of/rdma/rdma_cm.h
1096 file path=usr/include/sys/ib/clients/of/rdma/rdma_user_cm.h
1097 file path=usr/include/sys/ib/clients/of/sol_ofs/sol_cma.h
1098 file path=usr/include/sys/ib/clients/of/sol_ofs/sol_ib_cma.h
1099 file path=usr/include/sys/ib/clients/of/sol_ofs/sol_kverb_impl.h
1100 file path=usr/include/sys/ib/clients/of/sol_ofs/sol_ofs_common.h
1101 file path=usr/include/sys/ib/clients/of/sol_ucma/sol_rdma_user_cm.h
1102 file path=usr/include/sys/ib/clients/of/sol_ucma/sol_ucma.h
1103 file path=usr/include/sys/ib/clients/of/sol_umad/sol_umad.h
1104 file path=usr/include/sys/ib/clients/of/sol_uverbs/sol_uverbs.h
1105 file path=usr/include/sys/ib/clients/of/sol_uverbs/sol_uverbs2ucma.h
1106 file path=usr/include/sys/ib/clients/of/sol_uverbs/sol_uverbs_comp.h
1107 file path=usr/include/sys/ib/clients/of/sol_uverbs/sol_uverbs_event.h
1108 file path=usr/include/sys/ib/clients/of/sol_uverbs/sol_uverbs_hca.h
1109 file path=usr/include/sys/ib/clients/of/sol_uverbs/sol_uverbs_qp.h
1110 file path=usr/include/sys/ib/ib_pkt_hdrs.h
1111 file path=usr/include/sys/ib/ib_types.h
1112 file path=usr/include/sys/ib/ibnex/ibnex_devctl.h
1113 file path=usr/include/sys/ib/ibtl/ibci.h
1114 file path=usr/include/sys/ib/ibtl/ibti.h
```



```
1115 file path=usr/include/sys/ib/ibtl/ibti_cm.h
1116 file path=usr/include/sys/ib/ibtl/ibti_common.h
1117 file path=usr/include/sys/ib/ibtl/ibtl_ci_types.h
1118 file path=usr/include/sys/ib/ibtl/ibtl_status.h
1119 file path=usr/include/sys/ib/ibtl/ibtl_types.h
1120 file path=usr/include/sys/ib/ibtl/ibvti.h
1121 file path=usr/include/sys/ib/ibtl/impl/ibtl_util.h
1122 file path=usr/include/sys/ib/mgt/ib_dm_attr.h
1123 file path=usr/include/sys/ib/mgt/ib_mad.h
1124 file path=usr/include/sys/ib/mgt/ibmf/ibmf.h
1125 file path=usr/include/sys/ib/mgt/ibmf/ibmf_msg.h
1126 file path=usr/include/sys/ib/mgt/ibmf/ibmf_saa.h
1127 file path=usr/include/sys/ib/mgt/ibmf/ibmf_utils.h
1128 file path=usr/include/sys/ib/mgt/sa_recs.h
1129 file path=usr/include/sys/ib/mgt/sm_attr.h
1130 file path=usr/include/sys/ibpart.h
1131 file path=usr/include/sys/id32.h
1132 file path=usr/include/sys/id_space.h
1133 file path=usr/include/sys/idmap.h
1134 file path=usr/include/sys/inline.h
1135 file path=usr/include/sys/instance.h
1136 file path=usr/include/sys/int_const.h
1137 file path=usr/include/sys/int_fmtio.h
1138 file path=usr/include/sys/int_limits.h
1139 file path=usr/include/sys/int_types.h
1140 file path=usr/include/sys/inttypes.h
1141 file path=usr/include/sys/ioccom.h
1142 file path=usr/include/sys/ioctl.h
1143 $(i386_ONLY)file path=usr/include/sys/iommulib.h
1144 file path=usr/include/sys/ipc.h
1145 file path=usr/include/sys/ipc_impl.h
1146 file path=usr/include/sys/ipc_rctl.h
1147 file path=usr/include/sys/isa_defs.h
1148 file path=usr/include/sys/iso/signal_iso.h
1149 file path=usr/include/sys/jioctl.h
1150 file path=usr/include/sys/kbd.h
1151 file path=usr/include/sys/kbdreg.h
1152 file path=usr/include/sys/kbio.h
1153 file path=usr/include/sys/kcpc.h
1154 file path=usr/include/sys/kd.h
1155 file path=usr/include/sys/kdi.h
1156 file path=usr/include/sys/kdi_impl.h
1157 file path=usr/include/sys/kdi_machimpl.h
1158 $(i386_ONLY)file path=usr/include/sys/kdi_regs.h
1159 file path=usr/include/sys/kiconv.h
1160 file path=usr/include/sys/kidmap.h
1161 file path=usr/include/sys/klpd.h
1162 file path=usr/include/sys/klwp.h
1163 file path=usr/include/sys/kmem.h
1164 file path=usr/include/sys/kmem_impl.h
1165 file path=usr/include/sys/kobj.h
1166 file path=usr/include/sys/kobj_impl.h
1167 file path=usr/include/sys/ksocket.h
1168 file path=usr/include/sys/kstat.h
1169 file path=usr/include/sys/kstr.h
1170 file path=usr/include/sys/ksyms.h
1171 file path=usr/include/sys/ksynch.h
1172 file path=usr/include/sys/lc_core.h
1173 file path=usr/include/sys/ldterm.h
1174 file path=usr/include/sys/lgrp.h
1175 file path=usr/include/sys/lgrp_user.h
1176 file path=usr/include/sys/link.h
1177 file path=usr/include/sys/list.h
1178 file path=usr/include/sys/list_impl.h
1179 file path=usr/include/sys/llc1.h
1180 file path=usr/include/sys/loadavg.h
```

```
1181 file path=usr/include/sys/localedef.h
1182 file path=usr/include/sys/lock.h
1183 file path=usr/include/sys/lockfs.h
1184 file path=usr/include/sys/lofi.h
1185 file path=usr/include/sys/log.h
1186 file path=usr/include/sys/logindmux.h
1187 file path=usr/include/sys/lvm/md_basic.h
1188 file path=usr/include/sys/lvm/md_convert.h
1189 file path=usr/include/sys/lvm/md_crc.h
1190 file path=usr/include/sys/lvm/md_hotspares.h
1191 file path=usr/include/sys/lvm/md_mddb.h
1192 file path=usr/include/sys/lvm/md_mdiox.h
1193 file path=usr/include/sys/lvm/md_mhdx.h
1194 file path=usr/include/sys/lvm/md_mirror.h
1195 file path=usr/include/sys/lvm/md_mirror_shared.h
1196 file path=usr/include/sys/lvm/md_names.h
1197 file path=usr/include/sys/lvm/md_notify.h
1198 file path=usr/include/sys/lvm/md_raid.h
1199 file path=usr/include/sys/lvm/md_rename.h
1200 file path=usr/include/sys/lvm/md_sp.h
1201 file path=usr/include/sys/lvm/md_stripe.h
1202 file path=usr/include/sys/lvm/md_trans.h
1203 file path=usr/include/sys/lvm/mdio.h
1204 file path=usr/include/sys/lvm/mdmed.h
1205 file path=usr/include/sys/lvm/mdmn_commd.h
1206 file path=usr/include/sys/lvm/mdvar.h
1207 file path=usr/include/sys/lwp.h
1208 file path=usr/include/sys/lwp_timer_impl.h
1209 file path=usr/include/sys/lwp_upimutex_impl.h
1210 file path=usr/include/sys/mac.h
1211 file path=usr/include/sys/mac_ether.h
1212 file path=usr/include/sys/mac_flow.h
1213 file path=usr/include/sys/mac_provider.h
1214 file path=usr/include/sys/machelf.h
1215 file path=usr/include/sys/machlock.h
1216 file path=usr/include/sys/machsig.h
1217 file path=usr/include/sys/machtypes.h
1218 file path=usr/include/sys/map.h
1219 $(i386_ONLY)file path=usr/include/sys/mc.h
1220 $(i386_ONLY)file path=usr/include/sys/mc_amd.h
1221 $(i386_ONLY)file path=usr/include/sys/mc_intel.h
1222 $(i386_ONLY)file path=usr/include/sys/mca_amd.h
1223 $(i386_ONLY)file path=usr/include/sys/mca_x86.h
1224 file path=usr/include/sys/md4.h
1225 file path=usr/include/sys/md5.h
1226 file path=usr/include/sys/md5_consts.h
1227 file path=usr/include/sys/mdi_impldefs.h
1228 file path=usr/include/sys/mem.h
1229 file path=usr/include/sys/mem_config.h
1230 file path=usr/include/sys/memlist.h
1231 file path=usr/include/sys/mhd.h
1232 file path=usr/include/sys/mii.h
1233 file path=usr/include/sys/miiregs.h
1234 file path=usr/include/sys/mkdev.h
1235 file path=usr/include/sys/mman.h
1236 file path=usr/include/sys/mmapobj.h
1237 file path=usr/include/sys/mntent.h
1238 file path=usr/include/sys/mntio.h
1239 file path=usr/include/sys/mnttab.h
1240 file path=usr/include/sys/modctl.h
1241 file path=usr/include/sys/mode.h
1242 file path=usr/include/sys/model.h
1243 file path=usr/include/sys/modhash.h
1244 file path=usr/include/sys/modhash_impl.h
1245 file path=usr/include/sys/mount.h
1246 file path=usr/include/sys/mouse.h
```



```
1247 file path=usr/include/sys/msacct.h
1248 file path=usr/include/sys/msg.h
1249 file path=usr/include/sys/msg_impl.h
1250 file path=usr/include/sys/msio.h
1251 file path=usr/include/sys/msreg.h
1252 file path=usr/include/sys/mtio.h
1253 file path=usr/include/sys/multidata.h
1254 file path=usr/include/sys/mutex.h
1255 $(i386_ONLY)file path=usr/include/sys/mutex_impl.h
1256 file path=usr/include/sys/nbmlock.h
1257 file path=usr/include/sys/ndi_impldefs.h
1258 file path=usr/include/sys/ndifm.h
1259 file path=usr/include/sys/netconfig.h
1260 file path=usr/include/sys/neti.h
1261 file path=usr/include/sys/netstack.h
1262 file path=usr/include/sys/nexusdefs.h
1263 file path=usr/include/sys/note.h
1264 file path=usr/include/sys/nvpair.h
1265 file path=usr/include/sys/nvpair_impl.h
1266 file path=usr/include/sys/objfs.h
1267 file path=usr/include/sys/objfs_impl.h
1268 file path=usr/include/sys/obpdefs.h
1269 file path=usr/include/sys/old_procfs.h
1270 file path=usr/include/sys/open.h
1271 file path=usr/include/sys/openpromio.h
1272 file path=usr/include/sys/panic.h
1273 file path=usr/include/sys/param.h
1274 file path=usr/include/sys/pathconf.h
1275 file path=usr/include/sys/pathname.h
1276 file path=usr/include/sys/pattr.h
1277 file path=usr/include/sys/pbio.h
1278 file path=usr/include/sys/pcb.h
1279 file path=usr/include/sys/pccard.h
1280 file path=usr/include/sys/pci.h
1281 $(i386_ONLY)file path=usr/include/sys/pcic_reg.h
1282 $(i386_ONLY)file path=usr/include/sys/pcic_var.h
1283 file path=usr/include/sys/pcie.h
1284 file path=usr/include/sys/pcmcia.h
1285 file path=usr/include/sys/pctypes.h
1286 file path=usr/include/sys/pfmod.h
1287 file path=usr/include/sys/pg.h
1288 file path=usr/include/sys/pghw.h
1289 file path=usr/include/sys/physmem.h
1290 $(i386_ONLY)file path=usr/include/sys/pic.h
1291 $(i386_ONLY)file path=usr/include/sys/pit.h
1292 file path=usr/include/sys/pkp_hash.h
1293 file path=usr/include/sys/pm.h
1294 $(i386_ONLY)file path=usr/include/sys/pmem.h
1295 file path=usr/include/sys/policy.h
1296 file path=usr/include/sys/poll.h
1297 file path=usr/include/sys/poll_impl.h
1298 file path=usr/include/sys/pool.h
1299 file path=usr/include/sys/pool_impl.h
1300 file path=usr/include/sys/pool_pset.h
1301 file path=usr/include/sys/port.h
1302 file path=usr/include/sys/port_impl.h
1303 file path=usr/include/sys/port_kernel.h
1304 file path=usr/include/sys/ppmio.h
1305 file path=usr/include/sys/priocntl.h
1306 file path=usr/include/sys/priv.h
1307 file path=usr/include/sys/priv_const.h
1308 file path=usr/include/sys/priv_impl.h
1309 file path=usr/include/sys/priv_names.h
1310 $(i386_ONLY)file path=usr/include/sys/privmregs.h
1311 $(i386_ONLY)file path=usr/include/sys/privregs.h
1312 file path=usr/include/sys/prnio.h
```

```
1313 file path=usr/include/sys/proc.h
1314 file path=usr/include/sys/proc/prdata.h
1315 file path=usr/include/sys/processor.h
1316 file path=usr/include/sys/procfs.h
1317 file path=usr/include/sys/procfs_isa.h
1318 file path=usr/include/sys/procset.h
1319 file path=usr/include/sys/project.h
1320 $(i386_ONLY)file path=usr/include/sys/prom_emul.h
1321 $(i386_ONLY)file path=usr/include/sys/prom_isa.h
1322 $(i386_ONLY)file path=usr/include/sys/prom_plat.h
1323 file path=usr/include/sys/promif.h
1324 file path=usr/include/sys/promimpl.h
1325 file path=usr/include/sys/protosw.h
1326 file path=usr/include/sys/prsystm.h
1327 file path=usr/include/sys/pset.h
1328 file path=usr/include/sys/psw.h
1329 $(i386_ONLY)file path=usr/include/sys/pte.h
1330 file path=usr/include/sys/ptem.h
1331 file path=usr/include/sys/ptms.h
1332 file path=usr/include/sys/ptyvar.h
1333 file path=usr/include/sys/queue.h
1334 file path=usr/include/sys/raidioctl.h
1335 file path=usr/include/sys/ramdisk.h
1336 file path=usr/include/sys/random.h
1337 file path=usr/include/sys/rctl.h
1338 file path=usr/include/sys/rctl_impl.h
1339 file path=usr/include/sys/rds.h
1340 file path=usr/include/sys/reboot.h
1341 file path=usr/include/sys/refstr.h
1342 file path=usr/include/sys/refstr_impl.h
1343 file path=usr/include/sys/reg.h
1344 file path=usr/include/sys/regset.h
1345 file path=usr/include/sys/resource.h
1346 file path=usr/include/sys/rlioctl.h
1347 file path=usr/include/sys/rsm/rsm.h
1348 file path=usr/include/sys/rsm/rsm_common.h
1349 file path=usr/include/sys/rsm/rsmapi_common.h
1350 file path=usr/include/sys/rsm/rsmka_path_int.h
1351 file path=usr/include/sys/rsm/rsmndi.h
1352 file path=usr/include/sys/rsm/rsmpi.h
1353 file path=usr/include/sys/rsm/rsmpi_driver.h
1354 file path=usr/include/sys/rt.h
1355 $(i386_ONLY)file path=usr/include/sys/rtc.h
1356 file path=usr/include/sys/rtpriocntl.h
1357 file path=usr/include/sys/rwlock.h
1358 file path=usr/include/sys/rwlock_impl.h
1359 file path=usr/include/sys/rwstlock.h
1360 file path=usr/include/sys/sad.h
1361 $(i386_ONLY)file path=usr/include/sys/sata/sata_defs.h
1362 $(i386_ONLY)file path=usr/include/sys/sata/sata_hba.h
1363 file path=usr/include/sys/schedctl.h
1364 $(sparc_ONLY)file path=usr/include/sys/scsi/adapters/ifpio.h
1365 file path=usr/include/sys/scsi/adapters/scsi_vhci.h
1366 $(sparc_ONLY)file path=usr/include/sys/scsi/adapters/sfvar.h
1367 file path=usr/include/sys/scsi/conf/autoconf.h
1368 file path=usr/include/sys/scsi/conf/device.h
1369 file path=usr/include/sys/scsi/generic/commands.h
1370 file path=usr/include/sys/scsi/generic/dad_mode.h
1371 file path=usr/include/sys/scsi/generic/inquiry.h
1372 file path=usr/include/sys/scsi/generic/message.h
1373 file path=usr/include/sys/scsi/generic/mode.h
1374 file path=usr/include/sys/scsi/generic/persist.h
1375 file path=usr/include/sys/scsi/generic/sense.h
1376 file path=usr/include/sys/scsi/generic/sff_frames.h
1377 file path=usr/include/sys/scsi/generic/smp_frames.h
1378 file path=usr/include/sys/scsi/generic/status.h
```



```
1379 file path=usr/include/sys/scsi/impl/commands.h
1380 file path=usr/include/sys/scsi/impl/inquiry.h
1381 file path=usr/include/sys/scsi/impl/mode.h
1382 file path=usr/include/sys/scsi/impl/scsi_reset_notify.h
1383 file path=usr/include/sys/scsi/impl/scsi_sas.h
1384 file path=usr/include/sys/scsi/impl/sense.h
1385 file path=usr/include/sys/scsi/impl/services.h
1386 file path=usr/include/sys/scsi/impl/smp_transport.h
1387 file path=usr/include/sys/scsi/impl/spc3_types.h
1388 file path=usr/include/sys/scsi/impl/status.h
1389 file path=usr/include/sys/scsi/impl/transport.h
1390 file path=usr/include/sys/scsi/impl/types.h
1391 file path=usr/include/sys/scsi/impl/uscsi.h
1392 file path=usr/include/sys/scsi/impl/usmp.h
1393 file path=usr/include/sys/scsi/scsi.h
1394 file path=usr/include/sys/scsi/scsi_address.h
1395 file path=usr/include/sys/scsi/scsi_ctl.h
1396 file path=usr/include/sys/scsi/scsi_fm.h
1397 file path=usr/include/sys/scsi/scsi_params.h
1398 file path=usr/include/sys/scsi/scsi_pkt.h
1399 file path=usr/include/sys/scsi/scsi_resource.h
1400 file path=usr/include/sys/scsi/scsi_types.h
1401 file path=usr/include/sys/scsi/scsi_watch.h
1402 file path=usr/include/sys/scsi/targets/sddef.h
1403 file path=usr/include/sys/scsi/targets/ses.h
1404 file path=usr/include/sys/scsi/targets/sesio.h
1405 file path=usr/include/sys/scsi/targets/sgendef.h
1406 file path=usr/include/sys/scsi/targets/smp.h
1407 $(sparc_ONLY)file path=usr/include/sys/scsi/targets/ssddef.h
1408 file path=usr/include/sys/scsi/targets/stdef.h
1409 file path=usr/include/sys/secflags.h
1410 #endif /* ! codereview */
1411 $(i386_ONLY)file path=usr/include/sys/segment.h
1412 $(i386_ONLY)file path=usr/include/sys/segments.h
1413 file path=usr/include/sys/select.h
1414 file path=usr/include/sys/sem.h
1415 file path=usr/include/sys/sem_impl.h
1416 file path=usr/include/sys/sema_impl.h
1417 file path=usr/include/sys/semaphore.h
1418 file path=usr/include/sys/sendfile.h
1419 $(sparc_ONLY)file path=usr/include/sys/ser_async.h
1420 file path=usr/include/sys/ser_sync.h
1421 $(sparc_ONLY)file path=usr/include/sys/ser_zscc.h
1422 file path=usr/include/sys/serializer.h
1423 file path=usr/include/sys/session.h
1424 file path=usr/include/sys/sha1.h
1425 file path=usr/include/sys/sha2.h
1426 file path=usr/include/sys/share.h
1427 file path=usr/include/sys/shm.h
1428 file path=usr/include/sys/shm_impl.h
1429 file path=usr/include/sys/sid.h
1430 file path=usr/include/sys/siginfo.h
1431 file path=usr/include/sys/signal.h
1432 file path=usr/include/sys/sleepq.h
1433 file path=usr/include/sys/smbios.h
1434 file path=usr/include/sys/smbios_impl.h
1435 file path=usr/include/sys/smedia.h
1436 file path=usr/include/sys/sobject.h
1437 $(sparc_ONLY)file path=usr/include/sys/socal_cq_defs.h
1438 $(sparc_ONLY)file path=usr/include/sys/socalio.h
1439 $(sparc_ONLY)file path=usr/include/sys/socalmap.h
1440 $(sparc_ONLY)file path=usr/include/sys/socalreg.h
1441 $(sparc_ONLY)file path=usr/include/sys/socalvar.h
1442 file path=usr/include/sys/socket.h
1443 file path=usr/include/sys/socket_impl.h
1444 file path=usr/include/sys/socket_proto.h
```

```
1445 file path=usr/include/sys/socketvar.h
1446 file path=usr/include/sys/sockio.h
1447 file path=usr/include/sys/spl.h
1448 file path=usr/include/sys/squeue.h
1449 file path=usr/include/sys/squeue_impl.h
1450 file path=usr/include/sys/sservice.h
1451 file path=usr/include/sys/stack.h
1452 file path=usr/include/sys/stat.h
1453 file path=usr/include/sys/stat_impl.h
1454 file path=usr/include/sys/statfs.h
1455 file path=usr/include/sys/statvfs.h
1456 file path=usr/include/sys/stdbool.h
1457 file path=usr/include/sys/stdint.h
1458 file path=usr/include/sys/stermio.h
1459 file path=usr/include/sys/stream.h
1460 file path=usr/include/sys/strft.h
1461 file path=usr/include/sys/strlog.h
1462 file path=usr/include/sys/strmdep.h
1463 file path=usr/include/sys/stropts.h
1464 file path=usr/include/sys/strredir.h
1465 file path=usr/include/sys/strstat.h
1466 file path=usr/include/sys/strsubr.h
1467 file path=usr/include/sys/strsun.h
1468 file path=usr/include/sys/strtty.h
1469 file path=usr/include/sys/sunddi.h
1470 file path=usr/include/sys/sunldi.h
1471 file path=usr/include/sys/sunldi_impl.h
1472 file path=usr/include/sys/sunmdi.h
1473 file path=usr/include/sys/sunndi.h
1474 file path=usr/include/sys/sunpm.h
1475 file path=usr/include/sys/suntpi.h
1476 file path=usr/include/sys/suntty.h
1477 file path=usr/include/sys/swap.h
1478 file path=usr/include/sys/synch.h
1479 file path=usr/include/sys/syscall.h
1480 file path=usr/include/sys/sysconf.h
1481 file path=usr/include/sys/sysconfig.h
1482 file path=usr/include/sys/sysconfig_impl.h
1483 file path=usr/include/sys/sysdc.h
1484 file path=usr/include/sys/sysdc_impl.h
1485 file path=usr/include/sys/sysevent.h
1486 file path=usr/include/sys/sysevent/ap_driver.h
1487 file path=usr/include/sys/sysevent/dev.h
1488 file path=usr/include/sys/sysevent/domain.h
1489 file path=usr/include/sys/sysevent/dr.h
1490 file path=usr/include/sys/sysevent/env.h
1491 file path=usr/include/sys/sysevent/eventdefs.h
1492 file path=usr/include/sys/sysevent/ipmp.h
1493 file path=usr/include/sys/sysevent/pwrctl.h
1494 file path=usr/include/sys/sysevent/svm.h
1495 file path=usr/include/sys/sysevent/vrrp.h
1496 file path=usr/include/sys/sysevent_impl.h
1497 $(i386_ONLY)file path=usr/include/sys/sysi86.h
1498 file path=usr/include/sys/sysinfo.h
1499 file path=usr/include/sys/syslog.h
1500 file path=usr/include/sys/sysmacros.h
1501 file path=usr/include/sys/systeminfo.h
1502 file path=usr/include/sys/systm.h
1503 file path=usr/include/sys/t_kuser.h
1504 file path=usr/include/sys/t_lock.h
1505 file path=usr/include/sys/task.h
1506 file path=usr/include/sys/taskq.h
1507 file path=usr/include/sys/taskq_impl.h
1508 file path=usr/include/sys/telioctl.h
1509 file path=usr/include/sys/termio.h
1510 file path=usr/include/sys/termios.h
```



```
1511 file path=usr/include/sys/termiox.h
1512 file path=usr/include/sys/thread.h
1513 file path=usr/include/sys/ticlts.h
1514 file path=usr/include/sys/ticots.h
1515 file path=usr/include/sys/ticotsord.h
1516 file path=usr/include/sys/tihdr.h
1517 file path=usr/include/sys/time.h
1518 file path=usr/include/sys/time_impl.h
1519 file path=usr/include/sys/time_std_impl.h
1520 file path=usr/include/sys/timeb.h
1521 file path=usr/include/sys/timer.h
1522 file path=usr/include/sys/times.h
1523 file path=usr/include/sys/timex.h
1524 file path=usr/include/sys/timod.h
1525 file path=usr/include/sys/tirdwr.h
1526 file path=usr/include/sys/tiuser.h
1527 file path=usr/include/sys/tl.h
1528 file path=usr/include/sys/tnf.h
1529 file path=usr/include/sys/tnf_com.h
1530 file path=usr/include/sys/tnf_probe.h
1531 file path=usr/include/sys/tnf_writer.h
1532 file path=usr/include/sys/todio.h
1533 file path=usr/include/sys/tpicommon.h
1534 file path=usr/include/sys/trap.h
1535 $(i386_ONLY)file path=usr/include/sys/traptrace.h
1536 file path=usr/include/sys/ts.h
1537 file path=usr/include/sys/tsol/label.h
1538 file path=usr/include/sys/tsol/label_macro.h
1539 file path=usr/include/sys/tsol/priv.h
1540 file path=usr/include/sys/tsol/tndb.h
1541 file path=usr/include/sys/tsol/tsyscall.h
1542 file path=usr/include/sys/tspriocntl.h
1543 $(i386_ONLY)file path=usr/include/sys/tss.h
1544 file path=usr/include/sys/ttcompat.h
1545 file path=usr/include/sys/ttold.h
1546 file path=usr/include/sys/tty.h
1547 file path=usr/include/sys/ttychars.h
1548 file path=usr/include/sys/ttydev.h
1549 $(sparc_ONLY)file path=usr/include/sys/ttymux.h
1550 $(sparc_ONLY)file path=usr/include/sys/ttymuxuser.h
1551 file path=usr/include/sys/tuneable.h
1552 file path=usr/include/sys/turnstile.h
1553 file path=usr/include/sys/types.h
1554 file path=usr/include/sys/types32.h
1555 file path=usr/include/sys/tzfile.h
1556 file path=usr/include/sys/u8_textprep.h
1557 file path=usr/include/sys/uadmin.h
1558 $(i386_ONLY)file path=usr/include/sys/ucode.h
1559 file path=usr/include/sys/ucontext.h
1560 file path=usr/include/sys/uio.h
1561 file path=usr/include/sys/ulimit.h
1562 file path=usr/include/sys/un.h
1563 file path=usr/include/sys/unistd.h
1564 file path=usr/include/sys/user.h
1565 file path=usr/include/sys/ustat.h
1566 file path=usr/include/sys/utime.h
1567 file path=usr/include/sys/utrap.h
1568 file path=usr/include/sys/utsname.h
1569 file path=usr/include/sys/utssys.h
1570 file path=usr/include/sys/uuid.h
1571 file path=usr/include/sys/va_impl.h
1572 file path=usr/include/sys/va_list.h
1573 file path=usr/include/sys/var.h
1574 file path=usr/include/sys/varargs.h
1575 file path=usr/include/sys/vfs.h
1576 file path=usr/include/sys/vfs_opreg.h
```

```
1577 file path=usr/include/sys/vfstab.h
1578 file path=usr/include/sys/videodev2.h
1579 file path=usr/include/sys/visual_io.h
1580 file path=usr/include/sys/vm.h
1581 file path=usr/include/sys/vm_usage.h
1582 file path=usr/include/sys/vmem.h
1583 file path=usr/include/sys/vmem_impl.h
1584 file path=usr/include/sys/vmem_impl_user.h
1585 file path=usr/include/sys/vmparam.h
1586 file path=usr/include/sys/vmsystm.h
1587 file path=usr/include/sys/vnode.h
1588 file path=usr/include/sys/vt.h
1589 file path=usr/include/sys/vtdaemon.h
1590 file path=usr/include/sys/vtoc.h
1591 file path=usr/include/sys/vtrace.h
1592 file path=usr/include/sys/vuid_event.h
1593 file path=usr/include/sys/vuid_queue.h
1594 file path=usr/include/sys/vuid_state.h
1595 file path=usr/include/sys/vuid_store.h
1596 file path=usr/include/sys/vuid_wheel.h
1597 file path=usr/include/sys/wait.h
1598 file path=usr/include/sys/waitq.h
1599 file path=usr/include/sys/watchpoint.h
1600 $(i386_ONLY)file path=usr/include/sys/x86_archext.h
1601 $(i386_ONLY)file path=usr/include/sys/xen_errno.h
1602 file path=usr/include/sys/xti_inet.h
1603 file path=usr/include/sys/xti_osi.h
1604 file path=usr/include/sys/xti_xtiopt.h
1605 file path=usr/include/sys/zcons.h
1606 file path=usr/include/sys/zmod.h
1607 file path=usr/include/sys/zone.h
1608 $(sparc_ONLY)file path=usr/include/sys/zsdev.h
1609 file path=usr/include/sysexits.h
1610 file path=usr/include/syslog.h
1611 file path=usr/include/tar.h
1612 file path=usr/include/tcpd.h
1613 file path=usr/include/term.h
1614 file path=usr/include/termcap.h
1615 file path=usr/include/termio.h
1616 file path=usr/include/termios.h
1617 file path=usr/include/thread.h
1618 file path=usr/include/thread_db.h
1619 file path=usr/include/time.h
1620 file path=usr/include/tiuser.h
1621 file path=usr/include/tsol/label.h
1622 file path=usr/include/tzfile.h
1623 file path=usr/include/ucontext.h
1624 file path=usr/include/ucred.h
1625 file path=usr/include/uid_stp.h
1626 file path=usr/include/ulimit.h
1627 file path=usr/include/umem.h
1628 file path=usr/include/umem_impl.h
1629 file path=usr/include/unctrl.h
1630 file path=usr/include/unistd.h
1631 file path=usr/include/user_attr.h
1632 file path=usr/include/userdefs.h
1633 file path=usr/include/ustat.h
1634 file path=usr/include/utility.h
1635 file path=usr/include/utime.h
1636 file path=usr/include/utmp.h
1637 file path=usr/include/utmpx.h
1638 file path=usr/include/uuid/uuid.h
1639 $(sparc_ONLY)file path=usr/include/v7/sys/machpcb.h
1640 $(sparc_ONLY)file path=usr/include/v7/sys/machtrap.h
1641 $(sparc_ONLY)file path=usr/include/v7/sys/mutex_impl.h
1642 $(sparc_ONLY)file path=usr/include/v7/sys/privregs.h
```



```
1643 $(sparc_ONLY)file path=usr/include/v7/sys/prom_isa.h
1644 $(sparc_ONLY)file path=usr/include/v7/sys/psr.h
1645 $(sparc_ONLY)file path=usr/include/v7/sys/traptrace.h
1646 $(sparc_ONLY)file path=usr/include/v9/sys/asi.h
1647 $(sparc_ONLY)file path=usr/include/v9/sys/machpcb.h
1648 $(sparc_ONLY)file path=usr/include/v9/sys/machtrap.h
1649 $(sparc_ONLY)file path=usr/include/v9/sys/membar.h
1650 $(sparc_ONLY)file path=usr/include/v9/sys/mutex_impl.h
1651 $(sparc_ONLY)file path=usr/include/v9/sys/privregs.h
1652 $(sparc_ONLY)file path=usr/include/v9/sys/prom_isa.h
1653 $(sparc_ONLY)file path=usr/include/v9/sys/psr_compat.h
1654 $(sparc_ONLY)file path=usr/include/v9/sys/vis_simulator.h
1655 file path=usr/include/valtools.h
1656 file path=usr/include/values.h
1657 file path=usr/include/varargs.h
1658 file path=usr/include/vm/anon.h
1659 file path=usr/include/vm/as.h
1660 file path=usr/include/vm/faultcode.h
1661 file path=usr/include/vm/hat.h
1662 file path=usr/include/vm/kpm.h
1663 file path=usr/include/vm/page.h
1664 file path=usr/include/vm/pvn.h
1665 file path=usr/include/vm/rm.h
1666 file path=usr/include/vm/seg.h
1667 file path=usr/include/vm/seg_dev.h
1668 file path=usr/include/vm/seg_enum.h
1669 file path=usr/include/vm/seg_kmem.h
1670 file path=usr/include/vm/seg_kp.h
1671 file path=usr/include/vm/seg_kpm.h
1672 file path=usr/include/vm/seg_map.h
1673 file path=usr/include/vm/seg_spt.h
1674 file path=usr/include/vm/seg_vn.h
1675 file path=usr/include/vm/vpage.h
1676 file path=usr/include/vm/vpm.h
1677 file path=usr/include/volmgt.h
1678 file path=usr/include/wait.h
1679 file path=usr/include/wchar.h
1680 file path=usr/include/wchar_impl.h
1681 file path=usr/include/wctype.h
1682 file path=usr/include/widec.h
1683 file path=usr/include/wordexp.h
1684 file path=usr/include/xlocale.h
1685 file path=usr/include/xti.h
1686 file path=usr/include/xti_inet.h
1687 file path=usr/include/zone.h
1688 file path=usr/include/zonestat.h
1689 $(i386_ONLY)file path=usr/platform/i86pc/include/sys/acpidev.h
1690 $(i386_ONLY)file path=usr/platform/i86pc/include/sys/amd_iommu.h
1691 $(i386_ONLY)file path=usr/platform/i86pc/include/sys/asm_misc.h
1692 $(i386_ONLY)file path=usr/platform/i86pc/include/sys/clock.h
1693 $(i386_ONLY)file path=usr/platform/i86pc/include/sys/cram.h
1694 $(i386_ONLY)file path=usr/platform/i86pc/include/sys/ddi_subrdefs.h
1695 $(i386_ONLY)file path=usr/platform/i86pc/include/sys/debug_info.h
1696 $(i386_ONLY)file path=usr/platform/i86pc/include/sys/fastboot.h
1697 $(i386_ONLY)file path=usr/platform/i86pc/include/sys/mach_mmu.h
1698 $(i386_ONLY)file path=usr/platform/i86pc/include/sys/machclock.h
1699 $(i386_ONLY)file path=usr/platform/i86pc/include/sys/machcpuvar.h
1700 $(i386_ONLY)file path=usr/platform/i86pc/include/sys/machparam.h
1701 $(i386_ONLY)file path=usr/platform/i86pc/include/sys/machprivregs.h
1702 $(i386_ONLY)file path=usr/platform/i86pc/include/sys/machsystm.h
1703 $(i386_ONLY)file path=usr/platform/i86pc/include/sys/machthread.h
1704 $(i386_ONLY)file path=usr/platform/i86pc/include/sys/memnode.h
1705 $(i386_ONLY)file path=usr/platform/i86pc/include/sys/pc_mmu.h
1706 $(i386_ONLY)file path=usr/platform/i86pc/include/sys/psm.h
1707 $(i386_ONLY)file path=usr/platform/i86pc/include/sys/psm_defs.h
1708 $(i386_ONLY)file path=usr/platform/i86pc/include/sys/psm_modctl.h
```

```
$(i386_ONLY)file path=usr/platform/i86pc/include/sys/psm_types.h
$(i386_ONLY)file path=usr/platform/i86pc/include/sys/rm_platter.h
$(i386_ONLY)file path=usr/platform/i86pc/include/sys/sbd_ioctl.h
$(i386_ONLY)file path=usr/platform/i86pc/include/sys/smp_impldefs.h
$(i386_ONLY)file path=usr/platform/i86pc/include/sys/vm_machparam.h
$(i386_ONLY)file path=usr/platform/i86pc/include/sys/x_call.h
$(i386_ONLY)file path=usr/platform/i86pc/include/sys/xc_levels.h
$(i386_ONLY)file path=usr/platform/i86pc/include/sys/xsvc.h
$(i386_ONLY)file path=usr/platform/i86pc/include/vm/hat_i86.h
$(i386_ONLY)file path=usr/platform/i86pc/include/vm/hat_pte.h
$(i386_ONLY)file path=usr/platform/i86pc/include/vm/hment.h
$(i386_ONLY)file path=usr/platform/i86pc/include/vm/htable.h
$(i386_ONLY)file path=usr/platform/i86pc/include/vm/kboot_mmu.h
$(i386_ONLY)file path=usr/platform/i86xpv/include/sys/balloon.h
$(i386_ONLY)file path=usr/platform/i86xpv/include/sys/machprivregs.h
$(i386_ONLY)file path=usr/platform/i86xpv/include/sys/xen_mmu.h
$(i386_ONLY)file path=usr/platform/i86xpv/include/sys/xpv_impl.h
$(i386_ONLY)file path=usr/platform/i86xpv/include/vm/seg_mf.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/ac.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/async.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/cheetahregs.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/cherrystone.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/clock.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/cmp.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/cpc_ultra.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/cpr_impl.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/cpu_impl.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/cpu_sgnblk_defs.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/cvc.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/daktari.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/ddi_subrdefs.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/dvma.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/ecc_kstat.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/eeprom.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/envctrl.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/envctrl_gen.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/envctrl_ue250.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/envctrl_ue450.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/environ.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/errclassify.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/fhc.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/gpio_87317.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/hpc3130_events.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/i2c/clients/hpc3130.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/i2c/clients/i2c_client.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/i2c/clients/lm75.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/i2c/clients/max1617.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/i2c/clients/pcf8591.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/i2c/clients/ssc050.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/i2c/misc/i2c_svc.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/idprom.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/intr.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/intreg.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/iocache.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/iommu.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/ivintr.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/lom_io.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/machasi.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/machclock.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/machcpuvar.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/machparam.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/machsystm.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/machthread.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/mem_cache.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/memlist_plat.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/memnode.h
```



```
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/mmu.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/nexusdebug.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/opl_hwdesc.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/opl_module.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/prom_debug.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/prom_plat.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/pte.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/sbd_ioctl.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/scb.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/scsb_led.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/simmstat.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/spitregs.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/sram.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/starfire.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/sun4asi.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/sysctrl.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/sysioerr.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/sysiosbus.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/tod.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/todmostek.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/trapstat.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/traptrace.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/vis.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/vm_machparam.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/x_call.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/xc_impl.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/sys/zsmach.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/vm/hat_sfmmu.h
$(sparc_ONLY)file path=usr/platform/sun4u/include/vm/mach_sfmmu.h
$(sparc_ONLY)file path=usr/platform/sun4v/include/sys/clock.h
$(sparc_ONLY)file path=usr/platform/sun4v/include/sys/cmp.h
$(sparc_ONLY)file path=usr/platform/sun4v/include/sys/cpc_ultra.h
$(sparc_ONLY)file path=usr/platform/sun4v/include/sys/cpu_sgnblk_defs.h
$(sparc_ONLY)file path=usr/platform/sun4v/include/sys/ddi_subrdefs.h
$(sparc_ONLY)file path=usr/platform/sun4v/include/sys/ds_pri.h
$(sparc_ONLY)file path=usr/platform/sun4v/include/sys/ds_snmp.h
$(sparc_ONLY)file path=usr/platform/sun4v/include/sys/dvma.h
$(sparc_ONLY)file path=usr/platform/sun4v/include/sys/eeprom.h
$(sparc_ONLY)file path=usr/platform/sun4v/include/sys/fcode.h
$(sparc_ONLY)file path=usr/platform/sun4v/include/sys/hsvc.h
$(sparc_ONLY)file path=usr/platform/sun4v/include/sys/hypervisor_api.h
$(sparc_ONLY)file path=usr/platform/sun4v/include/sys/idprom.h
$(sparc_ONLY)file path=usr/platform/sun4v/include/sys/intr.h
$(sparc_ONLY)file path=usr/platform/sun4v/include/sys/intreg.h
$(sparc_ONLY)file path=usr/platform/sun4v/include/sys/ivintr.h
$(sparc_ONLY)file path=usr/platform/sun4v/include/sys/machasi.h
$(sparc_ONLY)file path=usr/platform/sun4v/include/sys/machclock.h
$(sparc_ONLY)file path=usr/platform/sun4v/include/sys/machcpuvar.h
$(sparc_ONLY)file path=usr/platform/sun4v/include/sys/machintreg.h
$(sparc_ONLY)file path=usr/platform/sun4v/include/sys/machparam.h
$(sparc_ONLY)file path=usr/platform/sun4v/include/sys/machsystm.h
$(sparc_ONLY)file path=usr/platform/sun4v/include/sys/machthread.h
$(sparc_ONLY)file path=usr/platform/sun4v/include/sys/memlist_plat.h
$(sparc_ONLY)file path=usr/platform/sun4v/include/sys/memnode.h
$(sparc_ONLY)file path=usr/platform/sun4v/include/sys/mmu.h
$(sparc_ONLY)file path=usr/platform/sun4v/include/sys/nexusdebug.h
$(sparc_ONLY)file path=usr/platform/sun4v/include/sys/niagaraasi.h
$(sparc_ONLY)file path=usr/platform/sun4v/include/sys/niagararegs.h
$(sparc_ONLY)file path=usr/platform/sun4v/include/sys/ntwdt.h
$(sparc_ONLY)file path=usr/platform/sun4v/include/sys/pri.h
$(sparc_ONLY)file path=usr/platform/sun4v/include/sys/prom_debug.h
$(sparc_ONLY)file path=usr/platform/sun4v/include/sys/prom_plat.h
$(sparc_ONLY)file path=usr/platform/sun4v/include/sys/pte.h
$(sparc_ONLY)file path=usr/platform/sun4v/include/sys/qcn.h
$(sparc_ONLY)file path=usr/platform/sun4v/include/sys/scb.h
$(sparc_ONLY)file path=usr/platform/sun4v/include/sys/soft_state.h
```

```
$(sparc_ONLY)file path=usr/platform/sun4v/include/sys/sun4asi.h
$(sparc_ONLY)file path=usr/platform/sun4v/include/sys/tod.h
$(sparc_ONLY)file path=usr/platform/sun4v/include/sys/trapstat.h
$(sparc_ONLY)file path=usr/platform/sun4v/include/sys/traptrace.h
$(sparc_ONLY)file path=usr/platform/sun4v/include/sys/vis.h
$(sparc_ONLY)file path=usr/platform/sun4v/include/sys/vm_machparam.h
$(sparc_ONLY)file path=usr/platform/sun4v/include/sys/x_call.h
$(sparc_ONLY)file path=usr/platform/sun4v/include/sys/xc_impl.h
$(sparc_ONLY)file path=usr/platform/sun4v/include/sys/zsmach.h
$(sparc_ONLY)file path=usr/platform/sun4v/include/vm/hat_sfmmu.h
$(sparc_ONLY)file path=usr/platform/sun4v/include/vm/mach_sfmmu.h
file path=usr/share/man/man3head/acct.h.3head
file path=usr/share/man/man3head/aio.h.3head
file path=usr/share/man/man3head/ar.h.3head
file path=usr/share/man/man3head/archives.h.3head
file path=usr/share/man/man3head/assert.h.3head
file path=usr/share/man/man3head/complex.h.3head
file path=usr/share/man/man3head/cpio.h.3head
file path=usr/share/man/man3head/dirent.h.3head
file path=usr/share/man/man3head/errno.h.3head
file path=usr/share/man/man3head/fcntl.h.3head
file path=usr/share/man/man3head/fenv.h.3head
file path=usr/share/man/man3head/float.h.3head
file path=usr/share/man/man3head/floatingpoint.h.3head
file path=usr/share/man/man3head/fmtmsg.h.3head
file path=usr/share/man/man3head/fnmatch.h.3head
file path=usr/share/man/man3head/ftw.h.3head
file path=usr/share/man/man3head/glob.h.3head
file path=usr/share/man/man3head/grp.h.3head
file path=usr/share/man/man3head/iconv.h.3head
file path=usr/share/man/man3head/if.h.3head
file path=usr/share/man/man3head/in.h.3head
file path=usr/share/man/man3head/inet.h.3head
file path=usr/share/man/man3head/inttypes.h.3head
file path=usr/share/man/man3head/ipc.h.3head
file path=usr/share/man/man3head/iso646.h.3head
file path=usr/share/man/man3head/langinfo.h.3head
file path=usr/share/man/man3head/libgen.h.3head
file path=usr/share/man/man3head/libintl.h.3head
file path=usr/share/man/man3head/limits.h.3head
file path=usr/share/man/man3head/locale.h.3head
file path=usr/share/man/man3head/math.h.3head
file path=usr/share/man/man3head/mman.h.3head
file path=usr/share/man/man3head/monetary.h.3head
file path=usr/share/man/man3head/mqueue.h.3head
file path=usr/share/man/man3head/msg.h.3head
file path=usr/share/man/man3head/ndbm.h.3head
file path=usr/share/man/man3head/netdb.h.3head
file path=usr/share/man/man3head/nl_types.h.3head
file path=usr/share/man/man3head/poll.h.3head
file path=usr/share/man/man3head/pthread.h.3head
file path=usr/share/man/man3head/pwd.h.3head
file path=usr/share/man/man3head/regex.h.3head
file path=usr/share/man/man3head/resource.h.3head
file path=usr/share/man/man3head/sched.h.3head
file path=usr/share/man/man3head/search.h.3head
file path=usr/share/man/man3head/select.h.3head
file path=usr/share/man/man3head/sem.h.3head
file path=usr/share/man/man3head/semaphore.h.3head
file path=usr/share/man/man3head/setjmp.h.3head
file path=usr/share/man/man3head/shm.h.3head
file path=usr/share/man/man3head/siginfo.h.3head
file path=usr/share/man/man3head/signal.h.3head
file path=usr/share/man/man3head/socket.h.3head
file path=usr/share/man/man3head/spawn.h.3head
file path=usr/share/man/man3head/stat.h.3head
```



```
file path=usr/share/man/man3head/statvfs.h.3head
file path=usr/share/man/man3head/stdbool.h.3head
file path=usr/share/man/man3head/stddef.h.3head
file path=usr/share/man/man3head/stdint.h.3head
file path=usr/share/man/man3head/stdio.h.3head
file path=usr/share/man/man3head/stdlib.h.3head
file path=usr/share/man/man3head/string.h.3head
file path=usr/share/man/man3head/strings.h.3head
file path=usr/share/man/man3head/stropts.h.3head
file path=usr/share/man/man3head/syslog.h.3head
file path=usr/share/man/man3head/tar.h.3head
file path=usr/share/man/man3head/tcp.h.3head
file path=usr/share/man/man3head/termios.h.3head
file path=usr/share/man/man3head/tgmath.h.3head
file path=usr/share/man/man3head/time.h.3head
file path=usr/share/man/man3head/timeb.h.3head
file path=usr/share/man/man3head/times.h.3head
file path=usr/share/man/man3head/types.h.3head
file path=usr/share/man/man3head/types32.h.3head
file path=usr/share/man/man3head/ucontext.h.3head
file path=usr/share/man/man3head/uio.h.3head
file path=usr/share/man/man3head/ulimit.h.3head
file path=usr/share/man/man3head/un.h.3head
file path=usr/share/man/man3head/unistd.h.3head
file path=usr/share/man/man3head/utime.h.3head
file path=usr/share/man/man3head/utmpx.h.3head
file path=usr/share/man/man3head/utsname.h.3head
file path=usr/share/man/man3head/values.h.3head
file path=usr/share/man/man3head/wait.h.3head
file path=usr/share/man/man3head/wchar.h.3head
file path=usr/share/man/man3head/wctype.h.3head
file path=usr/share/man/man3head/wordexp.h.3head
file path=usr/share/man/man3head/xlocale.h.3head
file path=usr/share/man/man4/note.4
file path=usr/share/man/man5/prof.5
file path=usr/share/man/man7i/cdio.7i
file path=usr/share/man/man7i/dkio.7i
file path=usr/share/man/man7i/fbio.7i
file path=usr/share/man/man7i/fdio.7i
file path=usr/share/man/man7i/hdio.7i
file path=usr/share/man/man7i/iec61883.7i
file path=usr/share/man/man7i/mhd.7i
file path=usr/share/man/man7i/mtio.7i
file path=usr/share/man/man7i/prnio.7i
file path=usr/share/man/man7i/quotactl.7i
file path=usr/share/man/man7i/sesio.7i
file path=usr/share/man/man7i/sockio.7i
file path=usr/share/man/man7i/streamio.7i
file path=usr/share/man/man7i/termio.7i
file path=usr/share/man/man7i/termiox.7i
file path=usr/share/man/man7i/uscsi.7i
file path=usr/share/man/man7i/visual_io.7i
file path=usr/share/man/man7i/vt.7i
file path=usr/xpg4/include/curses.h
file path=usr/xpg4/include/term.h
file path=usr/xpg4/include/unctrl.h
legacy pkg=SUNWhea \
    desc="SunOS C/C++ header files for general development of software" \
    name="SunOS Header Files"
license cr_Sun license=cr_Sun
license lic_CDDL license=lic_CDDL
license license_in_headers license=license_in_headers
license usr/src/lib/pkcs11/include/THIRDPARTYLICENSE \
    license=usr/src/lib/pkcs11/include/THIRDPARTYLICENSE
link path=usr/include/iso/assert_iso.h target=../assert.h
link path=usr/include/iso/errno_iso.h target=../errno.h
```

```
1973 link path=usr/include/iso/float_iso.h target=../float.h
1974 link path=usr/include/iso/iso646_iso.h target=../iso646.h
1975 $(sparc_ONLY)link path=usr/platform/SUNW,A70/include target=../sun4u/include
1976 $(sparc_ONLY)link path=usr/platform/SUNW,Netra-T12/include \
1977     target=../sun4u/include
1978 $(sparc_ONLY)link path=usr/platform/SUNW,Netra-T4/include \
1979     target=../sun4u/include
1980 $(sparc_ONLY)link path=usr/platform/SUNW,SPARC-Enterprise/include \
1981     target=../sun4u/include
1982 $(sparc_ONLY)link path=usr/platform/SUNW,Serverblade1/include \
1983     target=../sun4u/include
1984 $(sparc_ONLY)link path=usr/platform/SUNW,Sun-Blade-100/include \
1985     target=../sun4u/include
1986 $(sparc_ONLY)link path=usr/platform/SUNW,Sun-Blade-1000/include \
1987     target=../sun4u/include
1988 $(sparc_ONLY)link path=usr/platform/SUNW,Sun-Blade-1500/include \
1989     target=../sun4u/include
1990 $(sparc_ONLY)link path=usr/platform/SUNW,Sun-Blade-2500/include \
1991     target=../sun4u/include
1992 $(sparc_ONLY)link path=usr/platform/SUNW,Sun-Fire-15000/include \
1993     target=../sun4u/include
1994 $(sparc_ONLY)link path=usr/platform/SUNW,Sun-Fire-280R/include \
1995     target=../sun4u/include
1996 $(sparc_ONLY)link path=usr/platform/SUNW,Sun-Fire-480R/include \
1997     target=../sun4u/include
1998 $(sparc_ONLY)link path=usr/platform/SUNW,Sun-Fire-880/include \
1999     target=../sun4u/include
2000 $(sparc_ONLY)link path=usr/platform/SUNW,Sun-Fire-V215/include \
2001     target=../sun4u/include
2002 $(sparc_ONLY)link path=usr/platform/SUNW,Sun-Fire-V240/include \
2003     target=../sun4u/include
2004 $(sparc_ONLY)link path=usr/platform/SUNW,Sun-Fire-V250/include \
2005     target=../sun4u/include
2006 $(sparc_ONLY)link path=usr/platform/SUNW,Sun-Fire-V440/include \
2007     target=../sun4u/include
2008 $(sparc_ONLY)link path=usr/platform/SUNW,Sun-Fire-V445/include \
2009     target=../sun4u/include
2010 $(sparc_ONLY)link path=usr/platform/SUNW,Sun-Fire-V490/include \
2011     target=../sun4u/include
2012 $(sparc_ONLY)link path=usr/platform/SUNW,Sun-Fire-V890/include \
2013     target=../sun4u/include
2014 $(sparc_ONLY)link path=usr/platform/SUNW,Sun-Fire/include \
2015     target=../sun4u/include
2016 $(sparc_ONLY)link path=usr/platform/SUNW,Ultra-2/include \
2017     target=../sun4u/include
2018 $(sparc_ONLY)link path=usr/platform/SUNW,Ultra-250/include \
2019     target=../sun4u/include
2020 $(sparc_ONLY)link path=usr/platform/SUNW,Ultra-4/include \
2021     target=../sun4u/include
2022 $(sparc_ONLY)link path=usr/platform/SUNW,Ultra-Enterprise-10000/include \
2023     target=../sun4u/include
2024 $(sparc_ONLY)link path=usr/platform/SUNW,Ultra-Enterprise/include \
2025     target=../sun4u/include
2026 $(sparc_ONLY)link path=usr/platform/SUNW,UltraSPARC-IIe-NetraCT-40/include \
2027     target=../sun4u/include
2028 $(sparc_ONLY)link path=usr/platform/SUNW,UltraSPARC-IIe-NetraCT-60/include \
2029     target=../sun4u/include
2030 $(sparc_ONLY)link path=usr/platform/SUNW,UltraSPARC-IIi-Netract/include \
2031     target=../sun4u/include
2032 link path=usr/share/man/man3head/acct.3head target=acct.h.3head
2033 link path=usr/share/man/man3head/aio.3head target=aio.h.3head
2034 link path=usr/share/man/man3head/ar.3head target=ar.h.3head
2035 link path=usr/share/man/man3head/archives.3head target=archives.h.3head
2036 link path=usr/share/man/man3head/assert.3head target=assert.h.3head
2037 link path=usr/share/man/man3head/complex.3head target=complex.h.3head
2038 link path=usr/share/man/man3head/cpio.3head target=cpio.h.3head
```



```
2039 link path=usr/share/man/man3head/dirent.3head target=dirent.h.3head
2040 link path=usr/share/man/man3head/errno.3head target=errno.h.3head
2041 link path=usr/share/man/man3head/fcntl.3head target=fcntl.h.3head
2042 link path=usr/share/man/man3head/fenv.3head target=fenv.h.3head
2043 link path=usr/share/man/man3head/float.3head target=float.h.3head
2044 link path=usr/share/man/man3head/floatingpoint.3head \
2045     target=floatingpoint.h.3head
2046 link path=usr/share/man/man3head/fmtmsg.3head target=fmtmsg.h.3head
2047 link path=usr/share/man/man3head/fnmatch.3head target=fnmatch.h.3head
2048 link path=usr/share/man/man3head/ftw.3head target=ftw.h.3head
2049 link path=usr/share/man/man3head/glob.3head target=glob.h.3head
2050 link path=usr/share/man/man3head/grp.3head target=grp.h.3head
2051 link path=usr/share/man/man3head/iconv.3head target=iconv.h.3head
2052 link path=usr/share/man/man3head/if.3head target=if.h.3head
2053 link path=usr/share/man/man3head/in.3head target=in.h.3head
2054 link path=usr/share/man/man3head/inet.3head target=inet.h.3head
2055 link path=usr/share/man/man3head/inttypes.3head target=inttypes.h.3head
2056 link path=usr/share/man/man3head/ipc.3head target=ipc.h.3head
2057 link path=usr/share/man/man3head/iso646.3head target=iso646.h.3head
2058 link path=usr/share/man/man3head/langinfo.3head target=langinfo.h.3head
2059 link path=usr/share/man/man3head/libgen.3head target=libgen.h.3head
2060 link path=usr/share/man/man3head/libintl.3head target=libintl.h.3head
2061 link path=usr/share/man/man3head/limits.3head target=limits.h.3head
2062 link path=usr/share/man/man3head/locale.3head target=locale.h.3head
2063 link path=usr/share/man/man3head/math.3head target=math.h.3head
2064 link path=usr/share/man/man3head/mman.3head target=mman.h.3head
2065 link path=usr/share/man/man3head/monetary.3head target=monetary.h.3head
2066 link path=usr/share/man/man3head/mqueue.3head target=mqueue.h.3head
2067 link path=usr/share/man/man3head/msg.3head target=msg.h.3head
2068 link path=usr/share/man/man3head/ndbm.3head target=ndbm.h.3head
2069 link path=usr/share/man/man3head/netdb.3head target=netdb.h.3head
2070 link path=usr/share/man/man3head/nl_types.3head target=nl_types.h.3head
2071 link path=usr/share/man/man3head/poll.3head target=poll.h.3head
2072 link path=usr/share/man/man3head/pthread.3head target=pthread.h.3head
2073 link path=usr/share/man/man3head/pwd.3head target=pwd.h.3head
2074 link path=usr/share/man/man3head/regex.3head target=regex.h.3head
2075 link path=usr/share/man/man3head/resource.3head target=resource.h.3head
2076 link path=usr/share/man/man3head/sched.3head target=sched.h.3head
2077 link path=usr/share/man/man3head/search.3head target=search.h.3head
2078 link path=usr/share/man/man3head/select.3head target=select.h.3head
2079 link path=usr/share/man/man3head/sem.3head target=sem.h.3head
2080 link path=usr/share/man/man3head/semaphore.3head target=semaphore.h.3head
2081 link path=usr/share/man/man3head/setjmp.3head target=setjmp.h.3head
2082 link path=usr/share/man/man3head/shm.3head target=shm.h.3head
2083 link path=usr/share/man/man3head/siginfo.3head target=siginfo.h.3head
2084 link path=usr/share/man/man3head/signal.3head target=signal.h.3head
2085 link path=usr/share/man/man3head/socket.3head target=socket.h.3head
2086 link path=usr/share/man/man3head/spawn.3head target=spawn.h.3head
2087 link path=usr/share/man/man3head/stat.3head target=stat.h.3head
2088 link path=usr/share/man/man3head/statvfs.3head target=statvfs.h.3head
2089 link path=usr/share/man/man3head/stdbool.3head target=stdbool.h.3head
2090 link path=usr/share/man/man3head/stddef.3head target=stddef.h.3head
2091 link path=usr/share/man/man3head/stdint.3head target=stdint.h.3head
2092 link path=usr/share/man/man3head/stdio.3head target=stdio.h.3head
2093 link path=usr/share/man/man3head/stdlib.3head target=stdlib.h.3head
2094 link path=usr/share/man/man3head/string.3head target=string.h.3head
2095 link path=usr/share/man/man3head/strings.3head target=strings.h.3head
2096 link path=usr/share/man/man3head/stropts.3head target=stropts.h.3head
2097 link path=usr/share/man/man3head/syslog.3head target=syslog.h.3head
2098 link path=usr/share/man/man3head/tar.3head target=tar.h.3head
2099 link path=usr/share/man/man3head/tcp.3head target=tcp.h.3head
2100 link path=usr/share/man/man3head/termios.3head target=termios.h.3head
2101 link path=usr/share/man/man3head/tgmath.3head target=tgmath.h.3head
2102 link path=usr/share/man/man3head/time.3head target=time.h.3head
2103 link path=usr/share/man/man3head/timeb.3head target=timeb.h.3head
2104 link path=usr/share/man/man3head/times.3head target=times.h.3head
```

```
2105 link path=usr/share/man/man3head/types.3head target=types.h.3head
2106 link path=usr/share/man/man3head/types32.3head target=types32.h.3head
2107 link path=usr/share/man/man3head/ucontext.3head target=ucontext.h.3head
2108 link path=usr/share/man/man3head/uio.3head target=uio.h.3head
2109 link path=usr/share/man/man3head/ulimit.3head target=ulimit.h.3head
2110 link path=usr/share/man/man3head/un.3head target=un.h.3head
2111 link path=usr/share/man/man3head/unistd.3head target=unistd.h.3head
2112 link path=usr/share/man/man3head/utime.3head target=utime.h.3head
2113 link path=usr/share/man/man3head/utmpx.3head target=utmpx.h.3head
2114 link path=usr/share/man/man3head/utsname.3head target=utsname.h.3head
2115 link path=usr/share/man/man3head/values.3head target=values.h.3head
2116 link path=usr/share/man/man3head/wait.3head target=wait.h.3head
2117 link path=usr/share/man/man3head/wchar.3head target=wchar.h.3head
2118 link path=usr/share/man/man3head/wctype.3head target=wctype.h.3head
2119 link path=usr/share/man/man3head/wordexp.3head target=wordexp.h.3head
2120 link path=usr/share/man/man3head/xlocale.3head target=xlocale.h.3head
2121 $(i386_ONLY)link path=usr/share/src/uts/i86pc/sys \
2122     target=../../../../platform/i86pc/include/sys
2123 $(i386_ONLY)link path=usr/share/src/uts/i86pc/vm \
2124     target=../../../../platform/i86pc/include/vm
2125 $(i386_ONLY)link path=usr/share/src/uts/i86xpv/sys \
2126     target=../../../../platform/i86xpv/include/sys
2127 $(i386_ONLY)link path=usr/share/src/uts/i86xpv/vm \
2128     target=../../../../platform/i86xpv/include/vm
2129 $(sparc_ONLY)link path=usr/share/src/uts/sun4u/sys \
2130     target=../../../../platform/sun4u/include/sys
2131 $(sparc_ONLY)link path=usr/share/src/uts/sun4u/vm \
2132     target=../../../../platform/sun4u/include/vm
2133 $(sparc_ONLY)link path=usr/share/src/uts/sun4v/sys \
2134     target=../../../../platform/sun4v/include/sys
2135 $(sparc_ONLY)link path=usr/share/src/uts/sun4v/vm \
2136     target=../../../../platform/sun4v/include/vm
```

```
**********************************************************
   44752 Wed May 27 19:49:23 2015
new/usr/src/uts/common/Makefile.files
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#

#
# Copyright (c) 1991, 2010, Oracle and/or its affiliates. All rights reserved.
# Copyright (c) 2013 by Delphix. All rights reserved.
# Copyright (c) 2013 by Saso Kiselkov. All rights reserved.
# Copyright 2014 Nexenta Systems, Inc.  All rights reserved.
#

#
# This Makefile defines all file modules for the directory uts/common
# and its children. These are the source files which may be considered
# common to all SunOS systems.

i386_CORE_OBJS += \
                atomic.o        \
                avintr.o        \
                pic.o

sparc_CORE_OBJS +=

COMMON_CORE_OBJS +=             \
                beep.o          \
                bitset.o        \
                bp_map.o        \
                brand.o         \
                cpucaps.o       \
                cmt.o           \
                cmt_policy.o    \
                cpu.o           \
                cpu_event.o     \
                cpu_intr.o      \
                cpu_pm.o        \
                cpupart.o       \
                cap_util.o      \
                disp.o          \
                group.o         \
                kstat_fr.o      \
                iscsiboot_prop.o        \
```



```
                lgrp.o          \
                lgrp_topo.o     \
                mmapobj.o       \
                mutex.o         \
                page_lock.o     \
                page_retire.o   \
                panic.o         \
                param.o         \
                pg.o            \
                pghw.o          \
                putnext.o       \
                rctl_proc.o     \
                rwlock.o        \
                seg_kmem.o      \
                softint.o       \
                string.o        \
                strtol.o        \
                strtoul.o       \
                strtoll.o       \
                strtoull.o      \
                thread_intr.o   \
                vm_page.o       \
                vm_pagelist.o   \
                zlib_obj.o      \
                clock_tick.o

CORE_OBJS +=    $(COMMON_CORE_OBJS) $($(MACH)_CORE_OBJS)

ZLIB_OBJS =     zutil.o zmod.o zmod_subr.o \
                adler32.o crc32.o deflate.o inffast.o \
                inflate.o inftrees.o trees.o

GENUNIX_OBJS += \
                access.o        \
                acl.o           \
                acl_common.o    \
                adjtime.o       \
                alarm.o         \
                aio_subr.o      \
                auditsys.o      \
                audit_core.o    \
                audit_zone.o    \
                audit_memory.o  \
                autoconf.o      \
                avl.o           \
                bdev_dsort.o    \
                bio.o           \
                bitmap.o        \
                blabel.o        \
                brandsys.o      \
                bz2blocksort.o  \
                bz2compress.o   \
                bz2decompress.o \
                bz2randtable.o  \
                bz2bzlib.o      \
                bz2crctable.o   \
                bz2huffman.o    \
                callb.o         \
                callout.o       \
                chdir.o         \
                chmod.o         \
                chown.o         \
                cladm.o         \
                class.o         \
                clock.o         \
                clock_highres.o \
```

```
125                 clock_realtime.o\
126                 close.o         \
127                 compress.o      \
128                 condvar.o       \
129                 conf.o          \
130                 console.o       \
131                 contract.o      \
132                 copyops.o       \
133                 core.o          \
134                 corectl.o       \
135                 cred.o          \
136                 cs_stubs.o      \
137                 dacf.o          \
138                 dacf_clnt.o     \
139                 damap.o \
140                 cyclic.o        \
141                 ddi.o           \
142                 ddifm.o         \
143                 ddi_hp_impl.o   \
144                 ddi_hp_ndi.o    \
145                 ddi_intr.o      \
146                 ddi_intr_impl.o \
147                 ddi_intr_irm.o  \
148                 ddi_nodeid.o    \
149                 ddi_periodic.o  \
150                 devcfg.o        \
151                 devcache.o      \
152                 device.o        \
153                 devid.o         \
154                 devid_cache.o   \
155                 devid_scsi.o    \
156                 devid_smp.o     \
157                 devpolicy.o     \
158                 disp_lock.o     \
159                 dnlc.o          \
160                 driver.o        \
161                 dumpsubr.o      \
162                 driver_lyr.o    \
163                 dtrace_subr.o   \
164                 errorq.o        \
165                 etheraddr.o     \
166                 evchannels.o    \
167                 exacct.o        \
168                 exacct_core.o   \
169                 exec.o          \
170                 exit.o          \
171                 fbio.o          \
172                 fcntl.o         \
173                 fdbuffer.o      \
174                 fdsync.o        \
175                 fem.o           \
176                 ffs.o           \
177                 fio.o           \
178                 flock.o         \
179                 fm.o            \
180                 fork.o          \
181                 vpm.o           \
182                 fs_reparse.o    \
183                 fs_subr.o       \
184                 fsflush.o       \
185                 ftrace.o        \
186                 getcwd.o        \
187                 getdents.o      \
188                 getloadavg.o    \
189                 getpagesizes.o  \
190                 getpid.o        \
```



```
191                 gfs.o           \
192                 rusagesys.o     \
193                 gid.o           \
194                 groups.o        \
195                 grow.o          \
196                 hat_refmod.o    \
197                 id32.o          \
198                 id_space.o      \
199                 inet_ntop.o     \
200                 instance.o      \
201                 ioctl.o         \
202                 ip_cksum.o      \
203                 issetugid.o     \
204                 ippconf.o       \
205                 kcpc.o          \
206                 kdi.o           \
207                 kiconv.o        \
208                 klpd.o          \
209                 kmem.o          \
210                 ksyms_snapshot.o        \
211                 l_strplumb.o    \
212                 labelsys.o      \
213                 link.o          \
214                 list.o          \
215                 lockstat_subr.o \
216                 log_sysevent.o  \
217                 logsubr.o       \
218                 lookup.o        \
219                 lseek.o         \
220                 ltos.o          \
221                 lwp.o           \
222                 lwp_create.o    \
223                 lwp_info.o      \
224                 lwp_self.o      \
225                 lwp_sobj.o      \
226                 lwp_timer.o     \
227                 lwpsys.o        \
228                 main.o          \
229                 mmapobjsys.o    \
230                 memcntl.o       \
231                 memstr.o        \
232                 lgrpsys.o       \
233                 mkdir.o         \
234                 mknod.o         \
235                 mount.o         \
236                 move.o          \
237                 msacct.o        \
238                 multidata.o     \
239                 nbmlock.o       \
240                 ndifm.o         \
241                 nice.o          \
242                 netstack.o      \
243                 ntptime.o       \
244                 nvpair.o        \
245                 nvpair_alloc_system.o   \
246                 nvpair_alloc_fixed.o    \
247                 fnvpair.o       \
248                 octet.o         \
249                 open.o          \
250                 p_online.o      \
251                 pathconf.o      \
252                 pathname.o      \
253                 pause.o         \
254                 serializer.o    \
255                 pci_intr_lib.o  \
256                 pci_cap.o       \
```

```
257                 pcifm.o         \
258                 pgrp.o          \
259                 pgrpsys.o       \
260                 pid.o           \
261                 pkp_hash.o      \
262                 policy.o        \
263                 poll.o          \
264                 pool.o          \
265                 pool_pset.o     \
266                 port_subr.o     \
267                 ppriv.o         \
268                 printf.o        \
269                 priocntl.o      \
270                 priv.o          \
271                 priv_const.o    \
272                 proc.o          \
273                 psecflags.o     \
274 #endif /* ! codereview */
275                 procset.o       \
276                 processor_bind.o        \
277                 processor_info.o        \
278                 profil.o        \
279                 project.o       \
280                 qsort.o         \
281                 getrandom.o     \
282                 rctl.o          \
283                 rctlsys.o       \
284                 readlink.o      \
285                 refstr.o        \
286                 rename.o        \
287                 resolvepath.o   \
288                 retire_store.o  \
289                 process.o       \
290                 rlimit.o        \
291                 rmap.o          \
292                 rw.o            \
293                 rwstlock.o      \
294                 sad_conf.o      \
295                 sid.o           \
296                 sidsys.o        \
297                 sched.o         \
298                 schedctl.o      \
299                 sctp_crc32.o    \
300                 seg_dev.o       \
301                 seg_kp.o        \
302                 seg_kpm.o       \
303                 seg_map.o       \
304                 seg_vn.o        \
305                 seg_spt.o       \
306                 semaphore.o     \
307                 sendfile.o      \
308                 session.o       \
309                 share.o         \
310                 shuttle.o       \
311                 sig.o           \
312                 sigaction.o     \
313                 sigaltstack.o   \
314                 signotify.o     \
315                 sigpending.o    \
316                 sigprocmask.o   \
317                 sigqueue.o      \
318                 sigsendset.o    \
319                 sigsuspend.o    \
320                 sigtimedwait.o  \
321                 sleepq.o        \
322                 sock_conf.o     \
```



```
323                 space.o         \
324                 sscanf.o        \
325                 stat.o          \
326                 statfs.o        \
327                 statvfs.o       \
328                 stol.o          \
329                 str_conf.o      \
330                 strcalls.o      \
331                 stream.o        \
332                 streamio.o      \
333                 strext.o        \
334                 strsubr.o       \
335                 strsun.o        \
336                 subr.o          \
337                 sunddi.o        \
338                 sunmdi.o        \
339                 sunndi.o        \
340                 sunpci.o        \
341                 sunpm.o         \
342                 sundlpi.o       \
343                 suntpi.o        \
344                 swap_subr.o     \
345                 swap_vnops.o    \
346                 symlink.o       \
347                 sync.o          \
348                 sysclass.o      \
349                 sysconfig.o     \
350                 sysent.o        \
351                 sysfs.o         \
352                 systeminfo.o    \
353                 task.o          \
354                 taskq.o         \
355                 tasksys.o       \
356                 time.o          \
357                 timer.o         \
358                 times.o         \
359                 timers.o        \
360                 thread.o        \
361                 tlabel.o        \
362                 tnf_res.o       \
363                 turnstile.o     \
364                 tty_common.o    \
365                 u8_textprep.o   \
366                 uadmin.o        \
367                 uconv.o         \
368                 ucredsys.o      \
369                 uid.o           \
370                 umask.o         \
371                 umount.o        \
372                 uname.o         \
373                 unix_bb.o       \
374                 unlink.o        \
375                 urw.o           \
376                 utime.o         \
377                 utssys.o        \
378                 uucopy.o        \
379                 vfs.o           \
380                 vfs_conf.o      \
381                 vmem.o          \
382                 vm_anon.o       \
383                 vm_as.o         \
384                 vm_meter.o      \
385                 vm_pageout.o    \
386                 vm_pvn.o        \
387                 vm_rm.o         \
388                 vm_seg.o        \
```

```
389                 vm_subr.o       \
390                 vm_swap.o       \
391                 vm_usage.o      \
392                 vnode.o         \
393                 vuid_queue.o    \
394                 vuid_store.o    \
395                 waitq.o         \
396                 watchpoint.o    \
397                 yield.o         \
398                 scsi_confdata.o \
399                 xattr.o         \
400                 xattr_common.o  \
401                 xdr_mblk.o      \
402                 xdr_mem.o       \
403                 xdr.o           \
404                 xdr_array.o     \
405                 xdr_refer.o     \
406                 xhat.o          \
407                 zone.o

409 #
410 #       Stubs for the stand-alone linker/loader
411 #
412 sparc_GENSTUBS_OBJS =   \
413         kobj_stubs.o

415 i386_GENSTUBS_OBJS =

417 COMMON_GENSTUBS_OBJS =

419 GENSTUBS_OBJS += $(COMMON_GENSTUBS_OBJS) $($(MACH)_GENSTUBS_OBJS)

421 #
422 #       DTrace and DTrace Providers
423 #
424 DTRACE_OBJS += dtrace.o dtrace_isa.o dtrace_asm.o

426 SDT_OBJS += sdt_subr.o

428 PROFILE_OBJS += profile.o

430 SYSTRACE_OBJS += systrace.o

432 LOCKSTAT_OBJS += lockstat.o

434 FASTTRAP_OBJS += fasttrap.o fasttrap_isa.o

436 DCPC_OBJS += dcpc.o

438 #
439 #       Driver (pseudo-driver) Modules
440 #
441 IPP_OBJS +=     ippctl.o

443 AUDIO_OBJS += audio_client.o audio_ddi.o audio_engine.o \
444             audio_fltdata.o audio_format.o audio_ctrl.o \
445             audio_grc3.o audio_output.o audio_input.o \
446             audio_oss.o audio_sun.o

448 AUDIOEMU10K_OBJS += audioemu10k.o

450 AUDIOENS_OBJS += audioens.o

452 AUDIOVIA823X_OBJS += audiovia823x.o

454 AUDIOVIA97_OBJS += audiovia97.o
```



```
456 AUDIO1575_OBJS += audio1575.o

458 AUDIO810_OBJS += audio810.o

460 AUDIOCMI_OBJS += audiocmi.o

462 AUDIOCMIHD_OBJS += audiocmihd.o

464 AUDIOHD_OBJS += audiohd.o

466 AUDIOIXP_OBJS += audioixp.o

468 AUDIOLS_OBJS += audiols.o

470 AUDIOP16X_OBJS += audiop16x.o

472 AUDIOPCI_OBJS += audiopci.o

474 AUDIOSOLO_OBJS += audiosolo.o

476 AUDIOTS_OBJS += audiots.o

478 AC97_OBJS += ac97.o ac97_ad.o ac97_alc.o ac97_cmi.o

480 BLKDEV_OBJS += blkdev.o

482 CARDBUS_OBJS += cardbus.o cardbus_hp.o cardbus_cfg.o

484 CONSKBD_OBJS += conskbd.o

486 CONSMS_OBJS +=  consms.o

488 OLDPTY_OBJS +=  tty_ptyconf.o

490 PTC_OBJS +=     tty_pty.o

492 PTSL_OBJS +=    tty_pts.o

494 PTM_OBJS +=     ptm.o

496 MII_OBJS +=     mii.o mii_cicada.o mii_natsemi.o mii_intel.o mii_qualsemi.o \
497                 mii_marvell.o mii_realtek.o mii_other.o

499 PTS_OBJS +=     pts.o

501 PTY_OBJS +=     ptms_conf.o

503 SAD_OBJS +=     sad.o

505 MD4_OBJS +=     md4.o md4_mod.o

507 MD5_OBJS +=     md5.o md5_mod.o

509 SHA1_OBJS +=    sha1.o sha1_mod.o

511 SHA2_OBJS +=    sha2.o sha2_mod.o

513 IPGPC_OBJS +=   classifierddi.o classifier.o filters.o trie.o table.o \
514                 ba_table.o

516 DSCPMK_OBJS +=  dscpmk.o dscpmkddi.o

518 DLCOSMK_OBJS += dlcosmk.o dlcosmkddi.o

520 FLOWACCT_OBJS +=        flowacctddi.o flowacct.o
```

```
TOKENMT_OBJS += tokenmt.o tokenmtddi.o

TSWTCL_OBJS +=  tswtcl.o tswtclddi.o

ARP_OBJS +=     arpddi.o

ICMP_OBJS +=    icmpddi.o

ICMP6_OBJS +=   icmp6ddi.o

RTS_OBJS +=     rtsddi.o

IP_ICMP_OBJS =  icmp.o icmp_opt_data.o
IP_RTS_OBJS =   rts.o rts_opt_data.o
IP_TCP_OBJS =   tcp.o tcp_fusion.o tcp_opt_data.o tcp_sack.o tcp_stats.o \
                tcp_misc.o tcp_timers.o tcp_time_wait.o tcp_tpi.o tcp_output.o \
                tcp_input.o tcp_socket.o tcp_bind.o tcp_cluster.o tcp_tunables.o
IP_UDP_OBJS =   udp.o udp_opt_data.o udp_tunables.o udp_stats.o
IP_SCTP_OBJS =  sctp.o sctp_opt_data.o sctp_output.o \
                sctp_init.o sctp_input.o sctp_cookie.o \
                sctp_conn.o sctp_error.o sctp_snmp.o \
                sctp_tunables.o sctp_shutdown.o sctp_common.o \
                sctp_timer.o sctp_heartbeat.o sctp_hash.o \
                sctp_bind.o sctp_notify.o sctp_asconf.o \
                sctp_addr.o tn_ipopt.o tnet.o ip_netinfo.o \
                sctp_misc.o
IP_ILB_OBJS =   ilb.o ilb_nat.o ilb_conn.o ilb_alg_hash.o ilb_alg_rr.o

IP_OBJS +=      igmp.o ipmp.o ip.o ip6.o ip6_asp.o ip6_if.o ip6_ire.o \
                ip6_rts.o ip_if.o ip_ire.o ip_listutils.o ip_mroute.o \
                ip_multi.o ip2mac.o ip_ndp.o ip_rts.o ip_srcid.o \
                ipddi.o ipdrop.o mi.o nd.o tunables.o optcom.o snmpcom.o \
                ipsec_loader.o spd.o ipclassifier.o inet_common.o ip_squeue.o \
                squeue.o ip_sadb.o ip_ftable.o proto_set.o radix.o ip_dummy.o \
                ip_helper_stream.o ip_tunables.o \
                ip_output.o ip_input.o ip6_input.o ip6_output.o ip_arp.o \
                conn_opt.o ip_attr.o ip_dce.o \
                $(IP_ICMP_OBJS) \
                $(IP_RTS_OBJS) \
                $(IP_TCP_OBJS) \
                $(IP_UDP_OBJS) \
                $(IP_SCTP_OBJS) \
                $(IP_ILB_OBJS)

IP6_OBJS +=     ip6ddi.o

HOOK_OBJS +=    hook.o

NETI_OBJS +=    neti_impl.o neti_mod.o neti_stack.o

KEYSOCK_OBJS += keysockddi.o keysock.o keysock_opt_data.o

IPNET_OBJS +=   ipnet.o ipnet_bpf.o

SPDSOCK_OBJS += spdsockddi.o spdsock.o spdsock_opt_data.o

IPSECESP_OBJS += ipsecespddi.o ipsecesp.o

IPSECAH_OBJS += ipsecahddi.o ipsecah.o sadb.o

SPPP_OBJS +=    sppp.o sppp_dlpi.o sppp_mod.o s_common.o

SPPPTUN_OBJS += sppptun.o sppptun_mod.o

SPPPASYN_OBJS += spppasyn.o spppasyn_mod.o
```



```
SPPPCOMP_OBJS += spppcomp.o spppcomp_mod.o deflate.o bsd-comp.o vjcompress.o \
                 zlib.o

TCP_OBJS +=     tcpddi.o

TCP6_OBJS +=    tcp6ddi.o

NCA_OBJS +=     ncaddi.o

SDP_SOCK_MOD_OBJS += sockmod_sdp.o socksdp.o socksdpsubr.o

SCTP_SOCK_MOD_OBJS += sockmod_sctp.o socksctp.o socksctpsubr.o

PFP_SOCK_MOD_OBJS += sockmod_pfp.o

RDS_SOCK_MOD_OBJS += sockmod_rds.o

RDS_OBJS +=     rdsddi.o rdssubr.o rds_opt.o rds_ioctl.o

RDSIB_OBJS +=   rdsib.o rdsib_ib.o rdsib_cm.o rdsib_ep.o rdsib_buf.o \
                rdsib_debug.o rdsib_sc.o

RDSV3_OBJS +=   af_rds.o rdsv3_ddi.o bind.o loop.o threads.o connection.o \
                transport.o cong.o sysctl.o message.o rds_recv.o send.o \
                stats.o info.o page.o rdma_transport.o ib_ring.o ib_rdma.o \
                ib_recv.o ib.o ib_send.o ib_sysctl.o ib_stats.o ib_cm.o \
                rdsv3_sc.o rdsv3_debug.o rdsv3_impl.o rdma.o rdsv3_af_thr.o

ISER_OBJS +=    iser.o iser_cm.o iser_cq.o iser_ib.o iser_idm.o \
                iser_resource.o iser_xfer.o

UDP_OBJS +=     udpddi.o

UDP6_OBJS +=    udp6ddi.o

SY_OBJS +=      gentty.o

TCO_OBJS +=     ticots.o

TCOO_OBJS +=    ticotsord.o

TCL_OBJS +=     ticlts.o

TL_OBJS +=      tl.o

DUMP_OBJS +=    dump.o

BPF_OBJS +=     bpf.o bpf_filter.o bpf_mod.o bpf_dlt.o bpf_mac.o

CLONE_OBJS +=   clone.o

CN_OBJS +=      cons.o

DLD_OBJS +=     dld_drv.o dld_proto.o dld_str.o dld_flow.o

DLS_OBJS +=     dls.o dls_link.o dls_mod.o dls_stat.o dls_mgmt.o

GLD_OBJS +=     gld.o gldutil.o

MAC_OBJS +=     mac.o  mac_bcast.o mac_client.o mac_datapath_setup.o mac_flow.o
                mac_hio.o mac_mod.o mac_ndd.o mac_provider.o mac_sched.o \
                mac_protect.o mac_soft_ring.o mac_stat.o mac_util.o

MAC_6TO4_OBJS +=        mac_6to4.o
```

```
MAC_ETHER_OBJS +=       mac_ether.o

MAC_IPV4_OBJS +=        mac_ipv4.o

MAC_IPV6_OBJS +=        mac_ipv6.o

MAC_WIFI_OBJS +=        mac_wifi.o

MAC_IB_OBJS +=          mac_ib.o

IPTUN_OBJS +=   iptun_dev.o iptun_ctl.o iptun.o

AGGR_OBJS +=    aggr_dev.o aggr_ctl.o aggr_grp.o aggr_port.o \
                aggr_send.o aggr_recv.o aggr_lacp.o

SOFTMAC_OBJS += softmac_main.o softmac_ctl.o softmac_capab.o \
                softmac_dev.o softmac_stat.o softmac_pkt.o softmac_fp.o

NET80211_OBJS += net80211.o net80211_proto.o net80211_input.o \
                 net80211_output.o net80211_node.o net80211_crypto.o \
                 net80211_crypto_none.o net80211_crypto_wep.o net80211_ioctl.o \
                 net80211_crypto_tkip.o net80211_crypto_ccmp.o  \
                 net80211_ht.o

VNIC_OBJS +=    vnic_ctl.o vnic_dev.o

SIMNET_OBJS +=  simnet.o

IB_OBJS +=      ibnex.o ibnex_ioctl.o ibnex_hca.o

IBCM_OBJS +=    ibcm_impl.o ibcm_sm.o ibcm_ti.o ibcm_utils.o ibcm_path.o \
                ibcm_arp.o ibcm_arp_link.o

IBDM_OBJS +=    ibdm.o

IBDMA_OBJS +=   ibdma.o

IBMF_OBJS +=    ibmf.o ibmf_impl.o ibmf_dr.o ibmf_wqe.o ibmf_ud_dest.o ibmf_mod.
                ibmf_send.o ibmf_recv.o ibmf_handlers.o ibmf_trans.o \
                ibmf_timers.o ibmf_msg.o ibmf_utils.o ibmf_rmpp.o \
                ibmf_saa.o ibmf_saa_impl.o ibmf_saa_utils.o ibmf_saa_events.o

IBTL_OBJS +=    ibtl_impl.o ibtl_util.o ibtl_mem.o ibtl_handlers.o ibtl_qp.o \
                ibtl_cq.o ibtl_wr.o ibtl_hca.o ibtl_chan.o ibtl_cm.o \
                ibtl_mcg.o ibtl_ibnex.o ibtl_srq.o ibtl_part.o

TAVOR_OBJS +=   tavor.o tavor_agents.o tavor_cfg.o tavor_ci.o tavor_cmd.o \
                tavor_cq.o tavor_event.o tavor_ioctl.o tavor_misc.o \
                tavor_mr.o tavor_qp.o tavor_qpmod.o tavor_rsrc.o \
                tavor_srq.o tavor_stats.o tavor_umap.o tavor_wr.o

HERMON_OBJS +=  hermon.o hermon_agents.o hermon_cfg.o hermon_ci.o hermon_cmd.o \
                hermon_cq.o hermon_event.o hermon_ioctl.o hermon_misc.o \
                hermon_mr.o hermon_qp.o hermon_qpmod.o hermon_rsrc.o \
                hermon_srq.o hermon_stats.o hermon_umap.o hermon_wr.o \
                hermon_fcoib.o hermon_fm.o

DAPLT_OBJS +=   daplt.o

SOL_OFS_OBJS += sol_cma.o sol_ib_cma.o sol_uobj.o \
                sol_ofs_debug_util.o sol_ofs_gen_util.o \
                sol_kverbs.o

SOL_UCMA_OBJS +=        sol_ucma.o

SOL_UVERBS_OBJS +=      sol_uverbs.o sol_uverbs_comp.o sol_uverbs_event.o \
```



```
                        sol_uverbs_hca.o sol_uverbs_qp.o

SOL_UMAD_OBJS += sol_umad.o

KSTAT_OBJS +=   kstat.o

KSYMS_OBJS +=   ksyms.o

INSTANCE_OBJS += inst_sync.o

IWSCN_OBJS +=   iwscons.o

LOFI_OBJS +=    lofi.o LzmaDec.o

FSSNAP_OBJS +=  fssnap.o

FSSNAPIF_OBJS += fssnap_if.o

MM_OBJS +=      mem.o

PHYSMEM_OBJS += physmem.o

OPTIONS_OBJS += options.o

WINLOCK_OBJS += winlockio.o

PM_OBJS +=      pm.o
SRN_OBJS +=     srn.o

PSEUDO_OBJS +=  pseudonex.o

RAMDISK_OBJS += ramdisk.o

LLC1_OBJS += llc1.o

USBKBM_OBJS += usbkbm.o

USBWCM_OBJS += usbwcm.o

BOFI_OBJS += bofi.o

HID_OBJS += hid.o

USBSKEL_OBJS += usbskel.o

USBVC_OBJS += usbvc.o usbvc_v4l2.o

HIDPARSER_OBJS += hidparser.o

USB_AC_OBJS += usb_ac.o

USB_AS_OBJS += usb_as.o

USB_AH_OBJS += usb_ah.o

USBMS_OBJS += usbms.o

USBPRN_OBJS += usbprn.o

UGEN_OBJS += ugen.o

USBSER_OBJS += usbser.o usbser_rseq.o

USBSACM_OBJS += usbsacm.o

USBSER_KEYSPAN_OBJS += usbser_keyspan.o keyspan_dsd.o keyspan_pipe.o
```

```
786 USBS49_FW_OBJS += keyspan_49fw.o

788 USBSPRL_OBJS += usbser_pl2303.o pl2303_dsd.o

790 USBFTDI_OBJS += usbser_uftdi.o uftdi_dsd.o

792 USBECM_OBJS += usbecm.o

794 WC_OBJS += wscons.o vcons.o

796 VCONS_CONF_OBJS += vcons_conf.o

798 SCSI_OBJS +=    scsi_capabilities.o scsi_confsubr.o scsi_control.o \
799                 scsi_data.o scsi_fm.o scsi_hba.o scsi_reset_notify.o \
800                 scsi_resource.o scsi_subr.o scsi_transport.o scsi_watch.o \
801                 smp_transport.o

803 SCSI_VHCI_OBJS +=               scsi_vhci.o mpapi_impl.o scsi_vhci_tpgs.o

805 SCSI_VHCI_F_SYM_OBJS +=         sym.o

807 SCSI_VHCI_F_TPGS_OBJS +=        tpgs.o

809 SCSI_VHCI_F_ASYM_SUN_OBJS +=    asym_sun.o

811 SCSI_VHCI_F_SYM_HDS_OBJS +=     sym_hds.o

813 SCSI_VHCI_F_TAPE_OBJS +=        tape.o

815 SCSI_VHCI_F_TPGS_TAPE_OBJS +=   tpgs_tape.o

817 SGEN_OBJS +=    sgen.o

819 SMP_OBJS +=     smp.o

821 SATA_OBJS +=    sata.o

823 USBA_OBJS +=    hcdi.o  usba.o  usbai.o hubdi.o parser.o genconsole.o \
824                 usbai_pipe_mgmt.o usbai_req.o usbai_util.o usbai_register.o \
825                 usba_devdb.o usba10_calls.o usba_ugen.o

827 USBA10_OBJS +=  usba10.o

829 RSM_OBJS +=     rsm.o   rsmka_pathmanager.o     rsmka_util.o

831 RSMOPS_OBJS +=  rsmops.o

833 S1394_OBJS +=   t1394.o t1394_errmsg.o s1394.o s1394_addr.o s1394_asynch.o \
834                 s1394_bus_reset.o s1394_cmp.o s1394_csr.o s1394_dev_disc.o \
835                 s1394_fa.o s1394_fcp.o \
836                 s1394_hotplug.o s1394_isoch.o s1394_misc.o h1394.o nx1394.o

838 HCI1394_OBJS += hci1394.o hci1394_async.o hci1394_attach.o hci1394_buf.o \
839                 hci1394_csr.o hci1394_detach.o hci1394_extern.o \
840                 hci1394_ioctl.o hci1394_isoch.o hci1394_isr.o \
841                 hci1394_ixl_comp.o hci1394_ixl_isr.o hci1394_ixl_misc.o \
842                 hci1394_ixl_update.o hci1394_misc.o hci1394_ohci.o \
843                 hci1394_q.o hci1394_s1394if.o hci1394_tlabel.o \
844                 hci1394_tlist.o hci1394_vendor.o

846 AV1394_OBJS +=  av1394.o av1394_as.o av1394_async.o av1394_cfgrom.o \
847                 av1394_cmp.o av1394_fcp.o av1394_isoch.o av1394_isoch_chan.o \
848                 av1394_isoch_recv.o av1394_isoch_xmit.o av1394_list.o \
849                 av1394_queue.o
```



```
851 DCAM1394_OBJS += dcam.o dcam_frame.o dcam_param.o dcam_reg.o \
852                 dcam_ring_buff.o

854 SCSA1394_OBJS += hba.o sbp2_driver.o sbp2_bus.o

856 SBP2_OBJS += cfgrom.o sbp2.o

858 PMODEM_OBJS += pmodem.o pmodem_cis.o cis.o cis_callout.o cis_handlers.o cis_para

860 DSW_OBJS +=     dsw.o dsw_dev.o ii_tree.o

862 NCALL_OBJS +=   ncall.o \
863                 ncall_stub.o

865 RDC_OBJS +=     rdc.o \
866                 rdc_dev.o \
867                 rdc_io.o \
868                 rdc_clnt.o \
869                 rdc_prot_xdr.o \
870                 rdc_svc.o \
871                 rdc_bitmap.o \
872                 rdc_health.o \
873                 rdc_subr.o \
874                 rdc_diskq.o

876 RDCSRV_OBJS +=  rdcsrv.o

878 RDCSTUB_OBJS += rdc_stub.o

880 SDBC_OBJS +=    sd_bcache.o \
881                 sd_bio.o \
882                 sd_conf.o \
883                 sd_ft.o \
884                 sd_hash.o \
885                 sd_io.o \
886                 sd_misc.o \
887                 sd_pcu.o \
888                 sd_tdaemon.o \
889                 sd_trace.o \
890                 sd_iob_impl0.o \
891                 sd_iob_impl1.o \
892                 sd_iob_impl2.o \
893                 sd_iob_impl3.o \
894                 sd_iob_impl4.o \
895                 sd_iob_impl5.o \
896                 sd_iob_impl6.o \
897                 sd_iob_impl7.o \
898                 safestore.o \
899                 safestore_ram.o

901 NSCTL_OBJS +=   nsctl.o \
902                 nsc_cache.o \
903                 nsc_disk.o \
904                 nsc_dev.o \
905                 nsc_freeze.o \
906                 nsc_gen.o \
907                 nsc_mem.o \
908                 nsc_ncallio.o \
909                 nsc_power.o \
910                 nsc_resv.o \
911                 nsc_rmspin.o \
912                 nsc_solaris.o \
913                 nsc_trap.o \
914                 nsc_list.o
915 UNISTAT_OBJS += spuni.o \
916                 spcs_s_k.o
```

```
NSKERN_OBJS +=  nsc_ddi.o \
                nsc_proc.o \
                nsc_raw.o \
                nsc_thread.o \
                nskernd.o

SV_OBJS +=      sv.o

PMCS_OBJS += pmcs_attach.o pmcs_ds.o pmcs_intr.o pmcs_nvram.o pmcs_sata.o \
                pmcs_scsa.o pmcs_smhba.o pmcs_subr.o pmcs_fwlog.o

PMCS8001FW_C_OBJS +=    pmcs_fw_hdr.o
PMCS8001FW_OBJS +=              $(PMCS8001FW_C_OBJS) SPCBoot.o ila.o firmware.o

#
#       Build up defines and paths.

ST_OBJS +=      st.o    st_conf.o

EMLXS_OBJS +=   emlxs_clock.o emlxs_dfc.o emlxs_dhchap.o emlxs_diag.o \
                emlxs_download.o emlxs_dump.o emlxs_els.o emlxs_event.o \
                emlxs_fcf.o emlxs_fcp.o emlxs_fct.o emlxs_hba.o emlxs_ip.o \
                emlxs_mbox.o emlxs_mem.o emlxs_msg.o emlxs_node.o \
                emlxs_pkt.o emlxs_sli3.o emlxs_sli4.o emlxs_solaris.o \
                emlxs_thread.o

EMLXS_FW_OBJS +=        emlxs_fw.o

OCE_OBJS +=     oce_buf.o oce_fm.o oce_gld.o oce_hw.o oce_intr.o oce_main.o \
                oce_mbx.o oce_mq.o oce_queue.o oce_rx.o oce_stat.o oce_tx.o \
                oce_utils.o

FCT_OBJS += discovery.o fct.o

QLT_OBJS += 2400.o 2500.o 8100.o qlt.o qlt_dma.o

SRPT_OBJS += srpt_mod.o srpt_ch.o srpt_cm.o srpt_ioc.o srpt_stp.o

FCOE_OBJS += fcoe.o fcoe_eth.o fcoe_fc.o

FCOET_OBJS += fcoet.o fcoet_eth.o fcoet_fc.o

FCOEI_OBJS += fcoei.o fcoei_eth.o fcoei_lv.o

ISCSIT_SHARED_OBJS += \
                iscsit_common.o

ISCSIT_OBJS +=  $(ISCSIT_SHARED_OBJS) \
                iscsit.o iscsit_tgt.o iscsit_sess.o iscsit_login.o \
                iscsit_text.o iscsit_isns.o iscsit_radiusauth.o \
                iscsit_radiuspacket.o iscsit_auth.o iscsit_authclient.o

PPPT_OBJS +=    alua_ic_if.o pppt.o pppt_msg.o pppt_tgt.o

STMF_OBJS += lun_map.o stmf.o

STMF_SBD_OBJS += sbd.o sbd_scsi.o sbd_pgr.o sbd_zvol.o

SYSMSG_OBJS +=  sysmsg.o

SES_OBJS +=     ses.o ses_sen.o ses_safte.o ses_ses.o

TNF_OBJS +=     tnf_buf.o       tnf_trace.o     tnf_writer.o    trace_init.o \
                trace_funcs.o   tnf_probe.o     tnf.o
```



```
LOGINDMUX_OBJS += logindmux.o

DEVINFO_OBJS += devinfo.o

DEVPOLL_OBJS += devpoll.o

DEVPOOL_OBJS += devpool.o

I8042_OBJS +=   i8042.o

KB8042_OBJS +=  \
                at_keyprocess.o \
                kb8042.o        \
                kb8042_keytables.o

MOUSE8042_OBJS += mouse8042.o

FDC_OBJS +=     fdc.o

ASY_OBJS +=     asy.o

ECPP_OBJS +=    ecpp.o

VUIDM3P_OBJS += vuidmice.o vuidm3p.o

VUIDM4P_OBJS += vuidmice.o vuidm4p.o

VUIDM5P_OBJS += vuidmice.o vuidm5p.o

VUIDPS2_OBJS += vuidmice.o vuidps2.o

HPCSVC_OBJS += hpcsvc.o

PCIE_MISC_OBJS += pcie.o pcie_fault.o pcie_hp.o pciehpc.o pcishpc.o pcie_pwr.o p

PCIHPNEXUS_OBJS += pcihp.o

OPENEEPR_OBJS += openprom.o

RANDOM_OBJS += random.o

PSHOT_OBJS += pshot.o

GEN_DRV_OBJS += gen_drv.o

TCLIENT_OBJS += tclient.o

TPHCI_OBJS += tphci.o

TVHCI_OBJS += tvhci.o

EMUL64_OBJS += emul64.o emul64_bsd.o

FCP_OBJS += fcp.o

FCIP_OBJS += fcip.o

FCSM_OBJS += fcsm.o

FCTL_OBJS += fctl.o

FP_OBJS += fp.o

QLC_OBJS += ql_api.o ql_debug.o ql_hba_fru.o ql_init.o ql_iocb.o ql_ioctl.o \
        ql_isr.o ql_mbx.o ql_nx.o ql_xioctl.o ql_fw_table.o
```

```
1049 QLC_FW_2200_OBJS += ql_fw_2200.o

1051 QLC_FW_2300_OBJS += ql_fw_2300.o

1053 QLC_FW_2400_OBJS += ql_fw_2400.o

1055 QLC_FW_2500_OBJS += ql_fw_2500.o

1057 QLC_FW_6322_OBJS += ql_fw_6322.o

1059 QLC_FW_8100_OBJS += ql_fw_8100.o

1061 QLGE_OBJS += qlge.o qlge_dbg.o qlge_flash.o qlge_fm.o qlge_gld.o qlge_mpi.o

1063 ZCONS_OBJS += zcons.o

1065 NV_SATA_OBJS += nv_sata.o

1067 SI3124_OBJS += si3124.o

1069 AHCI_OBJS += ahci.o

1071 PCIIDE_OBJS += pci-ide.o

1073 PCEPP_OBJS += pcepp.o

1075 CPC_OBJS += cpc.o

1077 CPUID_OBJS += cpuid_drv.o

1079 SYSEVENT_OBJS += sysevent.o

1081 BL_OBJS += bl.o

1083 DRM_OBJS += drm_sunmod.o drm_kstat.o drm_agpsupport.o \
1084             drm_auth.o drm_bufs.o drm_context.o drm_dma.o \
1085             drm_drawable.o drm_drv.o drm_fops.o drm_ioctl.o drm_irq.o \
1086             drm_lock.o drm_memory.o drm_msg.o drm_pci.o drm_scatter.o \
1087             drm_cache.o drm_gem.o drm_mm.o ati_pcigart.o

1089 FM_OBJS += devfm.o devfm_machdep.o

1091 RTLS_OBJS +=    rtls.o

1093 #
1094 #                       exec modules
1095 #
1096 AOUTEXEC_OBJS +=aout.o

1098 ELFEXEC_OBJS += elf.o elf_notes.o old_notes.o

1100 INTPEXEC_OBJS +=intp.o

1102 SHBINEXEC_OBJS +=shbin.o

1104 JAVAEXEC_OBJS +=java.o

1106 #
1107 #                       file system modules
1108 #
1109 AUTOFS_OBJS +=  auto_vfsops.o auto_vnops.o auto_subr.o auto_xdr.o auto_sys.o

1111 CACHEFS_OBJS += cachefs_cnode.o         cachefs_cod.o \
1112                 cachefs_dir.o           cachefs_dlog.o  cachefs_filegrp.o \
1113                 cachefs_fscache.o       cachefs_ioctl.o cachefs_log.o \
1114                 cachefs_module.o \
```



```
1115                 cachefs_noopc.o         cachefs_resource.o \
1116                 cachefs_strict.o \
1117                 cachefs_subr.o          cachefs_vfsops.o \
1118                 cachefs_vnops.o

1120 DCFS_OBJS +=    dc_vnops.o

1122 DEVFS_OBJS +=   devfs_subr.o    devfs_vfsops.o  devfs_vnops.o

1124 DEV_OBJS  +=    sdev_subr.o     sdev_vfsops.o   sdev_vnops.o    \
1125                 sdev_ptsops.o   sdev_zvolops.o  sdev_comm.o     \
1126                 sdev_profile.o  sdev_ncache.o   sdev_netops.o   \
1127                 sdev_ipnetops.o \
1128                 sdev_vtops.o

1130 CTFS_OBJS +=    ctfs_all.o ctfs_cdir.o ctfs_ctl.o ctfs_event.o \
1131                 ctfs_latest.o ctfs_root.o ctfs_sym.o ctfs_tdir.o ctfs_tmpl.o

1133 OBJFS_OBJS +=   objfs_vfs.o     objfs_root.o    objfs_common.o \
1134                 objfs_odir.o    objfs_data.o

1136 FDFS_OBJS +=    fdops.o

1138 FIFO_OBJS +=    fifosubr.o      fifovnops.o

1140 PIPE_OBJS +=    pipe.o

1142 HSFS_OBJS +=    hsfs_node.o     hsfs_subr.o     hsfs_vfsops.o   hsfs_vnops.o \
1143                 hsfs_susp.o     hsfs_rrip.o     hsfs_susp_subr.o

1145 LOFS_OBJS +=    lofs_subr.o     lofs_vfsops.o   lofs_vnops.o

1147 NAMEFS_OBJS +=  namevfs.o       namevno.o

1149 NFS_OBJS +=     nfs_client.o    nfs_common.o    nfs_dump.o \
1150                 nfs_subr.o      nfs_vfsops.o    nfs_vnops.o \
1151                 nfs_xdr.o       nfs_sys.o       nfs_strerror.o \
1152                 nfs3_vfsops.o   nfs3_vnops.o    nfs3_xdr.o \
1153                 nfs_acl_vnops.o nfs_acl_xdr.o   nfs4_vfsops.o \
1154                 nfs4_vnops.o    nfs4_xdr.o      nfs4_idmap.o \
1155                 nfs4_shadow.o   nfs4_subr.o \
1156                 nfs4_attr.o     nfs4_rnode.o    nfs4_client.o \
1157                 nfs4_acache.o   nfs4_common.o   nfs4_client_state.o \
1158                 nfs4_callback.o nfs4_recovery.o nfs4_client_secinfo.o \
1159                 nfs4_client_debug.o     nfs_stats.o \
1160                 nfs4_acl.o      nfs4_stub_vnops.o       nfs_cmd.o

1162 NFSSRV_OBJS +=  nfs_server.o    nfs_srv.o       nfs3_srv.o \
1163                 nfs_acl_srv.o   nfs_auth.o      nfs_auth_xdr.o \
1164                 nfs_export.o    nfs_log.o       nfs_log_xdr.o \
1165                 nfs4_srv.o      nfs4_state.o    nfs4_srv_attr.o \
1166                 nfs4_srv_ns.o   nfs4_db.o       nfs4_srv_deleg.o \
1167                 nfs4_deleg_ops.o nfs4_srv_readdir.o nfs4_dispatch.o

1169 SMBSRV_SHARED_OBJS += \
1170                 smb_inet.o \
1171                 smb_match.o \
1172                 smb_msgbuf.o \
1173                 smb_oem.o \
1174                 smb_string.o \
1175                 smb_utf8.o \
1176                 smb_door_legacy.o \
1177                 smb_xdr.o \
1178                 smb_token.o \
1179                 smb_token_xdr.o \
1180                 smb_sid.o \
```

```
1181                smb_native.o \
1182                smb_netbios_util.o

1184 SMBSRV_OBJS +=  $(SMBSRV_SHARED_OBJS)                  \
1185                smb_acl.o                               \
1186                smb_alloc.o                             \
1187                smb_close.o                             \
1188                smb_common_open.o                       \
1189                smb_common_transact.o                   \
1190                smb_create.o                            \
1191                smb_delete.o                            \
1192                smb_directory.o                         \
1193                smb_dispatch.o                          \
1194                smb_echo.o                              \
1195                smb_fem.o                               \
1196                smb_find.o                              \
1197                smb_flush.o                             \
1198                smb_fsinfo.o                            \
1199                smb_fsops.o                             \
1200                smb_init.o                              \
1201                smb_kdoor.o                             \
1202                smb_kshare.o                            \
1203                smb_kutil.o                             \
1204                smb_lock.o                              \
1205                smb_lock_byte_range.o                   \
1206                smb_locking_andx.o                      \
1207                smb_logoff_andx.o                       \
1208                smb_mangle_name.o                       \
1209                smb_mbuf_marshaling.o                   \
1210                smb_mbuf_util.o                         \
1211                smb_negotiate.o                         \
1212                smb_net.o                               \
1213                smb_node.o                              \
1214                smb_nt_cancel.o                         \
1215                smb_nt_create_andx.o                    \
1216                smb_nt_transact_create.o                \
1217                smb_nt_transact_ioctl.o                 \
1218                smb_nt_transact_notify_change.o         \
1219                smb_nt_transact_quota.o                 \
1220                smb_nt_transact_security.o              \
1221                smb_odir.o                              \
1222                smb_ofile.o                             \
1223                smb_open_andx.o                         \
1224                smb_opipe.o                             \
1225                smb_oplock.o                            \
1226                smb_pathname.o                          \
1227                smb_print.o                             \
1228                smb_process_exit.o                      \
1229                smb_query_fileinfo.o                    \
1230                smb_read.o                              \
1231                smb_rename.o                            \
1232                smb_sd.o                                \
1233                smb_seek.o                              \
1234                smb_server.o                            \
1235                smb_session.o                           \
1236                smb_session_setup_andx.o                \
1237                smb_set_fileinfo.o                      \
1238                smb_signing.o                           \
1239                smb_tree.o                              \
1240                smb_trans2_create_directory.o           \
1241                smb_trans2_dfs.o                        \
1242                smb_trans2_find.o                       \
1243                smb_tree_connect.o                      \
1244                smb_unlock_byte_range.o                 \
1245                smb_user.o                              \
1246                smb_vfs.o                               \
```



```
1247                smb_vops.o                              \
1248                smb_vss.o                               \
1249                smb_write.o

1251 PCFS_OBJS +=    pc_alloc.o      pc_dir.o        pc_node.o       pc_subr.o \
1252                 pc_vfsops.o     pc_vnops.o

1254 PROC_OBJS +=    prcontrol.o     prioctl.o       prsubr.o        prusrio.o \
1255                 prvfsops.o      prvnops.o

1257 MNTFS_OBJS +=   mntvfsops.o     mntvnops.o

1259 SHAREFS_OBJS += sharetab.o      sharefs_vfsops.o        sharefs_vnops.o

1261 SPEC_OBJS +=    specsubr.o      specvfsops.o    specvnops.o

1263 SOCK_OBJS +=    socksubr.o      sockvfsops.o    sockparams.o    \
1264                 socksyscalls.o  socktpi.o       sockstr.o \
1265                 sockcommon_vnops.o      sockcommon_subr.o \
1266                 sockcommon_sops.o       sockcommon.o    \
1267                 sock_notsupp.o  socknotify.o \
1268                 nl7c.o          nl7curi.o       nl7chttp.o      nl7clogd.o \
1269                 nl7cnca.o       sodirect.o      sockfilter.o

1271 TMPFS_OBJS +=   tmp_dir.o       tmp_subr.o      tmp_tnode.o     tmp_vfsops.o \
1272                 tmp_vnops.o

1274 UDFS_OBJS +=    udf_alloc.o     udf_bmap.o      udf_dir.o       \
1275                 udf_inode.o     udf_subr.o      udf_vfsops.o    \
1276                 udf_vnops.o

1278 UFS_OBJS +=     ufs_alloc.o     ufs_bmap.o      ufs_dir.o       ufs_xattr.o \
1279                 ufs_inode.o     ufs_subr.o      ufs_tables.o    ufs_vfsops.o \
1280                 ufs_vnops.o     quota.o         quotacalls.o    quota_ufs.o \
1281                 ufs_filio.o     ufs_lockfs.o    ufs_thread.o    ufs_trans.o \
1282                 ufs_acl.o       ufs_panic.o     ufs_directio.o  ufs_log.o \
1283                 ufs_extvnops.o  ufs_snap.o      lufs.o          lufs_thread.o \
1284                 lufs_log.o      lufs_map.o      lufs_top.o      lufs_debug.o
1285 VSCAN_OBJS +=   vscan_drv.o     vscan_svc.o vscan_door.o

1287 NSMB_OBJS +=    smb_conn.o      smb_dev.o       smb_iod.o       smb_pass.o \
1288                 smb_rq.o        smb_sign.o      smb_smb.o       smb_subrs.o \
1289                 smb_time.o      smb_tran.o      smb_trantcp.o   smb_usr.o \
1290                 subr_mchain.o

1292 SMBFS_COMMON_OBJS += smbfs_ntacl.o
1293 SMBFS_OBJS +=    smbfs_vfsops.o  smbfs_vnops.o   smbfs_node.o    \
1294                 smbfs_acl.o     smbfs_client.o  smbfs_smb.o     \
1295                 smbfs_subr.o    smbfs_subr2.o   \
1296                 smbfs_rwlock.o  smbfs_xattr.o   \
1297                 $(SMBFS_COMMON_OBJS)


1300 #
1301 #                       LVM modules
1302 #
1303 MD_OBJS += md.o md_error.o md_ioctl.o md_mddb.o md_names.o \
1304         md_med.o md_rename.o md_subr.o

1306 MD_COMMON_OBJS = md_convert.o md_crc.o md_revchk.o

1308 MD_DERIVED_OBJS = metamed_xdr.o meta_basic_xdr.o

1310 SOFTPART_OBJS += sp.o sp_ioctl.o

1312 STRIPE_OBJS += stripe.o stripe_ioctl.o
```

```
1314 HOTSPARES_OBJS += hotspares.o

1316 RAID_OBJS += raid.o raid_ioctl.o raid_replay.o raid_resync.o raid_hotspare.o

1318 MIRROR_OBJS += mirror.o mirror_ioctl.o mirror_resync.o

1320 NOTIFY_OBJS += md_notify.o

1322 TRANS_OBJS += mdtrans.o trans_ioctl.o trans_log.o

1324 ZFS_COMMON_OBJS +=              \
1325         arc.o                   \
1326         blkptr.o                \
1327         bplist.o                \
1328         bpobj.o                 \
1329         bptree.o                \
1330         dbuf.o                  \
1331         ddt.o                   \
1332         ddt_zap.o               \
1333         dmu.o                   \
1334         dmu_diff.o              \
1335         dmu_send.o              \
1336         dmu_object.o            \
1337         dmu_objset.o            \
1338         dmu_traverse.o          \
1339         dmu_tx.o                \
1340         dnode.o                 \
1341         dnode_sync.o            \
1342         dsl_bookmark.o          \
1343         dsl_dir.o               \
1344         dsl_dataset.o           \
1345         dsl_deadlist.o          \
1346         dsl_destroy.o           \
1347         dsl_pool.o              \
1348         dsl_synctask.o          \
1349         dsl_userhold.o          \
1350         dmu_zfetch.o            \
1351         dsl_deleg.o             \
1352         dsl_prop.o              \
1353         dsl_scan.o              \
1354         zfeature.o              \
1355         gzip.o                  \
1356         lz4.o                   \
1357         lzjb.o                  \
1358         metaslab.o              \
1359         multilist.o             \
1360         range_tree.o            \
1361         refcount.o              \
1362         rrwlock.o               \
1363         sa.o                    \
1364         sha256.o                \
1365         spa.o                   \
1366         spa_config.o            \
1367         spa_errlog.o            \
1368         spa_history.o           \
1369         spa_misc.o              \
1370         space_map.o             \
1371         space_reftree.o         \
1372         txg.o                   \
1373         uberblock.o             \
1374         unique.o                \
1375         vdev.o                  \
1376         vdev_cache.o            \
1377         vdev_file.o             \
1378         vdev_label.o            \
```



```
1379         vdev_mirror.o           \
1380         vdev_missing.o          \
1381         vdev_queue.o            \
1382         vdev_raidz.o            \
1383         vdev_root.o             \
1384         zap.o                   \
1385         zap_leaf.o              \
1386         zap_micro.o             \
1387         zfs_byteswap.o          \
1388         zfs_debug.o             \
1389         zfs_fm.o                \
1390         zfs_fuid.o              \
1391         zfs_sa.o                \
1392         zfs_znode.o             \
1393         zil.o                   \
1394         zio.o                   \
1395         zio_checksum.o          \
1396         zio_compress.o          \
1397         zio_inject.o            \
1398         zle.o                   \
1399         zrlock.o

1401 ZFS_SHARED_OBJS +=              \
1402         zfeature_common.o       \
1403         zfs_comutil.o           \
1404         zfs_deleg.o             \
1405         zfs_fletcher.o          \
1406         zfs_namecheck.o         \
1407         zfs_prop.o              \
1408         zpool_prop.o            \
1409         zprop_common.o

1411 ZFS_OBJS +=                     \
1412         $(ZFS_COMMON_OBJS)      \
1413         $(ZFS_SHARED_OBJS)      \
1414         vdev_disk.o             \
1415         zfs_acl.o               \
1416         zfs_ctldir.o            \
1417         zfs_dir.o               \
1418         zfs_ioctl.o             \
1419         zfs_log.o               \
1420         zfs_onexit.o            \
1421         zfs_replay.o            \
1422         zfs_rlock.o             \
1423         zfs_vfsops.o            \
1424         zfs_vnops.o             \
1425         zvol.o

1427 ZUT_OBJS +=                     \
1428         zut.o

1430 #
1431 #                       streams modules
1432 #
1433 BUFMOD_OBJS     +=      bufmod.o

1435 CONNLD_OBJS +=  connld.o

1437 DEDUMP_OBJS +=  dedump.o

1439 DRCOMPAT_OBJS +=        drcompat.o

1441 LDLINUX_OBJS += ldlinux.o

1443 LDTERM_OBJS +=  ldterm.o uwidth.o
```

```
1445 PCKT_OBJS +=    pckt.o

1447 PFMOD_OBJS +=   pfmod.o

1449 PTEM_OBJS +=    ptem.o

1451 REDIRMOD_OBJS += strredirm.o

1453 TIMOD_OBJS +=   timod.o

1455 TIRDWR_OBJS +=  tirdwr.o

1457 TTCOMPAT_OBJS +=ttcompat.o

1459 LOG_OBJS +=     log.o

1461 PIPEMOD_OBJS += pipemod.o

1463 RPCMOD_OBJS +=  rpcmod.o        clnt_cots.o     clnt_clts.o \
1464                 clnt_gen.o      clnt_perr.o     mt_rpcinit.o    rpc_calmsg.o \
1465                 rpc_prot.o      rpc_sztypes.o   rpc_subr.o      rpcb_prot.o \
1466                 svc.o           svc_clts.o      svc_gen.o       svc_cots.o \
1467                 rpcsys.o        xdr_sizeof.o    clnt_rdma.o     svc_rdma.o \
1468                 xdr_rdma.o      rdma_subr.o     xdrrdma_sizeof.o

1470 KLMMOD_OBJS +=  klmmod.o \
1471                 nlm_impl.o \
1472                 nlm_rpc_handle.o \
1473                 nlm_dispatch.o \
1474                 nlm_rpc_svc.o \
1475                 nlm_client.o \
1476                 nlm_service.o \
1477                 nlm_prot_clnt.o \
1478                 nlm_prot_xdr.o \
1479                 nlm_rpc_clnt.o \
1480                 nsm_addr_clnt.o \
1481                 nsm_addr_xdr.o \
1482                 sm_inter_clnt.o \
1483                 sm_inter_xdr.o

1485 KLMOPS_OBJS +=  klmops.o

1487 TLIMOD_OBJS +=  tlimod.o        t_kalloc.o      t_kbind.o       t_kclose.o \
1488                 t_kconnect.o    t_kfree.o       t_kgtstate.o    t_kopen.o \
1489                 t_krcvudat.o    t_ksndudat.o    t_kspoll.o      t_kunbind.o \
1490                 t_kutil.o

1492 RLMOD_OBJS += rlmod.o

1494 TELMOD_OBJS += telmod.o

1496 CRYPTMOD_OBJS += cryptmod.o

1498 KB_OBJS +=      kbd.o           keytables.o

1500 #
1501 #                       ID mapping module
1502 #
1503 IDMAP_OBJS +=   idmap_mod.o     idmap_kapi.o    idmap_xdr.o     idmap_cache.o

1505 #
1506 #                       scheduling class modules
1507 #
1508 SDC_OBJS +=             sysdc.o

1510 RT_OBJS +=              rt.o
```



```
1511 RT_DPTBL_OBJS +=        rt_dptbl.o

1513 TS_OBJS +=              ts.o
1514 TS_DPTBL_OBJS +=        ts_dptbl.o

1516 IA_OBJS +=              ia.o

1518 FSS_OBJS +=             fss.o

1520 FX_OBJS +=              fx.o
1521 FX_DPTBL_OBJS +=        fx_dptbl.o

1523 #
1524 #                       Inter-Process Communication (IPC) modules
1525 #
1526 IPC_OBJS +=     ipc.o

1528 IPCMSG_OBJS +=  msg.o

1530 IPCSEM_OBJS +=  sem.o

1532 IPCSHM_OBJS +=  shm.o

1534 #
1535 #                       bignum module
1536 #
1537 COMMON_BIGNUM_OBJS += bignum_mod.o bignumimpl.o

1539 BIGNUM_OBJS += $(COMMON_BIGNUM_OBJS) $(BIGNUM_PSR_OBJS)

1541 #
1542 #                       kernel cryptographic framework
1543 #
1544 KCF_OBJS +=     kcf.o kcf_callprov.o kcf_cbufcall.o kcf_cipher.o kcf_crypto.o \
1545                 kcf_cryptoadm.o kcf_ctxops.o kcf_digest.o kcf_dual.o \
1546                 kcf_keys.o kcf_mac.o kcf_mech_tabs.o kcf_miscapi.o \
1547                 kcf_object.o kcf_policy.o kcf_prov_lib.o kcf_prov_tabs.o \
1548                 kcf_sched.o kcf_session.o kcf_sign.o kcf_spi.o kcf_verify.o \
1549                 kcf_random.o modes.o ecb.o cbc.o ctr.o ccm.o gcm.o \
1550                 fips_random.o

1552 CRYPTOADM_OBJS += cryptoadm.o

1554 CRYPTO_OBJS +=  crypto.o

1556 DPROV_OBJS +=   dprov.o

1558 DCA_OBJS +=     dca.o dca_3des.o dca_debug.o dca_dsa.o dca_kstat.o dca_rng.o \
1559                 dca_rsa.o

1561 AESPROV_OBJS += aes.o aes_impl.o aes_modes.o

1563 ARCFOURPROV_OBJS += arcfour.o arcfour_crypt.o

1565 BLOWFISHPROV_OBJS += blowfish.o blowfish_impl.o

1567 ECCPROV_OBJS += ecc.o ec.o ec2_163.o ec2_mont.o ecdecode.o ecl_mult.o \
1568                 ecp_384.o ecp_jac.o ec2_193.o ecl.o ecp_192.o ecp_521.o \
1569                 ecp_jm.o ec2_233.o ecl_curve.o ecp_224.o ecp_aff.o \
1570                 ecp_mont.o ec2_aff.o ec_naf.o ecl_gf.o ecp_256.o mp_gf2m.o \
1571                 mpi.o mplogic.o mpmontg.o mpprime.o oid.o \
1572                 secitem.o ec2_test.o ecp_test.o

1574 RSAPROV_OBJS += rsa.o rsa_impl.o pkcs1.o

1576 SWRANDPROV_OBJS += swrand.o
```

```
#
#                       kernel SSL
#
KSSL_OBJS +=    kssl.o kssliоctl.o

KSSL_SOCKFIL_MOD_OBJS += ksslfilter.o ksslapi.o ksslrec.o

#
#                       misc. modules
#

C2AUDIT_OBJS += adr.o audit.o audit_event.o audit_io.o \
                audit_path.o audit_start.o audit_syscalls.o audit_token.o \
                audit_mem.o

PCIC_OBJS +=    pcic.o

RPCSEC_OBJS +=  secmod.o        sec_clnt.o      sec_svc.o       sec_gen.o \
                auth_des.o      auth_kern.o     auth_none.o     auth_loopb.o\
                authdesprt.o    authdesubr.o    authu_prot.o \
                key_call.o      key_prot.o      svc_authu.o     svcauthdes.o

RPCSEC_GSS_OBJS +=      rpcsec_gssmod.o rpcsec_gss.o rpcsec_gss_misc.o \
                rpcsec_gss_utils.o svc_rpcsec_gss.o

CONSCONFIG_OBJS += consconfig.o

CONSCONFIG_DACF_OBJS  += consconfig_dacf.o consplat.o

TEM_OBJS += tem.o tem_safe.o 6x10.o 7x14.o 12x22.o

KBTRANS_OBJS +=                         \
                kbtrans.o               \
                kbtrans_keytables.o     \
                kbtrans_polled.o        \
                kbtrans_streams.o       \
                usb_keytables.o

KGSSD_OBJS +=   gssd_clnt_stubs.o gssd_handle.o gssd_prot.o \
                gss_display_name.o gss_release_name.o gss_import_name.o \
                gss_release_buffer.o gss_release_oid_set.o gen_oids.o gssdmod.o

KGSSD_DERIVED_OBJS = gssd_xdr.o

KGSS_DUMMY_OBJS += dmech.o

KSOCKET_OBJS += ksocket.o ksocket_mod.o

CRYPTO= cksumtypes.o decrypt.o encrypt.o encrypt_length.o etypes.o \
        nfold.o verify_checksum.o prng.o block_size.o make_checksum.o\
        checksum_length.o hmac.o default_state.o mandatory_sumtype.o

# crypto/des
CRYPTO_DES= f_cbc.o f_cksum.o f_parity.o weak_key.o d3_cbc.o ef_crypto.o

CRYPTO_DK= checksum.o derive.o dk_decrypt.o dk_encrypt.o

CRYPTO_ARCFOUR= k5_arcfour.o

# crypto/enc_provider
CRYPTO_ENC= des.o des3.o arcfour_provider.o aes_provider.o

# crypto/hash_provider
CRYPTO_HASH= hash_kef_generic.o hash_kmd5.o hash_crc32.o hash_ksha1.o
```



```
# crypto/keyhash_provider
CRYPTO_KEYHASH= descbc.o k5_kmd5des.o k_hmac_md5.o

# crypto/crc32
CRYPTO_CRC32= crc32.o

# crypto/old
CRYPTO_OLD= old_decrypt.o old_encrypt.o

# crypto/raw
CRYPTO_RAW= raw_decrypt.o raw_encrypt.o

K5_KRB= kfree.o copy_key.o \
        parse.o init_ctx.o \
        ser_adata.o ser_addr.o \
        ser_auth.o ser_cksum.o \
        ser_key.o ser_princ.o \
        serialize.o unparse.o \
        ser_actx.o

K5_OS=  timeofday.o toffset.o \
        init_os_ctx.o c_ustime.o

SEAL=   seal.o unseal.o

MECH=   delete_sec_context.o \
        import_sec_context.o \
        gssapi_krb5.o \
        k5seal.o k5unseal.o k5sealv3.o \
        ser_sctx.o \
        sign.o \
        util_crypt.o  \
        util_validate.o  util_ordering.o  \
        util_seqnum.o util_set.o util_seed.o \
        wrap_size_limit.o verify.o



MECH_GEN= util_token.o


KGSS_KRB5_OBJS += krb5mech.o \
        $(MECH) $(SEAL) $(MECH_GEN) \
        $(CRYPTO) $(CRYPTO_DES) $(CRYPTO_DK) $(CRYPTO_ARCFOUR) \
        $(CRYPTO_ENC) $(CRYPTO_HASH) \
        $(CRYPTO_KEYHASH) $(CRYPTO_CRC32) \
        $(CRYPTO_OLD) \
        $(CRYPTO_RAW) $(K5_KRB) $(K5_OS)

DES_OBJS +=     des_crypt.o des_impl.o des_ks.o des_soft.o

DLBOOT_OBJS +=  bootparam_xdr.o nfs_dlinet.o scan.o

KRTLD_OBJS +=   kobj_bootflags.o getoptstr.o \
                kobj.o kobj_kdi.o kobj_lm.o kobj_subr.o

MOD_OBJS +=     modctl.o modsubr.o modsysfile.o modconf.o modhash.o

STRPLUMB_OBJS += strplumb.o

CPR_OBJS +=     cpr_driver.o cpr_dump.o \
                cpr_main.o cpr_misc.o cpr_mod.o cpr_stat.o \
                cpr_uthread.o

PROF_OBJS +=    prf.o
```

```
1709 SE_OBJS += se_driver.o

1711 SYSACCT_OBJS += acct.o

1713 ACCTCTL_OBJS += acctctl.o

1715 EXACCTSYS_OBJS += exacctsys.o

1717 KAIO_OBJS += aio.o

1719 PCMCIA_OBJS += pcmcia.o cs.o cis.o cis_callout.o cis_handlers.o cis_params.o

1721 BUSRA_OBJS += busra.o

1723 PCS_OBJS += pcs.o

1725 PSET_OBJS +=    pset.o

1727 OHCI_OBJS += ohci.o ohci_hub.o ohci_polled.o

1729 UHCI_OBJS += uhci.o uhciutil.o uhcitgt.o uhcihub.o uhcipolled.o

1731 EHCI_OBJS += ehci.o ehci_hub.o ehci_xfer.o ehci_intr.o ehci_util.o ehci_polled.o

1733 HUBD_OBJS += hubd.o

1735 USB_MID_OBJS += usb_mid.o

1737 USB_IA_OBJS += usb_ia.o

1739 SCSA2USB_OBJS += scsa2usb.o usb_ms_bulkonly.o usb_ms_cbi.o

1741 IPF_OBJS += ip_fil_solaris.o fil.o solaris.o ip_state.o ip_frag.o ip_nat.o \
1742             ip_proxy.o ip_auth.o ip_pool.o ip_htable.o ip_lookup.o \
1743             ip_log.o misc.o ip_compat.o ip_nat6.o drand48.o

1745 IPD_OBJS += ipd.o

1747 IBD_OBJS +=     ibd.o ibd_cm.o

1749 EIBNX_OBJS +=   enx_main.o enx_hdlrs.o enx_ibt.o enx_log.o enx_fip.o \
1750                 enx_misc.o enx_q.o enx_ctl.o

1752 EOIB_OBJS +=    eib_adm.o eib_chan.o eib_cmn.o eib_ctl.o eib_data.o \
1753                 eib_fip.o eib_ibt.o eib_log.o eib_mac.o eib_main.o \
1754                 eib_rsrc.o eib_svc.o eib_vnic.o

1756 DLPISTUB_OBJS += dlpistub.o

1758 SDP_OBJS +=     sdpddi.o

1760 TRILL_OBJS +=   trill.o

1762 CTF_OBJS += ctf_create.o ctf_decl.o ctf_error.o ctf_hash.o ctf_labels.o \
1763         ctf_lookup.o ctf_open.o ctf_types.o ctf_util.o ctf_subr.o ctf_mod.o

1765 SMBIOS_OBJS += smb_error.o smb_info.o smb_open.o smb_subr.o smb_dev.o

1767 RPCIB_OBJS += rpcib.o

1769 KMDB_OBJS += kdrv.o

1771 AFE_OBJS += afe.o

1773 BGE_OBJS += bge_main2.o bge_chip2.o bge_kstats.o bge_log.o bge_ndd.o \
1774                 bge_atomic.o bge_mii.o bge_send.o bge_recv2.o bge_mii_5906.o
```



```
1776 DMFE_OBJS += dmfe_log.o dmfe_main.o dmfe_mii.o

1778 EFE_OBJS += efe.o

1780 ELXL_OBJS += elxl.o

1782 HME_OBJS += hme.o

1784 IXGB_OBJS += ixgb.o ixgb_atomic.o ixgb_chip.o ixgb_gld.o ixgb_kstats.o \
1785                 ixgb_log.o ixgb_ndd.o ixgb_rx.o ixgb_tx.o ixgb_xmii.o

1787 NGE_OBJS += nge_main.o nge_atomic.o nge_chip.o nge_ndd.o nge_kstats.o \
1788                 nge_log.o nge_rx.o nge_tx.o nge_xmii.o

1790 PCN_OBJS += pcn.o

1792 RGE_OBJS += rge_main.o rge_chip.o rge_ndd.o rge_kstats.o rge_log.o rge_rxtx.o

1794 URTW_OBJS += urtw.o

1796 ARN_OBJS += arn_hw.o arn_eeprom.o arn_mac.o arn_calib.o arn_ani.o arn_phy.o arn_
1797                 arn_main.o arn_recv.o arn_xmit.o arn_rc.o

1799 ATH_OBJS += ath_aux.o ath_main.o ath_osdep.o ath_rate.o

1801 ATU_OBJS += atu.o

1803 IPW_OBJS += ipw2100_hw.o ipw2100.o

1805 IWI_OBJS += ipw2200_hw.o ipw2200.o

1807 IWH_OBJS += iwh.o

1809 IWK_OBJS += iwk2.o

1811 IWP_OBJS += iwp.o

1813 MWL_OBJS += mwl.o

1815 MWLFW_OBJS += mwlfw_mode.o

1817 WPI_OBJS += wpi.o

1819 RAL_OBJS += rt2560.o ral_rate.o

1821 RUM_OBJS += rum.o

1823 RWD_OBJS += rt2661.o

1825 RWN_OBJS += rt2860.o

1827 UATH_OBJS += uath.o

1829 UATHFW_OBJS += uathfw_mod.o

1831 URAL_OBJS += ural.o

1833 RTW_OBJS += rtw.o smc93cx6.o rtwphy.o rtwphyio.o

1835 ZYD_OBJS += zyd.o zyd_usb.o zyd_hw.o zyd_fw.o

1837 MXFE_OBJS += mxfe.o

1839 MPTSAS_OBJS += mptsas.o mptsas_hash.o mptsas_impl.o mptsas_init.o \
1840                 mptsas_raid.o mptsas_smhba.o
```

```
1842 SFE_OBJS += sfe.o sfe_util.o

1844 BFE_OBJS += bfe.o

1846 BRIDGE_OBJS += bridge.o

1848 IDM_SHARED_OBJS += base64.o

1850 IDM_OBJS +=     $(IDM_SHARED_OBJS) \
1851                 idm.o idm_impl.o idm_text.o idm_conn_sm.o idm_so.o

1853 VR_OBJS += vr.o

1855 ATGE_OBJS += atge_main.o atge_l1e.o atge_mii.o atge_l1.o atge_l1c.o

1857 YGE_OBJS = yge.o

1859 SKD_OBJS = skd.o

1861 #
1862 #       Build up defines and paths.
1863 #
1864 LINT_DEFS       += -Dunix

1866 #
1867 #       This duality can be removed when the native and target compilers
1868 #       are the same (or at least recognize the same command line syntax!)
1869 #       It is a bug in the current compilation system that the assember
1870 #       can't process the -Y I, flag.
1871 #
1872 NATIVE_INC_PATH += $(INC_PATH) $(CCYFLAG)$(UTSBASE)/common
1873 AS_INC_PATH     += $(INC_PATH) -I$(UTSBASE)/common
1874 INCLUDE_PATH    += $(INC_PATH) $(CCYFLAG)$(UTSBASE)/common

1876 PCIEB_OBJS += pcieb.o

1878 #       Chelsio N110 10G NIC driver module
1879 #
1880 CH_OBJS = ch.o glue.o pe.o sge.o

1882 CH_COM_OBJS =   ch_mac.o ch_subr.o cspi.o espi.o ixf1010.o mc3.o mc4.o mc5.o \
1883                 mv88e1xxx.o mv88x201x.o my3126.o pm3393.o tp.o ulp.o \
1884                 vsc7321.o vsc7326.o xpak.o

1886 #
1887 #       Chelsio Terminator 4 10G NIC nexus driver module
1888 #
1889 CXGBE_FW_OBJS  =        t4_fw.o t4_cfg.o
1890 CXGBE_COM_OBJS =        t4_hw.o common.o
1891 CXGBE_NEX_OBJS =        t4_nexus.o t4_sge.o t4_mac.o t4_ioctl.o shared.o \
1892                         t4_l2t.o adapter.o osdep.o

1894 #
1895 #       Chelsio Terminator 4 10G NIC driver module
1896 #
1897 CXGBE_OBJS =    cxgbe.o

1899 #
1900 #       PCI strings file
1901 #
1902 PCI_STRING_OBJS = pci_strings.o

1904 NET_DACF_OBJS += net_dacf.o

1906 #
```



```
1907 #       Xframe 10G NIC driver module
1908 #
1909 XGE_OBJS = xge.o xgell.o

1911 XGE_HAL_OBJS =  xgehal-channel.o xgehal-fifo.o xgehal-ring.o  xgehal-config.o \
1912                 xgehal-driver.o  xgehal-mm.o xgehal-stats.o  xgehal-device.o \
1913                 xge-queue.o  xgehal-mgmt.o xgehal-mgmtaux.o

1915 #
1916 #       e1000/igb common objs
1917 #
1918 #       Historically e1000g and igb had separate copies of all of the common
1919 #       code. At this time while they are now sharing the same copy of it, they
1920 #       are building it into their own modules which is due to the differences
1921 #       in the osdep and debug portions of their code.
1922 #
1923 E1000API_OBJS += e1000_80003es2lan.o e1000_82540.o e1000_82541.o e1000_82542.o \
1924                 e1000_82543.o e1000_82571.o e1000_api.o e1000_ich8lan.o \
1925                 e1000_mac.o e1000_manage.o e1000_nvm.o e1000_phy.o \
1926                 e1000_82575.o e1000_i210.o e1000_mbx.o e1000_vf.o

1928 #
1929 #       e1000g module
1930 #
1931 E1000G_OBJS +=  e1000g_debug.o e1000g_main.o e1000g_alloc.o \
1932                 e1000g_tx.o e1000g_rx.o e1000g_stat.o \
1933                 e1000g_osdep.o e1000g_workarounds.o



1936 #
1937 #       Intel 82575 1G NIC driver module
1938 #
1939 IGB_OBJS =      igb_buf.o igb_debug.o igb_gld.o igb_log.o igb_main.o \
1940                 igb_rx.o igb_stat.o igb_tx.o igb_osdep.o

1942 #
1943 #       Intel Pro/100 NIC driver module
1944 #
1945 IPRB_OBJS =     iprb.o

1947 #
1948 #       Intel 10GbE PCIE NIC driver module
1949 #
1950 IXGBE_OBJS =    ixgbe_82598.o ixgbe_82599.o ixgbe_api.o         \
1951                 ixgbe_common.o ixgbe_phy.o                      \
1952                 ixgbe_buf.o ixgbe_debug.o ixgbe_gld.o           \
1953                 ixgbe_log.o ixgbe_main.o                        \
1954                 ixgbe_osdep.o ixgbe_rx.o ixgbe_stat.o           \
1955                 ixgbe_tx.o  ixgbe_x540.o ixgbe_mbx.o

1957 #
1958 #       NIU 10G/1G driver module
1959 #
1960 NXGE_OBJS =     nxge_mac.o nxge_ipp.o nxge_rxdma.o               \
1961                 nxge_txdma.o nxge_txc.o nxge_main.o             \
1962                 nxge_hw.o nxge_fzc.o nxge_virtual.o             \
1963                 nxge_send.o nxge_classify.o nxge_fflp.o         \
1964                 nxge_fflp_hash.o nxge_ndd.o nxge_kstats.o       \
1965                 nxge_zcp.o nxge_fm.o nxge_espc.o nxge_hv.o      \
1966                 nxge_hio.o nxge_hio_guest.o nxge_intr.o

1968 NXGE_NPI_OBJS = \
1969                 npi.o npi_mac.o npi_ipp.o                       \
1970                 npi_txdma.o npi_rxdma.o npi_txc.o               \
1971                 npi_zcp.o npi_espc.o npi_fflp.o                 \
1972                 npi_vir.o
```

```
1974 NXGE_HCALL_OBJS =       \
1975                 nxge_hcall.o

1977 #
1978 # Virtio modules
1979 #

1981 # Virtio core
1982 VIRTIO_OBJS = virtio.o

1984 # Virtio block driver
1985 VIOBLK_OBJS = vioblk.o

1987 #
1988 #       kiconv modules
1989 #
1990 KICONV_EMEA_OBJS += kiconv_emea.o

1992 KICONV_JA_OBJS += kiconv_ja.o

1994 KICONV_KO_OBJS += kiconv_cck_common.o kiconv_ko.o

1996 KICONV_SC_OBJS += kiconv_cck_common.o kiconv_sc.o

1998 KICONV_TC_OBJS += kiconv_cck_common.o kiconv_tc.o

2000 #
2001 #       AAC module
2002 #
2003 AAC_OBJS = aac.o aac_ioctl.o

2005 #
2006 #       sdcard modules
2007 #
2008 SDA_OBJS =      sda_cmd.o sda_host.o sda_init.o sda_mem.o sda_mod.o sda_slot.o
2009 SDHOST_OBJS =   sdhost.o

2011 #
2012 #       hxge 10G driver module
2013 #
2014 HXGE_OBJS =     hxge_main.o hxge_vmac.o hxge_send.o              \
2015                 hxge_txdma.o hxge_rxdma.o hxge_virtual.o        \
2016                 hxge_fm.o hxge_fzc.o hxge_hw.o hxge_kstats.o    \
2017                 hxge_ndd.o hxge_pfc.o                           \
2018                 hpi.o hpi_vmac.o hpi_rxdma.o hpi_txdma.o        \
2019                 hpi_vir.o hpi_pfc.o

2021 #
2022 #       MEGARAID_SAS module
2023 #
2024 MEGA_SAS_OBJS = megaraid_sas.o

2026 #
2027 #       MR_SAS module
2028 #
2029 MR_SAS_OBJS = ld_pd_map.o mr_sas.o mr_sas_tbolt.o mr_sas_list.o

2031 #
2032 #       CPQARY3 module
2033 #
2034 CPQARY3_OBJS =  cpqary3.o cpqary3_noe.o cpqary3_talk2ctlr.o     \
2035                 cpqary3_isr.o cpqary3_transport.o cpqary3_mem.o \
2036                 cpqary3_scsi.o cpqary3_util.o cpqary3_ioctl.o   \
2037                 cpqary3_bd.o
```



```
2039 #
2040 #       ISCSI_INITIATOR module
2041 #
2042 ISCSI_INITIATOR_OBJS =  chap.o iscsi_io.o iscsi_thread.o        \
2043                         iscsi_ioctl.o iscsid.o iscsi.o          \
2044                         iscsi_login.o isns_client.o iscsiAuthClient.o   \
2045                         iscsi_lun.o iscsiAuthClientGlue.o       \
2046                         iscsi_net.o nvfile.o iscsi_cmd.o        \
2047                         iscsi_queue.o persistent.o iscsi_conn.o \
2048                         iscsi_sess.o radius_auth.o iscsi_crc.o  \
2049                         iscsi_stats.o radius_packet.o iscsi_doorclt.o   \
2050                         iscsi_targetparam.o utils.o kifconf.o

2052 #
2053 #       ntxn 10Gb/1Gb NIC driver module
2054 #
2055 NTXN_OBJS =     unm_nic_init.o unm_gem.o unm_nic_hw.o unm_ndd.o \
2056                 unm_nic_main.o unm_nic_isr.o unm_nic_ctx.o niu.o

2058 #
2059 #       Myricom 10Gb NIC driver module
2060 #
2061 MYRI10GE_OBJS = myri10ge.o myri10ge_lro.o

2063 #       nulldriver module
2064 #
2065 NULLDRIVER_OBJS =       nulldriver.o

2067 TPM_OBJS =      tpm.o tpm_hcall.o

2069 #
2070 #       BNXE objects
2071 #
2072 BNXE_OBJS +=    bnxe_cfg.o              \
2073                 bnxe_fcoe.o             \
2074                 bnxe_debug.o            \
2075                 bnxe_gld.o              \
2076                 bnxe_hw.o               \
2077                 bnxe_intr.o             \
2078                 bnxe_kstat.o            \
2079                 bnxe_lock.o             \
2080                 bnxe_main.o             \
2081                 bnxe_mm.o               \
2082                 bnxe_mm_l4.o            \
2083                 bnxe_mm_l5.o            \
2084                 bnxe_rr.o               \
2085                 bnxe_rx.o               \
2086                 bnxe_timer.o            \
2087                 bnxe_tx.o               \
2088                 bnxe_workq.o            \
2089                 bnxe_clc.o              \
2090                 ecore_sp_verbs.o        \
2091                 bnxe_context.o          \
2092                 57710_init_values.o     \
2093                 57711_init_values.o     \
2094                 57712_init_values.o     \
2095                 bnxe_fw_funcs.o         \
2096                 bnxe_hw_debug.o         \
2097                 lm_l4fp.o               \
2098                 lm_l4rx.o               \
2099                 lm_l4sp.o               \
2100                 lm_l4tx.o               \
2101                 lm_l5.o                 \
2102                 lm_l5sp.o               \
2103                 lm_dcbx.o               \
2104                 lm_devinfo.o            \
```

```
2105                    lm_dmae.o               \
2106                    lm_er.o                 \
2107                    lm_hw_access.o          \
2108                    lm_hw_attn.o            \
2109                    lm_hw_init_reset.o      \
2110                    lm_main.o               \
2111                    lm_mcp.o                \
2112                    lm_niv.o                \
2113                    lm_nvram.o              \
2114                    lm_phy.o                \
2115                    lm_power.o              \
2116                    lm_recv.o               \
2117                    lm_resc.o               \
2118                    lm_sb.o                 \
2119                    lm_send.o               \
2120                    lm_sp.o                 \
2121                    lm_dcbx_mp.o            \
2122                    lm_sp_req_mgr.o         \
2123                    lm_stats.o              \
2124                    lm_util.o
```

```
**********************************************************
  131260 Wed May 27 19:49:24 2015
new/usr/src/uts/common/c2/audit_event.c
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
```

```
   1 /*
   2  * CDDL HEADER START
   3  *
   4  * The contents of this file are subject to the terms of the
   5  * Common Development and Distribution License (the "License").
   6  * You may not use this file except in compliance with the License.
   7  *
   8  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
   9  * or http://www.opensolaris.org/os/licensing.
  10  * See the License for the specific language governing permissions
  11  * and limitations under the License.
  12  *
  13  * When distributing Covered Code, include this CDDL HEADER in each
  14  * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  15  * If applicable, add the following below this CDDL HEADER, with the
  16  * fields enclosed by brackets "[]" replaced with your own identifying
  17  * information: Portions Copyright [yyyy] [name of copyright owner]
  18  *
  19  * CDDL HEADER END
  20  */

  22 /*
  23  * Copyright (c) 1992, 2010, Oracle and/or its affiliates. All rights reserved.
  24  * Copyright (c) 2011 Bayard G. Bell. All rights reserved.
  25  */

  27 /*
  28  * This file contains the audit event table used to control the production
  29  * of audit records for each system call.
  30  */

  32 #include <sys/policy.h>
  33 #include <sys/cred.h>
  34 #include <sys/types.h>
  35 #include <sys/systm.h>
  36 #include <sys/systeminfo.h>     /* for sysinfo auditing */
  37 #include <sys/utsname.h>        /* for sysinfo auditing */
  38 #include <sys/proc.h>
  39 #include <sys/vnode.h>
  40 #include <sys/mman.h>           /* for mmap(2) auditing etc. */
  41 #include <sys/fcntl.h>
  42 #include <sys/modctl.h>         /* for modctl auditing */
  43 #include <sys/vnode.h>
  44 #include <sys/user.h>
  45 #include <sys/types.h>
  46 #include <sys/processor.h>
  47 #include <sys/procset.h>
  48 #include <sys/acl.h>
  49 #include <sys/ipc.h>
  50 #include <sys/door.h>
  51 #include <sys/sem.h>
  52 #include <sys/msg.h>
  53 #include <sys/shm.h>
  54 #include <sys/kmem.h>
  55 #include <sys/file.h>           /* for accept */
  56 #include <sys/utssys.h>         /* for fuser */
  57 #include <sys/tsol/label.h>
  58 #include <sys/tsol/tndb.h>
```



```
  59 #include <sys/tsol/tsyscall.h>
  60 #include <c2/audit.h>
  61 #include <c2/audit_kernel.h>
  62 #include <c2/audit_kevents.h>
  63 #include <c2/audit_record.h>
  64 #include <sys/procset.h>
  65 #include <nfs/mount.h>
  66 #include <sys/param.h>
  67 #include <sys/debug.h>
  68 #include <sys/sysmacros.h>
  69 #include <sys/stream.h>
  70 #include <sys/strsubr.h>
  71 #include <sys/stropts.h>
  72 #include <sys/tihdr.h>
  73 #include <sys/socket.h>
  74 #include <sys/socketvar.h>
  75 #include <sys/vfs_opreg.h>
  76 #include <fs/sockfs/sockcommon.h>
  77 #include <netinet/in.h>
  78 #include <sys/ddi.h>
  79 #include <sys/port_impl.h>

  81 static au_event_t       aui_fchownat(au_event_t);
  82 static au_event_t       aui_fchmodat(au_event_t);
  83 static au_event_t       aui_open(au_event_t);
  84 static au_event_t       aui_openat(au_event_t);
  85 static au_event_t       aui_unlinkat(au_event_t);
  86 static au_event_t       aui_fstatat(au_event_t);
  87 static au_event_t       aui_msgsys(au_event_t);
  88 static au_event_t       aui_shmsys(au_event_t);
  89 static au_event_t       aui_semsys(au_event_t);
  90 static au_event_t       aui_utssys(au_event_t);
  91 static au_event_t       aui_fcntl(au_event_t);
  92 static au_event_t       aui_execve(au_event_t);
  93 static au_event_t       aui_memcntl(au_event_t);
  94 static au_event_t       aui_sysinfo(au_event_t);
  95 static au_event_t       aui_portfs(au_event_t);
  96 static au_event_t       aui_auditsys(au_event_t);
  97 static au_event_t       aui_modctl(au_event_t);
  98 static au_event_t       aui_acl(au_event_t);
  99 static au_event_t       aui_doorfs(au_event_t);
 100 static au_event_t       aui_privsys(au_event_t);
 101 static au_event_t       aui_forksys(au_event_t);
 102 static au_event_t       aui_labelsys(au_event_t);
 103 static au_event_t       aui_setpgrp(au_event_t);

 105 static void     aus_exit(struct t_audit_data *);
 106 static void     aus_open(struct t_audit_data *);
 107 static void     aus_openat(struct t_audit_data *);
 108 static void     aus_acl(struct t_audit_data *);
 109 static void     aus_acct(struct t_audit_data *);
 110 static void     aus_chown(struct t_audit_data *);
 111 static void     aus_fchown(struct t_audit_data *);
 112 static void     aus_lchown(struct t_audit_data *);
 113 static void     aus_fchownat(struct t_audit_data *);
 114 static void     aus_chmod(struct t_audit_data *);
 115 static void     aus_facl(struct t_audit_data *);
 116 static void     aus_fchmod(struct t_audit_data *);
 117 static void     aus_fchmodat(struct t_audit_data *);
 118 static void     aus_fcntl(struct t_audit_data *);
 119 static void     aus_mkdir(struct t_audit_data *);
 120 static void     aus_mkdirat(struct t_audit_data *);
 121 static void     aus_mknod(struct t_audit_data *);
 122 static void     aus_mknodat(struct t_audit_data *);
 123 static void     aus_mount(struct t_audit_data *);
 124 static void     aus_umount2(struct t_audit_data *);
```

```
125 static void     aus_msgsys(struct t_audit_data *);
126 static void     aus_semsys(struct t_audit_data *);
127 static void     aus_close(struct t_audit_data *);
128 static void     aus_fstatfs(struct t_audit_data *);
129 static void     aus_setgid(struct t_audit_data *);
130 static void     aus_setpgrp(struct t_audit_data *);
131 static void     aus_setuid(struct t_audit_data *);
132 static void     aus_shmsys(struct t_audit_data *);
133 static void     aus_doorfs(struct t_audit_data *);
134 static void     aus_ioctl(struct t_audit_data *);
135 static void     aus_memcntl(struct t_audit_data *);
136 static void     aus_mmap(struct t_audit_data *);
137 static void     aus_munmap(struct t_audit_data *);
138 static void     aus_priocntlsys(struct t_audit_data *);
139 static void     aus_psecflags(struct t_audit_data *);
140 #endif /* ! codereview */
141 static void     aus_setegid(struct t_audit_data *);
142 static void     aus_setgroups(struct t_audit_data *);
143 static void     aus_seteuid(struct t_audit_data *);
144 static void     aus_putmsg(struct t_audit_data *);
145 static void     aus_putpmsg(struct t_audit_data *);
146 static void     aus_getmsg(struct t_audit_data *);
147 static void     aus_getpmsg(struct t_audit_data *);
148 static void     aus_auditsys(struct t_audit_data *);
149 static void     aus_sysinfo(struct t_audit_data *);
150 static void     aus_modctl(struct t_audit_data *);
151 static void     aus_kill(struct t_audit_data *);
152 static void     aus_setregid(struct t_audit_data *);
153 static void     aus_setreuid(struct t_audit_data *);
154 static void     aus_labelsys(struct t_audit_data *);

156 static void     auf_mknod(struct t_audit_data *, int, rval_t *);
157 static void     auf_mknodat(struct t_audit_data *, int, rval_t *);
158 static void     auf_msgsys(struct t_audit_data *, int, rval_t *);
159 static void     auf_semsys(struct t_audit_data *, int, rval_t *);
160 static void     auf_shmsys(struct t_audit_data *, int, rval_t *);
161 static void     auf_read(struct t_audit_data *, int, rval_t *);
162 static void     auf_write(struct t_audit_data *, int, rval_t *);

164 static void     aus_sigqueue(struct t_audit_data *);
165 static void     aus_p_online(struct t_audit_data *);
166 static void     aus_processor_bind(struct t_audit_data *);
167 static void     aus_inst_sync(struct t_audit_data *);
168 static void     aus_brandsys(struct t_audit_data *);

170 static void     auf_accept(struct t_audit_data *, int, rval_t *);

172 static void     auf_bind(struct t_audit_data *, int, rval_t *);
173 static void     auf_connect(struct t_audit_data *, int, rval_t *);
174 static void     aus_shutdown(struct t_audit_data *);
175 static void     auf_setsockopt(struct t_audit_data *, int, rval_t *);
176 static void     aus_sockconfig(struct t_audit_data *);
177 static void     auf_recv(struct t_audit_data *, int, rval_t *);
178 static void     auf_recvmsg(struct t_audit_data *, int, rval_t *);
179 static void     auf_send(struct t_audit_data *, int, rval_t *);
180 static void     auf_sendmsg(struct t_audit_data *, int, rval_t *);
181 static void     auf_recvfrom(struct t_audit_data *, int, rval_t *);
182 static void     auf_sendto(struct t_audit_data *, int, rval_t *);
183 static void     aus_socket(struct t_audit_data *);
184 /*
185  * This table contains mapping information for converting system call numbers
186  * to audit event IDs. In several cases it is necessary to map a single system
187  * call to several events.
188  */

190 #define aui_null        NULL    /* NULL initialize function */
```



```
191 #define aus_null        NULL    /* NULL start function */
192 #define auf_null        NULL    /* NULL finish function */

194 struct audit_s2e audit_s2e[] =
195 {
196 /*
197  * ----------   ----------      ----------      ----------
198  * INITIAL      AUDIT           START           SYSTEM
199  * PROCESSING   EVENT           PROCESSING      CALL
200  * ----------   ----------      ----------      ----------
201  *              FINISH          EVENT
202  *              PROCESSING      CONTROL
203  * ------------------------------------------------------
204  */
205 aui_null,       AUE_NULL,       aus_null,       /* 0 unused (indirect) */
206                 auf_null,       0,
207 aui_null,       AUE_EXIT,       aus_exit,       /* 1 exit */
208                 auf_null,       S2E_NPT,
209 aui_null,       AUE_PSECFLAGS,  aus_psecflags,  /* 2 psecflags */
139 aui_null,       AUE_NULL,       aus_null,       /* 2 (loadable) was forkall */
210                 auf_null,       0,
211 aui_null,       AUE_READ,       aus_null,       /* 3 read */
212                 auf_read,       S2E_PUB,
213 aui_null,       AUE_WRITE,      aus_null,       /* 4 write */
214                 auf_write,      0,
215 aui_open,       AUE_OPEN,       aus_open,       /* 5 open */
216                 auf_null,       S2E_SP,
217 aui_null,       AUE_CLOSE,      aus_close,      /* 6 close */
218                 auf_null,       0,
219 aui_null,       AUE_LINK,       aus_null,       /* 7 linkat */
220                 auf_null,       0,
221 aui_null,       AUE_NULL,       aus_null,       /* 8 (loadable) was creat */
222                 auf_null,       0,
223 aui_null,       AUE_LINK,       aus_null,       /* 9 link */
224                 auf_null,       0,
225 aui_null,       AUE_UNLINK,     aus_null,       /* 10 unlink */
226                 auf_null,       0,
227 aui_null,       AUE_SYMLINK,    aus_null,       /* 11 symlinkat */
228                 auf_null,       0,
229 aui_null,       AUE_CHDIR,      aus_null,       /* 12 chdir */
230                 auf_null,       S2E_SP,
231 aui_null,       AUE_NULL,       aus_null,       /* 13 time */
232                 auf_null,       0,
233 aui_null,       AUE_MKNOD,      aus_mknod,      /* 14 mknod */
234                 auf_mknod,      S2E_MLD,
235 aui_null,       AUE_CHMOD,      aus_chmod,      /* 15 chmod */
236                 auf_null,       0,
237 aui_null,       AUE_CHOWN,      aus_chown,      /* 16 chown */
238                 auf_null,       0,
239 aui_null,       AUE_NULL,       aus_null,       /* 17 brk */
240                 auf_null,       0,
241 aui_null,       AUE_STAT,       aus_null,       /* 18 stat */
242                 auf_null,       S2E_PUB,
243 aui_null,       AUE_NULL,       aus_null,       /* 19 lseek */
244                 auf_null,       0,
245 aui_null,       AUE_NULL,       aus_null,       /* 20 getpid */
246                 auf_null,       0,
247 aui_null,       AUE_MOUNT,      aus_mount,      /* 21 mount */
248                 auf_null,       S2E_MLD,
249 aui_null,       AUE_READLINK,   aus_null,       /* 22 readlinkat */
250                 auf_null,       S2E_PUB,
251 aui_null,       AUE_SETUID,     aus_setuid,     /* 23 setuid */
252                 auf_null,       0,
253 aui_null,       AUE_NULL,       aus_null,       /* 24 getuid */
254                 auf_null,       0,
255 aui_null,       AUE_STIME,      aus_null,       /* 25 stime */
```

```
256                 auf_null,       0,
257 aui_null,       AUE_NULL,       aus_null,       /* 26 pcsample */
258                 auf_null,       0,
259 aui_null,       AUE_NULL,       aus_null,       /* 27 alarm */
260                 auf_null,       0,
261 aui_null,       AUE_NULL,       aus_null,       /* 28 fstat */
262                 auf_null,       0,
263 aui_null,       AUE_NULL,       aus_null,       /* 29 pause */
264                 auf_null,       0,
265 aui_null,       AUE_NULL,       aus_null,       /* 30 (loadable) was utime */
266                 auf_null,       0,
267 aui_null,       AUE_NULL,       aus_null,       /* 31 stty (TIOCSETP-audit?) */
268                 auf_null,       0,
269 aui_null,       AUE_NULL,       aus_null,       /* 32 gtty */
270                 auf_null,       0,
271 aui_null,       AUE_ACCESS,     aus_null,       /* 33 access */
272                 auf_null,       S2E_PUB,
273 aui_null,       AUE_NICE,       aus_null,       /* 34 nice */
274                 auf_null,       0,
275 aui_null,       AUE_STATFS,     aus_null,       /* 35 statfs */
276                 auf_null,       S2E_PUB,
277 aui_null,       AUE_NULL,       aus_null,       /* 36 sync */
278                 auf_null,       0,
279 aui_null,       AUE_KILL,       aus_kill,       /* 37 kill */
280                 auf_null,       0,
281 aui_null,       AUE_FSTATFS,    aus_fstatfs,    /* 38 fstatfs */
282                 auf_null,       S2E_PUB,
283 aui_setpgrp,    AUE_SETPGRP,    aus_setpgrp,    /* 39 setpgrp */
284                 auf_null,       0,
285 aui_null,       AUE_NULL,       aus_null,       /* 40 uucopystr */
286                 auf_null,       0,
287 aui_null,       AUE_NULL,       aus_null,       /* 41 (loadable) was dup */
288                 auf_null,       0,
289 aui_null,       AUE_PIPE,       aus_null,       /* 42 (loadable) pipe */
290                 auf_null,       0,
291 aui_null,       AUE_NULL,       aus_null,       /* 43 times */
292                 auf_null,       0,
293 aui_null,       AUE_NULL,       aus_null,       /* 44 profil */
294                 auf_null,       0,
295 aui_null,       AUE_ACCESS,     aus_null,       /* 45 faccessat */
296                 auf_null,       S2E_PUB,
297 aui_null,       AUE_SETGID,     aus_setgid,     /* 46 setgid */
298                 auf_null,       0,
299 aui_null,       AUE_NULL,       aus_null,       /* 47 getgid */
300                 auf_null,       0,
301 aui_null,       AUE_MKNOD,      aus_mknodat,    /* 48 mknodat */
302                 auf_mknodat,    S2E_MLD,
303 aui_msgsys,     AUE_MSGSYS,     aus_msgsys,     /* 49 (loadable) msgsys */
304                 auf_msgsys,     0,
305 #if defined(__x86)
306 aui_null,       AUE_NULL,       aus_null,       /* 50 sysi86 */
307                 auf_null,       0,
308 #else
309 aui_null,       AUE_NULL,       aus_null,       /* 50 (loadable) was sys3b */
310                 auf_null,       0,
311 #endif /* __x86 */
312 aui_null,       AUE_ACCT,       aus_acct,       /* 51 (loadable) sysacct */
313                 auf_null,       0,
314 aui_shmsys,     AUE_SHMSYS,     aus_shmsys,     /* 52 (loadable) shmsys */
315                 auf_shmsys,     0,
316 aui_semsys,     AUE_SEMSYS,     aus_semsys,     /* 53 (loadable) semsys */
317                 auf_semsys,     0,
318 aui_null,       AUE_IOCTL,      aus_ioctl,      /* 54 ioctl */
319                 auf_null,       0,
320 aui_null,       AUE_NULL,       aus_null,       /* 55 uadmin */
321                 auf_null,       0,
```



```
322 aui_fchownat,   AUE_NULL,       aus_fchownat,   /* 56 fchownat */
323                 auf_null,       0,
324 aui_utssys,     AUE_FUSERS,     aus_null,       /* 57 utssys */
325                 auf_null,       0,
326 aui_null,       AUE_NULL,       aus_null,       /* 58 fsync */
327                 auf_null,       0,
328 aui_execve,     AUE_EXECVE,     aus_null,       /* 59 exece */
329                 auf_null,       S2E_MLD,
330 aui_null,       AUE_NULL,       aus_null,       /* 60 umask */
331                 auf_null,       0,
332 aui_null,       AUE_CHROOT,     aus_null,       /* 61 chroot */
333                 auf_null,       S2E_SP,
334 aui_fcntl,      AUE_FCNTL,      aus_fcntl,      /* 62 fcntl */
335                 auf_null,       0,
336 aui_null,       AUE_NULL,       aus_null,       /* 63 ulimit */
337                 auf_null,       0,
338 aui_null,       AUE_RENAME,     aus_null,       /* 64 renameat */
339                 auf_null,       0,
340 aui_unlinkat,   AUE_NULL,       aus_null,       /* 65 unlinkat */
341                 auf_null,       0,
342 aui_fstatat,    AUE_NULL,       aus_null,       /* 66 fstatat */
343                 auf_null,       S2E_PUB,
344 aui_fstatat,    AUE_NULL,       aus_null,       /* 67 fstatat64 */
345                 auf_null,       S2E_PUB,
346 aui_openat,     AUE_OPEN,       aus_openat,     /* 68 openat */
347                 auf_null,       S2E_SP,
348 aui_openat,     AUE_OPEN,       aus_openat,     /* 69 openat64 */
349                 auf_null,       S2E_SP,
350 aui_null,       AUE_NULL,       aus_null,       /* 70 tasksys */
351                 auf_null,       0,
352 aui_null,       AUE_NULL,       aus_null,       /* 71 (loadable) acctctl */
353                 auf_null,       0,
354 aui_null,       AUE_NULL,       aus_null,       /* 72 (loadable) exacct */
355                 auf_null,       0,
356 aui_null,       AUE_NULL,       aus_null,       /* 73 getpagesizes */
357                 auf_null,       0,
358 aui_null,       AUE_NULL,       aus_null,       /* 74 rctlsys */
359                 auf_null,       0,
360 aui_null,       AUE_NULL,       aus_null,       /* 75 sidsys */
361                 auf_null,       0,
362 aui_null,       AUE_NULL,       aus_null,       /* 76 (loadable) was fsat */
363                 auf_null,       0,
364 aui_null,       AUE_NULL,       aus_null,       /* 77 syslwp_park */
365                 auf_null,       0,
366 aui_null,       AUE_NULL,       aus_null,       /* 78 sendfilev */
367                 auf_null,       0,
368 aui_null,       AUE_RMDIR,      aus_null,       /* 79 rmdir */
369                 auf_null,       0,
370 aui_null,       AUE_MKDIR,      aus_mkdir,      /* 80 mkdir */
371                 auf_null,       0,
372 aui_null,       AUE_NULL,       aus_null,       /* 81 getdents */
373                 auf_null,       0,
374 aui_privsys,    AUE_NULL,       aus_null,       /* 82 privsys */
375                 auf_null,       0,
376 aui_null,       AUE_NULL,       aus_null,       /* 83 ucredsys */
377                 auf_null,       0,
378 aui_null,       AUE_NULL,       aus_null,       /* 84 sysfs */
379                 auf_null,       0,
380 aui_null,       AUE_GETMSG,     aus_getmsg,     /* 85 getmsg */
381                 auf_null,       0,
382 aui_null,       AUE_PUTMSG,     aus_putmsg,     /* 86 putmsg */
383                 auf_null,       0,
384 aui_null,       AUE_NULL,       aus_null,       /* 87 (loadable) was poll */
385                 auf_null,       0,
386 aui_null,       AUE_LSTAT,      aus_null,       /* 88 lstat */
387                 auf_null,       S2E_PUB,
```

```
388 aui_null,       AUE_SYMLINK,    aus_null,       /* 89 symlink */
389                 auf_null,       0,
390 aui_null,       AUE_READLINK,   aus_null,       /* 90 readlink */
391                 auf_null,       S2E_PUB,
392 aui_null,       AUE_SETGROUPS,  aus_setgroups,  /* 91 setgroups */
393                 auf_null,       0,
394 aui_null,       AUE_NULL,       aus_null,       /* 92 getgroups */
395                 auf_null,       0,
396 aui_null,       AUE_FCHMOD,     aus_fchmod,     /* 93 fchmod */
397                 auf_null,       0,
398 aui_null,       AUE_FCHOWN,     aus_fchown,     /* 94 fchown */
399                 auf_null,       0,
400 aui_null,       AUE_NULL,       aus_null,       /* 95 sigprocmask */
401                 auf_null,       0,
402 aui_null,       AUE_NULL,       aus_null,       /* 96 sigsuspend */
403                 auf_null,       0,
404 aui_null,       AUE_NULL,       aus_null,       /* 97 sigaltstack */
405                 auf_null,       0,
406 aui_null,       AUE_NULL,       aus_null,       /* 98 sigaction */
407                 auf_null,       0,
408 aui_null,       AUE_NULL,       aus_null,       /* 99 sigpending */
409                 auf_null,       0,
410 aui_null,       AUE_NULL,       aus_null,       /* 100 setcontext */
411                 auf_null,       0,
412 aui_fchmodat,   AUE_NULL,       aus_fchmodat,   /* 101 fchmodat */
413                 auf_null,       0,
414 aui_null,       AUE_MKDIR,      aus_mkdirat,    /* 102 mkdirat */
415                 auf_null,       0,
416 aui_null,       AUE_STATVFS,    aus_null,       /* 103 statvfs */
417                 auf_null,       S2E_PUB,
418 aui_null,       AUE_NULL,       aus_null,       /* 104 fstatvfs */
419                 auf_null,       0,
420 aui_null,       AUE_NULL,       aus_null,       /* 105 getloadavg */
421                 auf_null,       0,
422 aui_null,       AUE_NULL,       aus_null,       /* 106 nfssys */
423                 auf_null,       0,
424 aui_null,       AUE_NULL,       aus_null,       /* 107 waitsys */
425                 auf_null,       0,
426 aui_null,       AUE_NULL,       aus_null,       /* 108 sigsendsys */
427                 auf_null,       0,
428 #if defined(__x86)
429 aui_null,       AUE_NULL,       aus_null,       /* 109 hrtsys */
430                 auf_null,       0,
431 #else
432 aui_null,       AUE_NULL,       aus_null,       /* 109 (loadable) */
433                 auf_null,       0,
434 #endif /* __x86 */
435 aui_null,       AUE_UTIMES,     aus_null,       /* 110 utimesys */
436                 auf_null,       0,
437 aui_null,       AUE_NULL,       aus_null,       /* 111 sigresend */
438                 auf_null,       0,
439 aui_null,       AUE_PRIOCNTLSYS, aus_priocntlsys, /* 112 priocntlsys */
440                 auf_null,       0,
441 aui_null,       AUE_PATHCONF,   aus_null,       /* 113 pathconf */
442                 auf_null,       S2E_PUB,
443 aui_null,       AUE_NULL,       aus_null,       /* 114 mincore */
444                 auf_null,       0,
445 aui_null,       AUE_MMAP,       aus_mmap,       /* 115 mmap */
446                 auf_null,       0,
447 aui_null,       AUE_NULL,       aus_null,       /* 116 mprotect */
448                 auf_null,       0,
449 aui_null,       AUE_MUNMAP,     aus_munmap,     /* 117 munmap */
450                 auf_null,       0,
451 aui_null,       AUE_NULL,       aus_null,       /* 118 fpathconf */
452                 auf_null,       0,
453 aui_null,       AUE_VFORK,      aus_null,       /* 119 vfork */
```



```
454                 auf_null,       0,
455 aui_null,       AUE_FCHDIR,     aus_null,       /* 120 fchdir */
456                 auf_null,       0,
457 aui_null,       AUE_READ,       aus_null,       /* 121 readv */
458                 auf_read,       S2E_PUB,
459 aui_null,       AUE_WRITE,      aus_null,       /* 122 writev */
460                 auf_write,      0,
461 aui_null,       AUE_NULL,       aus_null,       /* 123 (loadable) was xstat */
462                 auf_null,       0,
463 aui_null,       AUE_NULL,       aus_null,       /* 124 (loadable) was lxstat */
464                 auf_null,       0,
465 aui_null,       AUE_NULL,       aus_null,       /* 125 (loadable) was fxstat */
466                 auf_null,       0,
467 aui_null,       AUE_NULL,       aus_null,       /* 126 (loadable) was xmknod */
468                 auf_null,       0,
469 aui_null,       AUE_NULL,       aus_null,       /* 127 mmapobj */
470                 auf_null,       0,
471 aui_null,       AUE_SETRLIMIT,  aus_null,       /* 128 setrlimit */
472                 auf_null,       0,
473 aui_null,       AUE_NULL,       aus_null,       /* 129 getrlimit */
474                 auf_null,       0,
475 aui_null,       AUE_LCHOWN,     aus_lchown,     /* 130 lchown */
476                 auf_null,       0,
477 aui_memcntl,    AUE_MEMCNTL,    aus_memcntl,    /* 131 memcntl */
478                 auf_null,       0,
479 aui_null,       AUE_GETPMSG,    aus_getpmsg,    /* 132 getpmsg */
480                 auf_null,       0,
481 aui_null,       AUE_PUTPMSG,    aus_putpmsg,    /* 133 putpmsg */
482                 auf_null,       0,
483 aui_null,       AUE_RENAME,     aus_null,       /* 134 rename */
484                 auf_null,       0,
485 aui_null,       AUE_NULL,       aus_null,       /* 135 uname */
486                 auf_null,       0,
487 aui_null,       AUE_SETEGID,    aus_setegid,    /* 136 setegid */
488                 auf_null,       0,
489 aui_null,       AUE_NULL,       aus_null,       /* 137 sysconfig */
490                 auf_null,       0,
491 aui_null,       AUE_ADJTIME,    aus_null,       /* 138 adjtime */
492                 auf_null,       0,
493 aui_sysinfo,    AUE_SYSINFO,    aus_sysinfo,    /* 139 systeminfo */
494                 auf_null,       0,
495 aui_null,       AUE_NULL,       aus_null,       /* 140 (loadable) sharefs */
496                 auf_null,       0,
497 aui_null,       AUE_SETEUID,    aus_seteuid,    /* 141 seteuid */
498                 auf_null,       0,
499 aui_forksys,    AUE_NULL,       aus_null,       /* 142 forksys */
500                 auf_null,       0,
501 aui_null,       AUE_NULL,       aus_null,       /* 143 (loadable) was fork1 */
502                 auf_null,       0,
503 aui_null,       AUE_NULL,       aus_null,       /* 144 sigwait */
504                 auf_null,       0,
505 aui_null,       AUE_NULL,       aus_null,       /* 145 lwp_info */
506                 auf_null,       0,
507 aui_null,       AUE_NULL,       aus_null,       /* 146 yield */
508                 auf_null,       0,
509 aui_null,       AUE_NULL,       aus_null,       /* 147 (loadable) */
510                                                 /*      was lwp_sema_wait */
511                 auf_null,       0,
512 aui_null,       AUE_NULL,       aus_null,       /* 148 lwp_sema_post */
513                 auf_null,       0,
514 aui_null,       AUE_NULL,       aus_null,       /* 149 lwp_sema_trywait */
515                 auf_null,       0,
516 aui_null,       AUE_NULL,       aus_null,       /* 150 lwp_detach */
517                 auf_null,       0,
518 aui_null,       AUE_NULL,       aus_null,       /* 151 corectl */
519                 auf_null,       0,
```

```
aui_modctl,     AUE_MODCTL,     aus_modctl,     /* 152 modctl */
                auf_null,       0,
aui_null,       AUE_FCHROOT,    aus_null,       /* 153 fchroot */
                auf_null,       0,
aui_null,       AUE_NULL,       aus_null,       /* 154 (loadable) was utimes */
                auf_null,       0,
aui_null,       AUE_NULL,       aus_null,       /* 155 vhangup */
                auf_null,       0,
aui_null,       AUE_NULL,       aus_null,       /* 156 gettimeofday */
                auf_null,       0,
aui_null,       AUE_NULL,       aus_null,       /* 157 getitimer */
                auf_null,       0,
aui_null,       AUE_NULL,       aus_null,       /* 158 setitimer */
                auf_null,       0,
aui_null,       AUE_NULL,       aus_null,       /* 159 lwp_create */
                auf_null,       0,
aui_null,       AUE_NULL,       aus_null,       /* 160 lwp_exit */
                auf_null,       0,
aui_null,       AUE_NULL,       aus_null,       /* 161 lwp_suspend */
                auf_null,       0,
aui_null,       AUE_NULL,       aus_null,       /* 162 lwp_continue */
                auf_null,       0,
aui_null,       AUE_NULL,       aus_null,       /* 163 lwp_kill */
                auf_null,       0,
aui_null,       AUE_NULL,       aus_null,       /* 164 lwp_self */
                auf_null,       0,
aui_null,       AUE_NULL,       aus_null,       /* 165 lwp_sigmask */
                auf_null,       0,
aui_null,       AUE_NULL,       aus_null,       /* 166 lwp_private */
                auf_null,       0,
aui_null,       AUE_NULL,       aus_null,       /* 167 lwp_wait */
                auf_null,       0,
aui_null,       AUE_NULL,       aus_null,       /* 168 lwp_mutex_wakeup  */
                auf_null,       0,
aui_null,       AUE_NULL,       aus_null,       /* 169 (loadable) */
                                                /*      was lwp_mutex_lock */
                auf_null,       0,
aui_null,       AUE_NULL,       aus_null,       /* 170 lwp_cond_wait */
                auf_null,       0,
aui_null,       AUE_NULL,       aus_null,       /* 171 lwp_cond_signal */
                auf_null,       0,
aui_null,       AUE_NULL,       aus_null,       /* 172 lwp_cond_broadcast */
                auf_null,       0,
aui_null,       AUE_READ,       aus_null,       /* 173 pread */
                auf_read,       S2E_PUB,
aui_null,       AUE_WRITE,      aus_null,       /* 174 pwrite */
                auf_write,      0,
aui_null,       AUE_NULL,       aus_null,       /* 175 llseek */
                auf_null,       0,
aui_null,       AUE_INST_SYNC,  aus_inst_sync,  /* 176 (loadable) inst_sync */
                auf_null,       0,
aui_null,       AUE_BRANDSYS,   aus_brandsys,   /* 177 brandsys */
                auf_null,       0,
aui_null,       AUE_NULL,       aus_null,       /* 178 (loadable) kaio */
                auf_null,       0,
aui_null,       AUE_NULL,       aus_null,       /* 179 (loadable) cpc */
                auf_null,       0,
aui_null,       AUE_NULL,       aus_null,       /* 180 lgrpsys */
                auf_null,       0,
aui_null,       AUE_NULL,       aus_null,       /* 181 rusagesys */
                auf_null,       0,
aui_portfs,     AUE_PORTFS,     aus_null,       /* 182 (loadable) portfs */
                auf_null,       S2E_MLD,
aui_null,       AUE_NULL,       aus_null,       /* 183 pollsys */
                auf_null,       0,
aui_labelsys,   AUE_NULL,       aus_labelsys,   /* 184 labelsys */
```



```
                auf_null,       0,
aui_acl,        AUE_ACLSET,     aus_acl,        /* 185 acl */
                auf_null,       0,
aui_auditsys,   AUE_AUDITSYS,   aus_auditsys,   /* 186 auditsys  */
                auf_null,       0,
aui_null,       AUE_PROCESSOR_BIND, aus_processor_bind, /* 187 processor_bind */
                auf_null,       0,
aui_null,       AUE_NULL,       aus_null,       /* 188 processor_info */
                auf_null,       0,
aui_null,       AUE_P_ONLINE,   aus_p_online,   /* 189 p_online */
                auf_null,       0,
aui_null,       AUE_NULL,       aus_sigqueue,   /* 190 sigqueue */
                auf_null,       0,
aui_null,       AUE_NULL,       aus_null,       /* 191 clock_gettime */
                auf_null,       0,
aui_null,       AUE_CLOCK_SETTIME,      aus_null,       /* 192 clock_settime */
                auf_null,       0,
aui_null,       AUE_NULL,       aus_null,       /* 193 clock_getres */
                auf_null,       0,
aui_null,       AUE_NULL,       aus_null,       /* 194 timer_create */
                auf_null,       0,
aui_null,       AUE_NULL,       aus_null,       /* 195 timer_delete */
                auf_null,       0,
aui_null,       AUE_NULL,       aus_null,       /* 196 timer_settime */
                auf_null,       0,
aui_null,       AUE_NULL,       aus_null,       /* 197 timer_gettime */
                auf_null,       0,
aui_null,       AUE_NULL,       aus_null,       /* 198 timer_getoverrun */
                auf_null,       0,
aui_null,       AUE_NULL,       aus_null,       /* 199 nanosleep */
                auf_null,       0,
aui_acl,        AUE_FACLSET,    aus_facl,       /* 200 facl */
                auf_null,       0,
aui_doorfs,     AUE_DOORFS,     aus_doorfs,     /* 201 (loadable) doorfs */
                auf_null,       0,
aui_null,       AUE_SETREUID,   aus_setreuid,   /* 202 setreuid */
                auf_null,       0,
aui_null,       AUE_SETREGID,   aus_setregid,   /* 203 setregid */
                auf_null,       0,
aui_null,       AUE_NULL,       aus_null,       /* 204 install_utrap */
                auf_null,       0,
aui_null,       AUE_NULL,       aus_null,       /* 205 signotify */
                auf_null,       0,
aui_null,       AUE_NULL,       aus_null,       /* 206 schedctl */
                auf_null,       0,
aui_null,       AUE_NULL,       aus_null,       /* 207 (loadable) pset */
                auf_null,       0,
aui_null,       AUE_NULL,       aus_null,       /* 208 sparc_utrap_install */
                auf_null,       0,
aui_null,       AUE_NULL,       aus_null,       /* 209 resolvepath */
                auf_null,       0,
aui_null,       AUE_NULL,       aus_null,       /* 210 lwp_mutex_timedlock */
                auf_null,       0,
aui_null,       AUE_NULL,       aus_null,       /* 211 lwp_sema_timedwait */
                auf_null,       0,
aui_null,       AUE_NULL,       aus_null,       /* 212 lwp_rwlock_sys */
                auf_null,       0,
aui_null,       AUE_NULL,       aus_null,       /* 213 getdents64 */
                auf_null,       0,
aui_null,       AUE_MMAP,       aus_mmap,       /* 214 mmap64 */
                auf_null,       0,
aui_null,       AUE_STAT,       aus_null,       /* 215 stat64 */
                auf_null,       S2E_PUB,
aui_null,       AUE_LSTAT,      aus_null,       /* 216 lstat64 */
                auf_null,       S2E_PUB,
aui_null,       AUE_NULL,       aus_null,       /* 217 fstat64 */
```

```
652                 auf_null,       0,
653 aui_null,       AUE_STATVFS,    aus_null,       /* 218 statvfs64 */
654                 auf_null,       S2E_PUB,
655 aui_null,       AUE_NULL,       aus_null,       /* 219 fstatvfs64 */
656                 auf_null,       0,
657 aui_null,       AUE_SETRLIMIT,  aus_null,       /* 220 setrlimit64 */
658                 auf_null,       0,
659 aui_null,       AUE_NULL,       aus_null,       /* 221 getrlimit64 */
660                 auf_null,       0,
661 aui_null,       AUE_READ,       aus_null,       /* 222 pread64  */
662                 auf_read,       S2E_PUB,
663 aui_null,       AUE_WRITE,      aus_null,       /* 223 pwrite64 */
664                 auf_write,      0,
665 aui_null,       AUE_NULL,       aus_null,       /* 224 (loadable) was creat64 */
666                 auf_null,       0,
667 aui_open,       AUE_OPEN,       aus_open,       /* 225 open64 */
668                 auf_null,       S2E_SP,
669 aui_null,       AUE_NULL,       aus_null,       /* 226 (loadable) rpcsys */
670                 auf_null,       0,
671 aui_null,       AUE_NULL,       aus_null,       /* 227 zone */
672                 auf_null,       0,
673 aui_null,       AUE_NULL,       aus_null,       /* 228 (loadable) autofssys */
674                 auf_null,       0,
675 aui_null,       AUE_NULL,       aus_null,       /* 229 getcwd */
676                 auf_null,       0,
677 aui_null,       AUE_SOCKET,     aus_socket,     /* 230 so_socket */
678                 auf_null,       0,
679 aui_null,       AUE_NULL,       aus_null,       /* 231 so_socketpair */
680                 auf_null,       0,
681 aui_null,       AUE_BIND,       aus_null,       /* 232 bind */
682                 auf_bind,       0,
683 aui_null,       AUE_NULL,       aus_null,       /* 233 listen */
684                 auf_null,       0,
685 aui_null,       AUE_ACCEPT,     aus_null,       /* 234 accept */
686                 auf_accept,     0,
687 aui_null,       AUE_CONNECT,    aus_null,       /* 235 connect */
688                 auf_connect,    0,
689 aui_null,       AUE_SHUTDOWN,   aus_shutdown,   /* 236 shutdown */
690                 auf_null,       0,
691 aui_null,       AUE_READ,       aus_null,       /* 237 recv */
692                 auf_recv,       0,
693 aui_null,       AUE_RECVFROM,   aus_null,       /* 238 recvfrom */
694                 auf_recvfrom,   0,
695 aui_null,       AUE_RECVMSG,    aus_null,       /* 239 recvmsg */
696                 auf_recvmsg,    0,
697 aui_null,       AUE_WRITE,      aus_null,       /* 240 send */
698                 auf_send,       0,
699 aui_null,       AUE_SENDMSG,    aus_null,       /* 241 sendmsg */
700                 auf_sendmsg,    0,
701 aui_null,       AUE_SENDTO,     aus_null,       /* 242 sendto */
702                 auf_sendto,     0,
703 aui_null,       AUE_NULL,       aus_null,       /* 243 getpeername */
704                 auf_null,       0,
705 aui_null,       AUE_NULL,       aus_null,       /* 244 getsockname */
706                 auf_null,       0,
707 aui_null,       AUE_NULL,       aus_null,       /* 245 getsockopt */
708                 auf_null,       0,
709 aui_null,       AUE_SETSOCKOPT, aus_null,       /* 246 setsockopt */
710                 auf_setsockopt, 0,
711 aui_null,       AUE_SOCKCONFIG, aus_sockconfig, /* 247 sockconfig */
712                 auf_null,       0,
713 aui_null,       AUE_NULL,       aus_null,       /* 248 ntp_gettime */
714                 auf_null,       0,
715 aui_null,       AUE_NTP_ADJTIME, aus_null,      /* 249 ntp_adjtime */
716                 auf_null,       0,
717 aui_null,       AUE_NULL,       aus_null,       /* 250 lwp_mutex_unlock */
```



```
718                 auf_null,       0,
719 aui_null,       AUE_NULL,       aus_null,       /* 251 lwp_mutex_trylock */
720                 auf_null,       0,
721 aui_null,       AUE_NULL,       aus_null,       /* 252 lwp_mutex_register */
722                 auf_null,       0,
723 aui_null,       AUE_NULL,       aus_null,       /* 253 cladm */
724                 auf_null,       0,
725 aui_null,       AUE_NULL,       aus_null,       /* 254 uucopy */
726                 auf_null,       0,
727 aui_null,       AUE_UMOUNT2,    aus_umount2,    /* 255 umount2 */
728                 auf_null,       0
729 };
```

*unchanged_portion_omitted_*

```
748 /*ARGSUSED*/
749 static void
750 aus_psecflags(struct t_audit_data *tad)
751 {
752         struct a {
753                 uintptr_t psp;  /* procset_t */
754                 uint_t cmd;     /* psecflags_cmd_t */
755                 uint_t arg;
756         } *uap = (struct a *)ttolwp(curthread)->lwp_ap;
757
758         au_uwrite(au_to_arg32(2, "cmd", (uint_t)uap->cmd));
759         au_uwrite(au_to_arg32(3, "arg", (uint_t)uap->arg));
760 }
761
762 #endif /* ! codereview */
763 /* acct start function */
764 /*ARGSUSED*/
765 static void
766 aus_acct(struct t_audit_data *tad)
767 {
768         klwp_t *clwp = ttolwp(curthread);
769         uintptr_t fname;
770
771         struct a {
772                 long    fname;          /* char * */
773         } *uap = (struct a *)clwp->lwp_ap;
774
775         fname = (uintptr_t)uap->fname;
776
777         if (fname == 0)
778                 au_uwrite(au_to_arg32(1, "accounting off", (uint32_t)0));
779 }
780
781 /* chown start function */
782 /*ARGSUSED*/
783 static void
784 aus_chown(struct t_audit_data *tad)
785 {
786         klwp_t *clwp = ttolwp(curthread);
787         uint32_t uid, gid;
788
789         struct a {
790                 long    fname;          /* char * */
791                 long    uid;
792                 long    gid;
793         } *uap = (struct a *)clwp->lwp_ap;
794
795         uid = (uint32_t)uap->uid;
796         gid = (uint32_t)uap->gid;
797
798         au_uwrite(au_to_arg32(2, "new file uid", uid));
799         au_uwrite(au_to_arg32(3, "new file gid", gid));
```

```
}

/* fchown start function */
/*ARGSUSED*/
static void
aus_fchown(struct t_audit_data *tad)
{
        klwp_t *clwp = ttolwp(curthread);
        uint32_t uid, gid, fd;
        struct file  *fp;
        struct vnode *vp;
        struct f_audit_data *fad;

        struct a {
                long fd;
                long uid;
                long gid;
        } *uap = (struct a *)clwp->lwp_ap;

        fd  = (uint32_t)uap->fd;
        uid = (uint32_t)uap->uid;
        gid = (uint32_t)uap->gid;

        au_uwrite(au_to_arg32(2, "new file uid", uid));
        au_uwrite(au_to_arg32(3, "new file gid", gid));

                /*
                 * convert file pointer to file descriptor
                 *   Note: fd ref count incremented here.
                 */
        if ((fp = getf(fd)) == NULL)
                return;

        /* get path from file struct here */
        fad = F2A(fp);
        if (fad->fad_aupath != NULL) {
                au_uwrite(au_to_path(fad->fad_aupath));
        } else {
                au_uwrite(au_to_arg32(1, "no path: fd", fd));
        }

        vp = fp->f_vnode;
        audit_attributes(vp);

        /* decrement file descriptor reference count */
        releasef(fd);
}

/*ARGSUSED*/
static void
aus_lchown(struct t_audit_data *tad)
{
        klwp_t *clwp = ttolwp(curthread);
        uint32_t uid, gid;


        struct a {
                long    fname;          /* char * */
                long    uid;
                long    gid;
        } *uap = (struct a *)clwp->lwp_ap;

        uid = (uint32_t)uap->uid;
        gid = (uint32_t)uap->gid;

        au_uwrite(au_to_arg32(2, "new file uid", uid));
```



```
        au_uwrite(au_to_arg32(3, "new file gid", gid));
}

static au_event_t
aui_fchownat(au_event_t e)
{
        klwp_t *clwp = ttolwp(curthread);

        struct a {
                long    fd;
                long    fname;          /* char * */
                long    uid;
                long    gid;
                long    flags;
        } *uap = (struct a *)clwp->lwp_ap;

        if (uap->fname == NULL)
                e = AUE_FCHOWN;
        else if (uap->flags & AT_SYMLINK_NOFOLLOW)
                e = AUE_LCHOWN;
        else
                e = AUE_CHOWN;

        return (e);
}

/*ARGSUSED*/
static void
aus_fchownat(struct t_audit_data *tad)
{
        klwp_t *clwp = ttolwp(curthread);
        uint32_t uid, gid;

        struct a {
                long    fd;
                long    fname;          /* char * */
                long    uid;
                long    gid;
                long    flags;
        } *uap = (struct a *)clwp->lwp_ap;

        uid = (uint32_t)uap->uid;
        gid = (uint32_t)uap->gid;

        au_uwrite(au_to_arg32(3, "new file uid", uid));
        au_uwrite(au_to_arg32(4, "new file gid", gid));
}

/*ARGSUSED*/
static void
aus_chmod(struct t_audit_data *tad)
{
        klwp_t *clwp = ttolwp(curthread);
        uint32_t fmode;

        struct a {
                long    fname;          /* char * */
                long    fmode;
        } *uap = (struct a *)clwp->lwp_ap;

        fmode = (uint32_t)uap->fmode;

        au_uwrite(au_to_arg32(2, "new file mode", fmode&07777));
}

/*ARGSUSED*/
```

```
static void
aus_fchmod(struct t_audit_data *tad)
{
        klwp_t *clwp = ttolwp(curthread);
        uint32_t fmode, fd;
        struct file  *fp;
        struct vnode *vp;
        struct f_audit_data *fad;

        struct a {
                long    fd;
                long    fmode;
        } *uap = (struct a *)clwp->lwp_ap;

        fd = (uint32_t)uap->fd;
        fmode = (uint32_t)uap->fmode;

        au_uwrite(au_to_arg32(2, "new file mode", fmode&07777));

        /*
         * convert file pointer to file descriptor
         *   Note: fd ref count incremented here.
         */
        if ((fp = getf(fd)) == NULL)
                return;

        /* get path from file struct here */
        fad = F2A(fp);
        if (fad->fad_aupath != NULL) {
                au_uwrite(au_to_path(fad->fad_aupath));
        } else {
                au_uwrite(au_to_arg32(1, "no path: fd", fd));
        }

        vp = fp->f_vnode;
        audit_attributes(vp);

        /* decrement file descriptor reference count */
        releasef(fd);
}

static au_event_t
aui_fchmodat(au_event_t e)
{
        klwp_t *clwp = ttolwp(curthread);

        struct a {
                long    fd;
                long    fname;          /* char * */
                long    fmode;
                long    flag;
        } *uap = (struct a *)clwp->lwp_ap;

        if (uap->fname == NULL)
                e = AUE_FCHMOD;
        else
                e = AUE_CHMOD;

        return (e);
}

/*ARGSUSED*/
static void
aus_fchmodat(struct t_audit_data *tad)
{
        klwp_t *clwp = ttolwp(curthread);
```



```
        uint32_t fmode;
        uint32_t fd;
        struct file  *fp;
        struct vnode *vp;
        struct f_audit_data *fad;

        struct a {
                long    fd;
                long    fname;          /* char * */
                long    fmode;
                long    flag;
        } *uap = (struct a *)clwp->lwp_ap;

        fd = (uint32_t)uap->fd;
        fmode = (uint32_t)uap->fmode;

        au_uwrite(au_to_arg32(2, "new file mode", fmode&07777));

        if (fd == AT_FDCWD || uap->fname != NULL)       /* same as chmod() */
                return;

        /*
         * convert file pointer to file descriptor
         *   Note: fd ref count incremented here.
         */
        if ((fp = getf(fd)) == NULL)
                return;

        /* get path from file struct here */
        fad = F2A(fp);
        if (fad->fad_aupath != NULL) {
                au_uwrite(au_to_path(fad->fad_aupath));
        } else {
                au_uwrite(au_to_arg32(1, "no path: fd", fd));
        }

        vp = fp->f_vnode;
        audit_attributes(vp);

        /* decrement file descriptor reference count */
        releasef(fd);
}

/*
 * convert open mode to appropriate open event
 */
au_event_t
open_event(uint_t fm)
{
        au_event_t e;

        switch (fm & (O_ACCMODE | O_CREAT | O_TRUNC)) {
        case O_RDONLY:
                e = AUE_OPEN_R;
                break;
        case O_RDONLY | O_CREAT:
                e = AUE_OPEN_RC;
                break;
        case O_RDONLY | O_TRUNC:
                e = AUE_OPEN_RT;
                break;
        case O_RDONLY | O_TRUNC | O_CREAT:
                e = AUE_OPEN_RTC;
                break;
        case O_WRONLY:
                e = AUE_OPEN_W;
```

```
1064                 break;
1065         case O_WRONLY | O_CREAT:
1066                 e = AUE_OPEN_WC;
1067                 break;
1068         case O_WRONLY | O_TRUNC:
1069                 e = AUE_OPEN_WT;
1070                 break;
1071         case O_WRONLY | O_TRUNC | O_CREAT:
1072                 e = AUE_OPEN_WTC;
1073                 break;
1074         case O_RDWR:
1075                 e = AUE_OPEN_RW;
1076                 break;
1077         case O_RDWR | O_CREAT:
1078                 e = AUE_OPEN_RWC;
1079                 break;
1080         case O_RDWR | O_TRUNC:
1081                 e = AUE_OPEN_RWT;
1082                 break;
1083         case O_RDWR | O_TRUNC | O_CREAT:
1084                 e = AUE_OPEN_RWTC;
1085                 break;
1086         case O_SEARCH:
1087                 e = AUE_OPEN_S;
1088                 break;
1089         case O_EXEC:
1090                 e = AUE_OPEN_E;
1091                 break;
1092         default:
1093                 e = AUE_NULL;
1094                 break;
1095         }

1097         return (e);
1098 }

1100 /* ARGSUSED */
1101 static au_event_t
1102 aui_open(au_event_t e)
1103 {
1104         klwp_t *clwp = ttolwp(curthread);
1105         uint_t fm;

1107         struct a {
1108                 long    fnamep;         /* char * */
1109                 long    fmode;
1110                 long    cmode;
1111         } *uap = (struct a *)clwp->lwp_ap;

1113         fm = (uint_t)uap->fmode;

1115         return (open_event(fm));
1116 }

1118 static void
1119 aus_open(struct t_audit_data *tad)
1120 {
1121         klwp_t *clwp = ttolwp(curthread);
1122         uint_t fm;

1124         struct a {
1125                 long    fnamep;         /* char * */
1126                 long    fmode;
1127                 long    cmode;
1128         } *uap = (struct a *)clwp->lwp_ap;
```



```
1130         fm = (uint_t)uap->fmode;

1132         /* If no write, create, or trunc modes, mark as a public op */
1133         if ((fm & (O_RDONLY|O_WRONLY|O_RDWR|O_CREAT|O_TRUNC)) == O_RDONLY)
1134                 tad->tad_ctrl |= TAD_PUBLIC_EV;
1135 }

1137 /* ARGSUSED */
1138 static au_event_t
1139 aui_openat(au_event_t e)
1140 {
1141         t_audit_data_t *tad = T2A(curthread);
1142         klwp_t *clwp = ttolwp(curthread);
1143         uint_t fm;

1145         struct a {
1146                 long    filedes;
1147                 long    fnamep;         /* char * */
1148                 long    fmode;
1149                 long    cmode;
1150         } *uap = (struct a *)clwp->lwp_ap;

1152         fm = (uint_t)uap->fmode;

1154         /*
1155          * __openattrdirat() does an extra pathname lookup in order to
1156          * enter the extended system attribute namespace of the referenced
1157          * extended attribute filename.
1158          */
1159         if (fm & FXATTRDIROPEN)
1160                 tad->tad_ctrl |= TAD_MLD;

1162         return (open_event(fm));
1163 }

1165 static void
1166 aus_openat(struct t_audit_data *tad)
1167 {
1168         klwp_t *clwp = ttolwp(curthread);
1169         uint_t fm;

1171         struct a {
1172                 long    filedes;
1173                 long    fnamep;         /* char * */
1174                 long    fmode;
1175                 long    cmode;
1176         } *uap = (struct a *)clwp->lwp_ap;

1178         fm = (uint_t)uap->fmode;

1180         /* If no write, create, or trunc modes, mark as a public op */
1181         if ((fm & (O_RDONLY|O_WRONLY|O_RDWR|O_CREAT|O_TRUNC)) == O_RDONLY)
1182                 tad->tad_ctrl |= TAD_PUBLIC_EV;
1183 }

1185 static au_event_t
1186 aui_unlinkat(au_event_t e)
1187 {
1188         klwp_t *clwp = ttolwp(curthread);

1190         struct a {
1191                 long    filedes;
1192                 long    fnamep;         /* char * */
1193                 long    flags;
1194         } *uap = (struct a *)clwp->lwp_ap;
```

```
1196         if (uap->flags & AT_REMOVEDIR)
1197                 e = AUE_RMDIR;
1198         else
1199                 e = AUE_UNLINK;

1201         return (e);
1202 }

1204 static au_event_t
1205 aui_fstatat(au_event_t e)
1206 {
1207         klwp_t *clwp = ttolwp(curthread);

1209         struct a {
1210                 long    filedes;
1211                 long    fnamep;         /* char * */
1212                 long    statb;
1213                 long    flags;
1214         } *uap = (struct a *)clwp->lwp_ap;

1216         if (uap->fnamep == NULL)
1217                 e = AUE_FSTAT;
1218         else if (uap->flags & AT_SYMLINK_NOFOLLOW)
1219                 e = AUE_LSTAT;
1220         else
1221                 e = AUE_STAT;

1223         return (e);
1224 }

1226 /* msgsys */
1227 static au_event_t
1228 aui_msgsys(au_event_t e)
1229 {
1230         klwp_t *clwp = ttolwp(curthread);
1231         uint_t fm;

1233         struct a {
1234                 long    id;     /* function code id */
1235                 long    ap;     /* arg pointer for recvmsg */
1236         } *uap = (struct a *)clwp->lwp_ap;

1238         struct b {
1239                 long    msgid;
1240                 long    cmd;
1241                 long    buf;    /* struct msqid_ds * */
1242         } *uap1 = (struct b *)&clwp->lwp_ap[1];

1244         fm  = (uint_t)uap->id;

1246         switch (fm) {
1247         case 0:         /* msgget */
1248                 e = AUE_MSGGET;
1249                 break;
1250         case 1:         /* msgctl */
1251                 switch ((uint_t)uap1->cmd) {
1252                 case IPC_RMID:
1253                         e = AUE_MSGCTL_RMID;
1254                         break;
1255                 case IPC_SET:
1256                         e = AUE_MSGCTL_SET;
1257                         break;
1258                 case IPC_STAT:
1259                         e = AUE_MSGCTL_STAT;
1260                         break;
1261                 default:
```



```
1262                         e = AUE_MSGCTL;
1263                         break;
1264                 }
1265                 break;
1266         case 2:         /* msgrcv */
1267                 e = AUE_MSGRCV;
1268                 break;
1269         case 3:         /* msgsnd */
1270                 e = AUE_MSGSND;
1271                 break;
1272         default:        /* illegal system call */
1273                 e = AUE_NULL;
1274                 break;
1275         }

1277         return (e);
1278 }


1281 /* shmsys */
1282 static au_event_t
1283 aui_shmsys(au_event_t e)
1284 {
1285         klwp_t *clwp = ttolwp(curthread);
1286         int fm;

1288         struct a {              /* shmsys */
1289                 long    id;     /* function code id */
1290         } *uap = (struct a *)clwp->lwp_ap;

1292         struct b {              /* ctrl */
1293                 long    shmid;
1294                 long    cmd;
1295                 long    arg;            /* struct shmid_ds * */
1296         } *uap1 = (struct b *)&clwp->lwp_ap[1];
1297         fm  = (uint_t)uap->id;

1299         switch (fm) {
1300         case 0:         /* shmat */
1301                 e = AUE_SHMAT;
1302                 break;
1303         case 1:         /* shmctl */
1304                 switch ((uint_t)uap1->cmd) {
1305                 case IPC_RMID:
1306                         e = AUE_SHMCTL_RMID;
1307                         break;
1308                 case IPC_SET:
1309                         e = AUE_SHMCTL_SET;
1310                         break;
1311                 case IPC_STAT:
1312                         e = AUE_SHMCTL_STAT;
1313                         break;
1314                 default:
1315                         e = AUE_SHMCTL;
1316                         break;
1317                 }
1318                 break;
1319         case 2:         /* shmdt */
1320                 e = AUE_SHMDT;
1321                 break;
1322         case 3:         /* shmget */
1323                 e = AUE_SHMGET;
1324                 break;
1325         default:        /* illegal system call */
1326                 e = AUE_NULL;
1327                 break;
```

```
1328         }

1330         return (e);
1331 }


1334 /* semsys */
1335 static au_event_t
1336 aui_semsys(au_event_t e)
1337 {
1338         klwp_t *clwp = ttolwp(curthread);
1339         uint_t fm;

1341         struct a {              /* semsys */
1342                 long    id;
1343         } *uap = (struct a *)clwp->lwp_ap;

1345         struct b {              /* ctrl */
1346                 long    semid;
1347                 long    semnum;
1348                 long    cmd;
1349                 long    arg;
1350         } *uap1 = (struct b *)&clwp->lwp_ap[1];

1352         fm = (uint_t)uap->id;

1354         switch (fm) {
1355         case 0:         /* semctl */
1356                 switch ((uint_t)uap1->cmd) {
1357                 case IPC_RMID:
1358                         e = AUE_SEMCTL_RMID;
1359                         break;
1360                 case IPC_SET:
1361                         e = AUE_SEMCTL_SET;
1362                         break;
1363                 case IPC_STAT:
1364                         e = AUE_SEMCTL_STAT;
1365                         break;
1366                 case GETNCNT:
1367                         e = AUE_SEMCTL_GETNCNT;
1368                         break;
1369                 case GETPID:
1370                         e = AUE_SEMCTL_GETPID;
1371                         break;
1372                 case GETVAL:
1373                         e = AUE_SEMCTL_GETVAL;
1374                         break;
1375                 case GETALL:
1376                         e = AUE_SEMCTL_GETALL;
1377                         break;
1378                 case GETZCNT:
1379                         e = AUE_SEMCTL_GETZCNT;
1380                         break;
1381                 case SETVAL:
1382                         e = AUE_SEMCTL_SETVAL;
1383                         break;
1384                 case SETALL:
1385                         e = AUE_SEMCTL_SETALL;
1386                         break;
1387                 default:
1388                         e = AUE_SEMCTL;
1389                         break;
1390                 }
1391                 break;
1392         case 1:         /* semget */
1393                 e = AUE_SEMGET;
```



```
1394                 break;
1395         case 2:         /* semop */
1396                 e = AUE_SEMOP;
1397                 break;
1398         default:        /* illegal system call */
1399                 e = AUE_NULL;
1400                 break;
1401         }

1403         return (e);
1404 }

1406 /* utssys - uname(2), ustat(2), fusers(2) */
1407 static au_event_t
1408 aui_utssys(au_event_t e)
1409 {
1410         klwp_t *clwp = ttolwp(curthread);
1411         uint_t type;

1413         struct a {
1414                 union {
1415                         long    cbuf;           /* char * */
1416                         long    ubuf;           /* struct stat * */
1417                 } ub;
1418                 union {
1419                         long    mv;     /* for USTAT */
1420                         long    flags;  /* for FUSERS */
1421                 } un;
1422                 long    type;
1423                 long    outbp;          /* char * for FUSERS */
1424         } *uap = (struct a *)clwp->lwp_ap;

1426         type = (uint_t)uap->type;

1428         if (type == UTS_FUSERS)
1429                 return (e);
1430         else
1431                 return ((au_event_t)AUE_NULL);
1432 }

1434 static au_event_t
1435 aui_fcntl(au_event_t e)
1436 {
1437         klwp_t *clwp = ttolwp(curthread);
1438         uint_t cmd;

1440         struct a {
1441                 long    fdes;
1442                 long    cmd;
1443                 long    arg;
1444         } *uap = (struct a *)clwp->lwp_ap;

1446         cmd = (uint_t)uap->cmd;

1448         switch (cmd) {
1449         case F_GETLK:
1450         case F_SETLK:
1451         case F_SETLKW:
1452                 break;
1453         case F_SETFL:
1454         case F_GETFL:
1455         case F_GETFD:
1456                 break;
1457         default:
1458                 e = (au_event_t)AUE_NULL;
1459                 break;
```

```
1460         }
1461         return ((au_event_t)e);
1462 }

1464 /* null function for now */
1465 static au_event_t
1466 aui_execve(au_event_t e)
1467 {
1468         return (e);
1469 }

1471 /*ARGSUSED*/
1472 static void
1473 aus_fcntl(struct t_audit_data *tad)
1474 {
1475         klwp_t *clwp = ttolwp(curthread);
1476         uint32_t cmd, fd, flags;
1477         struct file  *fp;
1478         struct vnode *vp;
1479         struct f_audit_data *fad;

1481         struct a {
1482                 long    fd;
1483                 long    cmd;
1484                 long    arg;
1485         } *uap = (struct a *)clwp->lwp_ap;

1487         cmd     = (uint32_t)uap->cmd;
1488         fd      = (uint32_t)uap->fd;
1489         flags   = (uint32_t)uap->arg;

1491         au_uwrite(au_to_arg32(2, "cmd", cmd));

1493         if (cmd == F_SETFL)
1494                 au_uwrite(au_to_arg32(3, "flags", flags));

1496                 /*
1497                  * convert file pointer to file descriptor
1498                  *   Note: fd ref count incremented here.
1499                  */
1500         if ((fp = getf(fd)) == NULL)
1501                 return;

1503         /* get path from file struct here */
1504         fad = F2A(fp);
1505         if (fad->fad_aupath != NULL) {
1506                 au_uwrite(au_to_path(fad->fad_aupath));
1507         } else {
1508                 au_uwrite(au_to_arg32(1, "no path: fd", fd));
1509         }

1511         vp = fp->f_vnode;
1512         audit_attributes(vp);

1514         /* decrement file descriptor reference count */
1515         releasef(fd);
1516 }

1518 /*ARGSUSED*/
1519 static void
1520 aus_kill(struct t_audit_data *tad)
1521 {
1522         klwp_t *clwp = ttolwp(curthread);
1523         struct proc *p;
1524         uint32_t signo;
1525         uid_t uid, ruid;
```



```
1526         gid_t gid, rgid;
1527         pid_t pid;
1528         const auditinfo_addr_t *ainfo;
1529         cred_t *cr;

1531         struct a {
1532                 long    pid;
1533                 long    signo;
1534         } *uap = (struct a *)clwp->lwp_ap;

1536         pid   = (pid_t)uap->pid;
1537         signo = (uint32_t)uap->signo;

1539         au_uwrite(au_to_arg32(2, "signal", signo));
1540         if (pid > 0) {
1541                 mutex_enter(&pidlock);
1542                 if (((p = prfind(pid)) == (struct proc *)0) ||
1543                     (p->p_stat == SIDL)) {
1544                         mutex_exit(&pidlock);
1545                         au_uwrite(au_to_arg32(1, "process", (uint32_t)pid));
1546                         return;
1547                 }
1548                 mutex_enter(&p->p_lock); /* so process doesn't go away */
1549                 mutex_exit(&pidlock);

1551                 mutex_enter(&p->p_crlock);
1552                 crhold(cr = p->p_cred);
1553                 mutex_exit(&p->p_crlock);
1554                 mutex_exit(&p->p_lock);

1556                 ainfo = crgetauinfo(cr);
1557                 if (ainfo == NULL) {
1558                         crfree(cr);
1559                         au_uwrite(au_to_arg32(1, "process", (uint32_t)pid));
1560                         return;
1561                 }

1563                 uid  = crgetuid(cr);
1564                 gid  = crgetgid(cr);
1565                 ruid = crgetruid(cr);
1566                 rgid = crgetrgid(cr);
1567                 au_uwrite(au_to_process(uid, gid, ruid, rgid, pid,
1568                     ainfo->ai_auid, ainfo->ai_asid, &ainfo->ai_termid));

1570                 if (is_system_labeled())
1571                         au_uwrite(au_to_label(CR_SL(cr)));

1573                 crfree(cr);
1574         }
1575         else
1576                 au_uwrite(au_to_arg32(1, "process", (uint32_t)pid));
1577 }

1579 /*ARGSUSED*/
1580 static void
1581 aus_mkdir(struct t_audit_data *tad)
1582 {
1583         klwp_t *clwp = ttolwp(curthread);
1584         uint32_t dmode;

1586         struct a {
1587                 long    dirnamep;               /* char * */
1588                 long    dmode;
1589         } *uap = (struct a *)clwp->lwp_ap;

1591         dmode = (uint32_t)uap->dmode;
```

```
1593         au_uwrite(au_to_arg32(2, "mode", dmode));
1594 }

1596 /*ARGSUSED*/
1597 static void
1598 aus_mkdirat(struct t_audit_data *tad)
1599 {
1600         klwp_t *clwp = ttolwp(curthread);
1601         uint32_t dmode;

1603         struct a {
1604                 long    fd;
1605                 long    dirnamep;               /* char * */
1606                 long    dmode;
1607         } *uap = (struct a *)clwp->lwp_ap;

1609         dmode = (uint32_t)uap->dmode;

1611         au_uwrite(au_to_arg32(2, "mode", dmode));
1612 }

1614 /*ARGSUSED*/
1615 static void
1616 aus_mknod(struct t_audit_data *tad)
1617 {
1618         klwp_t *clwp = ttolwp(curthread);
1619         uint32_t fmode;
1620         dev_t dev;

1622         struct a {
1623                 long    pnamep;         /* char * */
1624                 long    fmode;
1625                 long    dev;
1626         } *uap = (struct a *)clwp->lwp_ap;

1628         fmode = (uint32_t)uap->fmode;
1629         dev   = (dev_t)uap->dev;

1631         au_uwrite(au_to_arg32(2, "mode", fmode));
1632 #ifdef _LP64
1633         au_uwrite(au_to_arg64(3, "dev", dev));
1634 #else
1635         au_uwrite(au_to_arg32(3, "dev", dev));
1636 #endif
1637 }

1639 /*ARGSUSED*/
1640 static void
1641 auf_mknod(struct t_audit_data *tad, int error, rval_t *rval)
1642 {
1643         klwp_t *clwp = ttolwp(curthread);
1644         vnode_t *dvp;
1645         caddr_t pnamep;

1647         struct a {
1648                 long    pnamep;         /* char * */
1649                 long    fmode;
1650                 long    dev;
1651         } *uap = (struct a *)clwp->lwp_ap;

1653         /* no error, then already path token in audit record */
1654         if (error != EPERM && error != EINVAL)
1655                 return;

1657         /* do the lookup to force generation of path token */
```



```
1658         pnamep = (caddr_t)uap->pnamep;
1659         tad->tad_ctrl |= TAD_NOATTRB;
1660         error = lookupname(pnamep, UIO_USERSPACE, NO_FOLLOW, &dvp, NULLVPP);
1661         if (error == 0)
1662                 VN_RELE(dvp);
1663 }

1665 /*ARGSUSED*/
1666 static void
1667 aus_mknodat(struct t_audit_data *tad)
1668 {
1669         klwp_t *clwp = ttolwp(curthread);
1670         uint32_t fmode;
1671         dev_t dev;

1673         struct a {
1674                 long    fd;
1675                 long    pnamep;         /* char * */
1676                 long    fmode;
1677                 long    dev;
1678         } *uap = (struct a *)clwp->lwp_ap;

1680         fmode = (uint32_t)uap->fmode;
1681         dev   = (dev_t)uap->dev;

1683         au_uwrite(au_to_arg32(2, "mode", fmode));
1684 #ifdef _LP64
1685         au_uwrite(au_to_arg64(3, "dev", dev));
1686 #else
1687         au_uwrite(au_to_arg32(3, "dev", dev));
1688 #endif
1689 }

1691 /*ARGSUSED*/
1692 static void
1693 auf_mknodat(struct t_audit_data *tad, int error, rval_t *rval)
1694 {
1695         klwp_t *clwp = ttolwp(curthread);
1696         vnode_t *startvp;
1697         vnode_t *dvp;
1698         caddr_t pnamep;
1699         int fd;

1701         struct a {
1702                 long    fd;
1703                 long    pnamep;         /* char * */
1704                 long    fmode;
1705                 long    dev;
1706         } *uap = (struct a *)clwp->lwp_ap;

1708         /* no error, then already path token in audit record */
1709         if (error != EPERM && error != EINVAL)
1710                 return;

1712         /* do the lookup to force generation of path token */
1713         fd = (int)uap->fd;
1714         pnamep = (caddr_t)uap->pnamep;
1715         if (pnamep == NULL ||
1716             fgetstartvp(fd, pnamep, &startvp) != 0)
1717                 return;
1718         tad->tad_ctrl |= TAD_NOATTRB;
1719         error = lookupnameat(pnamep, UIO_USERSPACE, NO_FOLLOW, &dvp, NULLVPP,
1720             startvp);
1721         if (error == 0)
1722                 VN_RELE(dvp);
1723         if (startvp != NULL)
```

```
1724                 VN_RELE(startvp);
1725 }

1727 /*ARGSUSED*/
1728 static void
1729 aus_mount(struct t_audit_data *tad)
1730 {       /* AUS_START */
1731         klwp_t *clwp = ttolwp(curthread);
1732         uint32_t flags;
1733         uintptr_t u_fstype, dataptr;
1734         STRUCT_DECL(nfs_args, nfsargs);
1735         size_t len;
1736         char *fstype, *hostname;

1738         struct a {
1739                 long    spec;           /* char    * */
1740                 long    dir;            /* char    * */
1741                 long    flags;
1742                 long    fstype;         /* char    * */
1743                 long    dataptr;        /* char    * */
1744                 long    datalen;
1745         } *uap = (struct a *)clwp->lwp_ap;

1747         u_fstype = (uintptr_t)uap->fstype;
1748         flags    = (uint32_t)uap->flags;
1749         dataptr  = (uintptr_t)uap->dataptr;

1751         fstype = kmem_alloc(MAXNAMELEN, KM_SLEEP);
1752         if (copyinstr((caddr_t)u_fstype, (caddr_t)fstype, MAXNAMELEN, &len))
1753                 goto mount_free_fstype;

1755         au_uwrite(au_to_arg32(3, "flags", flags));
1756         au_uwrite(au_to_text(fstype));

1758         if (strncmp(fstype, "nfs", 3) == 0) {

1760                 STRUCT_INIT(nfsargs, get_udatamodel());
1761                 bzero(STRUCT_BUF(nfsargs), STRUCT_SIZE(nfsargs));

1763                 if (copyin((caddr_t)dataptr,
1764                                 STRUCT_BUF(nfsargs),
1765                                 MIN(uap->datalen, STRUCT_SIZE(nfsargs)))) {
1766                         /* DEBUG debug_enter((char *)NULL); */
1767                         goto mount_free_fstype;
1768                 }
1769                 hostname = kmem_alloc(MAXNAMELEN, KM_SLEEP);
1770                 if (copyinstr(STRUCT_FGETP(nfsargs, hostname),
1771                                 (caddr_t)hostname,
1772                                 MAXNAMELEN, &len)) {
1773                         goto mount_free_hostname;
1774                 }
1775                 au_uwrite(au_to_text(hostname));
1776                 au_uwrite(au_to_arg32(3, "internal flags",
1777                         (uint_t)STRUCT_FGET(nfsargs, flags)));

1779 mount_free_hostname:
1780                 kmem_free(hostname, MAXNAMELEN);
1781         }

1783 mount_free_fstype:
1784         kmem_free(fstype, MAXNAMELEN);
1785 }       /* AUS_MOUNT */

1787 static void
1788 aus_umount_path(caddr_t umount_dir)
1789 {
```



```
1790         char                    *dir_path;
1791         struct audit_path       *path;
1792         size_t                  path_len, dir_len;

1794         /* length alloc'd for two string pointers */
1795         path_len = sizeof (struct audit_path) + sizeof (char *);
1796         path = kmem_alloc(path_len, KM_SLEEP);
1797         dir_path = kmem_alloc(MAXPATHLEN, KM_SLEEP);

1799         if (copyinstr(umount_dir, (caddr_t)dir_path,
1800             MAXPATHLEN, &dir_len))
1801                 goto umount2_free_dir;

1803         /*
1804          * the audit_path struct assumes that the buffer pointed to
1805          * by audp_sect[n] contains string 0 immediatedly followed
1806          * by string 1.
1807          */
1808         path->audp_sect[0] = dir_path;
1809         path->audp_sect[1] = dir_path + strlen(dir_path) + 1;
1810         path->audp_size = path_len;
1811         path->audp_ref = 1;             /* not used */
1812         path->audp_cnt = 1;             /* one path string */

1814         au_uwrite(au_to_path(path));

1816 umount2_free_dir:
1817         kmem_free(dir_path, MAXPATHLEN);
1818         kmem_free(path, path_len);
1819 }

1821 /*ARGSUSED*/
1822 static void
1823 aus_umount2(struct t_audit_data *tad)
1824 {
1825         klwp_t                  *clwp = ttolwp(curthread);
1826         struct a {
1827                 long    dir;            /* char    * */
1828                 long    flags;
1829         } *uap = (struct a *)clwp->lwp_ap;

1831         aus_umount_path((caddr_t)uap->dir);

1833         au_uwrite(au_to_arg32(2, "flags", (uint32_t)uap->flags));
1834 }

1836 static void
1837 aus_msgsys(struct t_audit_data *tad)
1838 {
1839         klwp_t *clwp = ttolwp(curthread);
1840         uint32_t msgid;

1842         struct b {
1843                 long    msgid;
1844                 long    cmd;
1845                 long    buf;            /* struct msqid_ds * */
1846         } *uap1 = (struct b *)&clwp->lwp_ap[1];

1848         msgid = (uint32_t)uap1->msgid;


1851         switch (tad->tad_event) {
1852         case AUE_MSGGET:                /* msgget */
1853                 au_uwrite(au_to_arg32(1, "msg key", msgid));
1854                 break;
1855         case AUE_MSGCTL:                /* msgctl */
```

```
1856         case AUE_MSGCTL_RMID:           /* msgctl */
1857         case AUE_MSGCTL_SET:            /* msgctl */
1858         case AUE_MSGCTL_STAT:           /* msgctl */
1859         case AUE_MSGRCV:                /* msgrcv */
1860         case AUE_MSGSND:                /* msgsnd */
1861                 au_uwrite(au_to_arg32(1, "msg ID", msgid));
1862                 break;
1863         }
1864 }

1866 /*ARGSUSED*/
1867 static void
1868 auf_msgsys(struct t_audit_data *tad, int error, rval_t *rval)
1869 {
1870         int id;

1872         if (error != 0)
1873                 return;
1874         if (tad->tad_event == AUE_MSGGET) {
1875                 uint32_t scid;
1876                 uint32_t sy_flags;

1878                 /* need to determine type of executing binary */
1879                 scid = tad->tad_scid;
1880 #ifdef _SYSCALL32_IMPL
1881                 if (lwp_getdatamodel(ttolwp(curthread)) == DATAMODEL_NATIVE)
1882                         sy_flags = sysent[scid].sy_flags & SE_RVAL_MASK;
1883                 else
1884                         sy_flags = sysent32[scid].sy_flags & SE_RVAL_MASK;
1885 #else
1886                 sy_flags = sysent[scid].sy_flags & SE_RVAL_MASK;
1887 #endif
1888                 if (sy_flags == SE_32RVAL1)
1889                         id = rval->r_val1;
1890                 if (sy_flags == (SE_32RVAL2|SE_32RVAL1))
1891                         id = rval->r_val1;
1892                 if (sy_flags == SE_64RVAL)
1893                         id = (int)rval->r_vals;

1895                 au_uwrite(au_to_ipc(AT_IPC_MSG, id));
1896         }
1897 }

1899 static void
1900 aus_semsys(struct t_audit_data *tad)
1901 {
1902         klwp_t *clwp = ttolwp(curthread);
1903         uint32_t semid;

1905         struct b {              /* ctrl */
1906                 long    semid;
1907                 long    semnum;
1908                 long    cmd;
1909                 long    arg;
1910         } *uap1 = (struct b *)&clwp->lwp_ap[1];

1912         semid = (uint32_t)uap1->semid;

1914         switch (tad->tad_event) {
1915         case AUE_SEMCTL_RMID:
1916         case AUE_SEMCTL_STAT:
1917         case AUE_SEMCTL_GETNCNT:
1918         case AUE_SEMCTL_GETPID:
1919         case AUE_SEMCTL_GETVAL:
1920         case AUE_SEMCTL_GETALL:
1921         case AUE_SEMCTL_GETZCNT:
```



```
1922         case AUE_SEMCTL_SET:
1923         case AUE_SEMCTL_SETVAL:
1924         case AUE_SEMCTL_SETALL:
1925         case AUE_SEMCTL:
1926         case AUE_SEMOP:
1927                 au_uwrite(au_to_arg32(1, "sem ID", semid));
1928                 break;
1929         case AUE_SEMGET:
1930                 au_uwrite(au_to_arg32(1, "sem key", semid));
1931                 break;
1932         }
1933 }

1935 /*ARGSUSED*/
1936 static void
1937 auf_semsys(struct t_audit_data *tad, int error, rval_t *rval)
1938 {
1939         int id;

1941         if (error != 0)
1942                 return;
1943         if (tad->tad_event == AUE_SEMGET) {
1944                 uint32_t scid;
1945                 uint32_t sy_flags;

1947                 /* need to determine type of executing binary */
1948                 scid = tad->tad_scid;
1949 #ifdef _SYSCALL32_IMPL
1950                 if (lwp_getdatamodel(ttolwp(curthread)) == DATAMODEL_NATIVE)
1951                         sy_flags = sysent[scid].sy_flags & SE_RVAL_MASK;
1952                 else
1953                         sy_flags = sysent32[scid].sy_flags & SE_RVAL_MASK;
1954 #else
1955                 sy_flags = sysent[scid].sy_flags & SE_RVAL_MASK;
1956 #endif
1957                 if (sy_flags == SE_32RVAL1)
1958                         id = rval->r_val1;
1959                 if (sy_flags == (SE_32RVAL2|SE_32RVAL1))
1960                         id = rval->r_val1;
1961                 if (sy_flags == SE_64RVAL)
1962                         id = (int)rval->r_vals;

1964                 au_uwrite(au_to_ipc(AT_IPC_SEM, id));
1965         }
1966 }

1968 /*ARGSUSED*/
1969 static void
1970 aus_close(struct t_audit_data *tad)
1971 {
1972         klwp_t *clwp = ttolwp(curthread);
1973         uint32_t fd;
1974         struct file *fp;
1975         struct f_audit_data *fad;
1976         struct vnode *vp;
1977         struct vattr attr;
1978         au_kcontext_t   *kctx = GET_KCTX_PZ;

1980         struct a {
1981                 long    i;
1982         } *uap = (struct a *)clwp->lwp_ap;

1984         fd = (uint32_t)uap->i;

1986         attr.va_mask = 0;
1987         au_uwrite(au_to_arg32(1, "fd", fd));
```

```
1989                 /*
1990                  * convert file pointer to file descriptor
1991                  *   Note: fd ref count incremented here.
1992                  */
1993         if ((fp = getf(fd)) == NULL)
1994                 return;

1996         fad = F2A(fp);
1997         tad->tad_evmod = (au_emod_t)fad->fad_flags;
1998         if (fad->fad_aupath != NULL) {
1999                 au_uwrite(au_to_path(fad->fad_aupath));
2000                 if ((vp = fp->f_vnode) != NULL) {
2001                         attr.va_mask = AT_ALL;
2002                         if (VOP_GETATTR(vp, &attr, 0, CRED(), NULL) == 0) {
2003                                 /*
2004                                  * When write was not used and the file can be
2005                                  * considered public, skip the audit.
2006                                  */
2007                                 if (((fp->f_flag & FWRITE) == 0) &&
2008                                     object_is_public(&attr)) {
2009                                         tad->tad_flag = 0;
2010                                         tad->tad_evmod = 0;
2011                                         /* free any residual audit data */
2012                                         au_close(kctx, &(u_ad), 0, 0, 0, NULL);
2013                                         releasef(fd);
2014                                         return;
2015                                 }
2016                                 au_uwrite(au_to_attr(&attr));
2017                                 audit_sec_attributes(&(u_ad), vp);
2018                         }
2019                 }
2020         }

2022         /* decrement file descriptor reference count */
2023         releasef(fd);
2024 }

2026 /*ARGSUSED*/
2027 static void
2028 aus_fstatfs(struct t_audit_data *tad)
2029 {
2030         klwp_t *clwp = ttolwp(curthread);
2031         uint32_t fd;
2032         struct file  *fp;
2033         struct vnode *vp;
2034         struct f_audit_data *fad;

2036         struct a {
2037                 long    fd;
2038                 long    buf;            /* struct statfs * */
2039         } *uap = (struct a *)clwp->lwp_ap;

2041         fd = (uint_t)uap->fd;

2043                 /*
2044                  * convert file pointer to file descriptor
2045                  *   Note: fd ref count incremented here.
2046                  */
2047         if ((fp = getf(fd)) == NULL)
2048                 return;

2050                 /* get path from file struct here */
2051         fad = F2A(fp);
2052         if (fad->fad_aupath != NULL) {
2053                 au_uwrite(au_to_path(fad->fad_aupath));
```



```
2054         } else {
2055                 au_uwrite(au_to_arg32(1, "no path: fd", fd));
2056         }

2058         vp = fp->f_vnode;
2059         audit_attributes(vp);

2061         /* decrement file descriptor reference count */
2062         releasef(fd);
2063 }

2065 static au_event_t
2066 aui_setpgrp(au_event_t e)
2067 {
2068         klwp_t *clwp = ttolwp(curthread);
2069         int flag;

2071         struct a {
2072                 long    flag;
2073                 long    pid;
2074                 long    pgid;
2075         } *uap = (struct a *)clwp->lwp_ap;

2077         flag = (int)uap->flag;


2080         switch (flag) {

2082         case 1: /* setpgrp() */
2083                 e = AUE_SETPGRP;
2084                 break;

2086         case 3: /* setsid() */
2087                 e = AUE_SETSID;
2088                 break;

2090         case 5: /* setpgid() */
2091                 e = AUE_SETPGID;
2092                 break;

2094         case 0: /* getpgrp()    - not security relevant */
2095         case 2: /* getsid()     - not security relevant */
2096         case 4: /* getpgid()    - not security relevant */
2097                 e = AUE_NULL;
2098                 break;

2100         default:
2101                 e = AUE_NULL;
2102                 break;
2103         }

2105         return (e);
2106 }

2108 /*ARGSUSED*/
2109 static void
2110 aus_setpgrp(struct t_audit_data *tad)
2111 {
2112         klwp_t          *clwp = ttolwp(curthread);
2113         pid_t           pgid;
2114         struct proc     *p;
2115         uid_t           uid, ruid;
2116         gid_t           gid, rgid;
2117         pid_t           pid;
2118         cred_t          *cr;
2119         int             flag;
```

```
2120         const auditinfo_addr_t  *ainfo;

2122         struct a {
2123                 long    flag;
2124                 long    pid;
2125                 long    pgid;
2126         } *uap = (struct a *)clwp->lwp_ap;

2128         flag = (int)uap->flag;
2129         pid  = (pid_t)uap->pid;
2130         pgid = (pid_t)uap->pgid;


2133         switch (flag) {

2135         case 0: /* getpgrp() */
2136         case 1: /* setpgrp() */
2137         case 2: /* getsid() */
2138         case 3: /* setsid() */
2139         case 4: /* getpgid() */
2140                 break;

2142         case 5: /* setpgid() */

2144                 /* current process? */
2145                 if (pid == 0) {
2146                         return;
2147                 }

2149                 mutex_enter(&pidlock);
2150                 p = prfind(pid);
2151                 if (p == NULL || p->p_as == &kas ||
2152                     p->p_stat == SIDL || p->p_stat == SZOMB) {
2153                         mutex_exit(&pidlock);
2154                         return;
2155                 }
2156                 mutex_enter(&p->p_lock);        /* so process doesn't go away */
2157                 mutex_exit(&pidlock);

2159                 mutex_enter(&p->p_crlock);
2160                 crhold(cr = p->p_cred);
2161                 mutex_exit(&p->p_crlock);
2162                 mutex_exit(&p->p_lock);

2164                 ainfo = crgetauinfo(cr);
2165                 if (ainfo == NULL) {
2166                         crfree(cr);
2167                         return;
2168                 }

2170                 uid  = crgetuid(cr);
2171                 gid  = crgetgid(cr);
2172                 ruid = crgetruid(cr);
2173                 rgid = crgetrgid(cr);
2174                 au_uwrite(au_to_process(uid, gid, ruid, rgid, pid,
2175                     ainfo->ai_auid, ainfo->ai_asid, &ainfo->ai_termid));
2176                 crfree(cr);
2177                 au_uwrite(au_to_arg32(2, "pgid", pgid));
2178                 break;

2180         default:
2181                 break;
2182         }
2183 }
```



```
2186 /*ARGSUSED*/
2187 static void
2188 aus_setregid(struct t_audit_data *tad)
2189 {
2190         klwp_t *clwp = ttolwp(curthread);
2191         uint32_t rgid, egid;

2193         struct a {
2194                 long     rgid;
2195                 long     egid;
2196         } *uap = (struct a *)clwp->lwp_ap;

2198         rgid  = (uint32_t)uap->rgid;
2199         egid  = (uint32_t)uap->egid;

2201         au_uwrite(au_to_arg32(1, "rgid", rgid));
2202         au_uwrite(au_to_arg32(2, "egid", egid));
2203 }

2205 /*ARGSUSED*/
2206 static void
2207 aus_setgid(struct t_audit_data *tad)
2208 {
2209         klwp_t *clwp = ttolwp(curthread);
2210         uint32_t gid;

2212         struct a {
2213                 long    gid;
2214         } *uap = (struct a *)clwp->lwp_ap;

2216         gid = (uint32_t)uap->gid;

2218         au_uwrite(au_to_arg32(1, "gid", gid));
2219 }


2222 /*ARGSUSED*/
2223 static void
2224 aus_setreuid(struct t_audit_data *tad)
2225 {
2226         klwp_t *clwp = ttolwp(curthread);
2227         uint32_t ruid, euid;

2229         struct a {
2230                 long    ruid;
2231                 long    euid;
2232         } *uap = (struct a *)clwp->lwp_ap;

2234         ruid = (uint32_t)uap->ruid;
2235         euid  = (uint32_t)uap->euid;

2237         au_uwrite(au_to_arg32(1, "ruid", ruid));
2238         au_uwrite(au_to_arg32(2, "euid", euid));
2239 }


2242 /*ARGSUSED*/
2243 static void
2244 aus_setuid(struct t_audit_data *tad)
2245 {
2246         klwp_t *clwp = ttolwp(curthread);
2247         uint32_t uid;

2249         struct a {
2250                 long    uid;
2251         } *uap = (struct a *)clwp->lwp_ap;
```

```
2253         uid = (uint32_t)uap->uid;

2255         au_uwrite(au_to_arg32(1, "uid", uid));
2256 }

2258 /*ARGSUSED*/
2259 static void
2260 aus_shmsys(struct t_audit_data *tad)
2261 {
2262         klwp_t *clwp = ttolwp(curthread);
2263         uint32_t id, cmd;

2265         struct b {
2266                 long    id;
2267                 long    cmd;
2268                 long    buf;            /* struct shmid_ds * */
2269         } *uap1 = (struct b *)&clwp->lwp_ap[1];

2271         id  = (uint32_t)uap1->id;
2272         cmd = (uint32_t)uap1->cmd;

2274         switch (tad->tad_event) {
2275         case AUE_SHMGET:                        /* shmget */
2276                 au_uwrite(au_to_arg32(1, "shm key", id));
2277                 break;
2278         case AUE_SHMCTL:                        /* shmctl */
2279         case AUE_SHMCTL_RMID:                   /* shmctl */
2280         case AUE_SHMCTL_STAT:                   /* shmctl */
2281         case AUE_SHMCTL_SET:                    /* shmctl */
2282                 au_uwrite(au_to_arg32(1, "shm ID", id));
2283                 break;
2284         case AUE_SHMDT:                         /* shmdt */
2285                 au_uwrite(au_to_arg32(1, "shm adr", id));
2286                 break;
2287         case AUE_SHMAT:                         /* shmat */
2288                 au_uwrite(au_to_arg32(1, "shm ID", id));
2289                 au_uwrite(au_to_arg32(2, "shm adr", cmd));
2290                 break;
2291         }
2292 }

2294 /*ARGSUSED*/
2295 static void
2296 auf_shmsys(struct t_audit_data *tad, int error, rval_t *rval)
2297 {
2298         int id;

2300         if (error != 0)
2301                 return;
2302         if (tad->tad_event == AUE_SHMGET) {
2303                 uint32_t scid;
2304                 uint32_t sy_flags;

2306                 /* need to determine type of executing binary */
2307                 scid = tad->tad_scid;
2308 #ifdef _SYSCALL32_IMPL
2309                 if (lwp_getdatamodel(ttolwp(curthread)) == DATAMODEL_NATIVE)
2310                         sy_flags = sysent[scid].sy_flags & SE_RVAL_MASK;
2311                 else
2312                         sy_flags = sysent32[scid].sy_flags & SE_RVAL_MASK;
2313 #else
2314                 sy_flags = sysent[scid].sy_flags & SE_RVAL_MASK;
2315 #endif
2316                 if (sy_flags == SE_32RVAL1)
2317                         id = rval->r_val1;
```



```
2318                 if (sy_flags == (SE_32RVAL2|SE_32RVAL1))
2319                         id = rval->r_val1;
2320                 if (sy_flags == SE_64RVAL)
2321                         id = (int)rval->r_vals;
2322                 au_uwrite(au_to_ipc(AT_IPC_SHM, id));
2323         }
2324 }


2327 /*ARGSUSED*/
2328 static void
2329 aus_ioctl(struct t_audit_data *tad)
2330 {
2331         klwp_t *clwp = ttolwp(curthread);
2332         struct file *fp;
2333         struct vnode *vp;
2334         struct f_audit_data *fad;
2335         uint32_t fd, cmd;
2336         uintptr_t cmarg;

2338         /* XX64 */
2339         struct a {
2340                 long    fd;
2341                 long    cmd;
2342                 long    cmarg;          /* caddr_t */
2343         } *uap = (struct a *)clwp->lwp_ap;

2345         fd    = (uint32_t)uap->fd;
2346         cmd   = (uint32_t)uap->cmd;
2347         cmarg = (uintptr_t)uap->cmarg;

2349                 /*
2350                  * convert file pointer to file descriptor
2351                  *   Note: fd ref count incremented here.
2352                  */
2353         if ((fp = getf(fd)) == NULL) {
2354                 au_uwrite(au_to_arg32(1, "fd", fd));
2355                 au_uwrite(au_to_arg32(2, "cmd", cmd));
2356 #ifndef _LP64
2357                         au_uwrite(au_to_arg32(3, "arg", (uint32_t)cmarg));
2358 #else
2359                         au_uwrite(au_to_arg64(3, "arg", (uint64_t)cmarg));
2360 #endif
2361                 return;
2362         }

2364         /* get path from file struct here */
2365         fad = F2A(fp);
2366         if (fad->fad_aupath != NULL) {
2367                 au_uwrite(au_to_path(fad->fad_aupath));
2368         } else {
2369                 au_uwrite(au_to_arg32(1, "no path: fd", fd));
2370         }

2372         vp = fp->f_vnode;
2373         audit_attributes(vp);

2375         /* decrement file descriptor reference count */
2376         releasef(fd);

2378         au_uwrite(au_to_arg32(2, "cmd", cmd));
2379 #ifndef _LP64
2380                 au_uwrite(au_to_arg32(3, "arg", (uint32_t)cmarg));
2381 #else
2382                 au_uwrite(au_to_arg64(3, "arg", (uint64_t)cmarg));
2383 #endif
```

```
}

/*
 * null function for memcntl for now. We might want to limit memcntl()
 * auditing to commands: MC_LOCKAS, MC_LOCK, MC_UNLOCKAS, MC_UNLOCK which
 * require privileges.
 */
static au_event_t
aui_memcntl(au_event_t e)
{
        return (e);
}

/*ARGSUSED*/
static au_event_t
aui_privsys(au_event_t e)
{
        klwp_t *clwp = ttolwp(curthread);

        struct a {
                long    opcode;
        } *uap = (struct a *)clwp->lwp_ap;

        switch (uap->opcode) {
        case PRIVSYS_SETPPRIV:
                return (AUE_SETPPRIV);
        default:
                return (AUE_NULL);
        }
}

/*ARGSUSED*/
static void
aus_memcntl(struct t_audit_data *tad)
{
        klwp_t *clwp = ttolwp(curthread);

        struct a {
                long    addr;
                long    len;
                long    cmd;
                long    arg;
                long    attr;
                long    mask;
        } *uap = (struct a *)clwp->lwp_ap;

#ifdef _LP64
        au_uwrite(au_to_arg64(1, "base", (uint64_t)uap->addr));
        au_uwrite(au_to_arg64(2, "len", (uint64_t)uap->len));
#else
        au_uwrite(au_to_arg32(1, "base", (uint32_t)uap->addr));
        au_uwrite(au_to_arg32(2, "len", (uint32_t)uap->len));
#endif
        au_uwrite(au_to_arg32(3, "cmd", (uint_t)uap->cmd));
#ifdef _LP64
        au_uwrite(au_to_arg64(4, "arg", (uint64_t)uap->arg));
#else
        au_uwrite(au_to_arg32(4, "arg", (uint32_t)uap->arg));
#endif
        au_uwrite(au_to_arg32(5, "attr", (uint_t)uap->attr));
        au_uwrite(au_to_arg32(6, "mask", (uint_t)uap->mask));
}

/*ARGSUSED*/
static void
aus_mmap(struct t_audit_data *tad)
```



```
{
        klwp_t *clwp = ttolwp(curthread);
        struct file *fp;
        struct f_audit_data *fad;
        struct vnode *vp;
        uint32_t fd;

        struct a {
                long    addr;
                long    len;
                long    prot;
                long    flags;
                long    fd;
                long    pos;
        } *uap = (struct a *)clwp->lwp_ap;

        fd = (uint32_t)uap->fd;

#ifdef _LP64
        au_uwrite(au_to_arg64(1, "addr", (uint64_t)uap->addr));
        au_uwrite(au_to_arg64(2, "len", (uint64_t)uap->len));
#else
        au_uwrite(au_to_arg32(1, "addr", (uint32_t)uap->addr));
        au_uwrite(au_to_arg32(2, "len", (uint32_t)uap->len));
#endif

        if ((fp = getf(fd)) == NULL) {
                au_uwrite(au_to_arg32(5, "fd", (uint32_t)uap->fd));
                return;
        }

        /*
         * Mark in the tad if write access is NOT requested... if
         * this is later detected (in audit_attributes) to be a
         * public object, the mmap event may be discarded.
         */
        if (((uap->prot) & PROT_WRITE) == 0) {
                tad->tad_ctrl |= TAD_PUBLIC_EV;
        }

        fad = F2A(fp);
        if (fad->fad_aupath != NULL) {
                au_uwrite(au_to_path(fad->fad_aupath));
        } else {
                au_uwrite(au_to_arg32(1, "no path: fd", fd));
        }

        vp = (struct vnode *)fp->f_vnode;
        audit_attributes(vp);

        /* mark READ/WRITE since we can't predict access */
        if (uap->prot & PROT_READ)
                fad->fad_flags |= FAD_READ;
        if (uap->prot & PROT_WRITE)
                fad->fad_flags |= FAD_WRITE;

        /* decrement file descriptor reference count */
        releasef(fd);

}       /* AUS_MMAP */



/*ARGSUSED*/
static void
```

```
2516 aus_munmap(struct t_audit_data *tad)
2517 {
2518         klwp_t *clwp = ttolwp(curthread);

2520         struct a {
2521                 long    addr;
2522                 long    len;
2523         } *uap = (struct a *)clwp->lwp_ap;

2525 #ifdef _LP64
2526         au_uwrite(au_to_arg64(1, "addr", (uint64_t)uap->addr));
2527         au_uwrite(au_to_arg64(2, "len", (uint64_t)uap->len));
2528 #else
2529         au_uwrite(au_to_arg32(1, "addr", (uint32_t)uap->addr));
2530         au_uwrite(au_to_arg32(2, "len", (uint32_t)uap->len));
2531 #endif

2533 }       /* AUS_MUNMAP */









2541 /*ARGSUSED*/
2542 static void
2543 aus_priocntlsys(struct t_audit_data *tad)
2544 {
2545         klwp_t *clwp = ttolwp(curthread);

2547         struct a {
2548                 long    pc_version;
2549                 long    psp;            /* procset_t */
2550                 long    cmd;
2551                 long    arg;
2552         } *uap = (struct a *)clwp->lwp_ap;

2554         au_uwrite(au_to_arg32(1, "pc_version", (uint32_t)uap->pc_version));
2555         au_uwrite(au_to_arg32(3, "cmd", (uint32_t)uap->cmd));

2557 }       /* AUS_PRIOCNTLSYS */


2560 /*ARGSUSED*/
2561 static void
2562 aus_setegid(struct t_audit_data *tad)
2563 {
2564         klwp_t *clwp = ttolwp(curthread);
2565         uint32_t gid;

2567         struct a {
2568                 long    gid;
2569         } *uap = (struct a *)clwp->lwp_ap;

2571         gid = (uint32_t)uap->gid;

2573         au_uwrite(au_to_arg32(1, "gid", gid));
2574 }       /* AUS_SETEGID */




2579 /*ARGSUSED*/
2580 static void
2581 aus_setgroups(struct t_audit_data *tad)
```



```
2582 {
2583         klwp_t *clwp = ttolwp(curthread);
2584         int i;
2585         int gidsetsize;
2586         uintptr_t gidset;
2587         gid_t *gidlist;

2589         struct a {
2590                 long    gidsetsize;
2591                 long    gidset;
2592         } *uap = (struct a *)clwp->lwp_ap;

2594         gidsetsize = (uint_t)uap->gidsetsize;
2595         gidset = (uintptr_t)uap->gidset;

2597         if ((gidsetsize > NGROUPS_MAX_DEFAULT) || (gidsetsize < 0))
2598                 return;
2599         if (gidsetsize != 0) {
2600                 gidlist = kmem_alloc(gidsetsize * sizeof (gid_t),
2601                     KM_SLEEP);
2602                 if (copyin((caddr_t)gidset, gidlist,
2603                     gidsetsize * sizeof (gid_t)) == 0)
2604                         for (i = 0; i < gidsetsize; i++)
2605                                 au_uwrite(au_to_arg32(1, "setgroups",
2606                                     (uint32_t)gidlist[i]));
2607                 kmem_free(gidlist, gidsetsize * sizeof (gid_t));
2608         } else
2609                 au_uwrite(au_to_arg32(1, "setgroups", (uint32_t)0));

2611 }       /* AUS_SETGROUPS */




2617 /*ARGSUSED*/
2618 static void
2619 aus_seteuid(struct t_audit_data *tad)
2620 {
2621         klwp_t *clwp = ttolwp(curthread);
2622         uint32_t uid;

2624         struct a {
2625                 long    uid;
2626         } *uap = (struct a *)clwp->lwp_ap;

2628         uid = (uint32_t)uap->uid;

2630         au_uwrite(au_to_arg32(1, "euid", uid));

2632 }       /* AUS_SETEUID */

2634 /*ARGSUSED*/
2635 static void
2636 aus_putmsg(struct t_audit_data *tad)
2637 {
2638         klwp_t *clwp = ttolwp(curthread);
2639         uint32_t fd, pri;
2640         struct file *fp;
2641         struct f_audit_data *fad;

2643         struct a {
2644                 long    fdes;
2645                 long    ctl;            /* struct strbuf * */
2646                 long    data;           /* struct strbuf * */
2647                 long    pri;
```

```
2648         } *uap = (struct a *)clwp->lwp_ap;

2650         fd  = (uint32_t)uap->fdes;
2651         pri = (uint32_t)uap->pri;

2653         au_uwrite(au_to_arg32(1, "fd", fd));

2655         if ((fp = getf(fd)) != NULL) {
2656                 fad = F2A(fp);

2658                 fad->fad_flags |= FAD_WRITE;

2660                 /* add path name to audit record */
2661                 if (fad->fad_aupath != NULL) {
2662                         au_uwrite(au_to_path(fad->fad_aupath));
2663                 }
2664                 audit_attributes(fp->f_vnode);

2666                 releasef(fd);
2667         }

2669         au_uwrite(au_to_arg32(4, "pri", pri));
2670 }

2672 /*ARGSUSED*/
2673 static void
2674 aus_putpmsg(struct t_audit_data *tad)
2675 {
2676         klwp_t *clwp = ttolwp(curthread);
2677         uint32_t fd, pri, flags;
2678         struct file *fp;
2679         struct f_audit_data *fad;

2681         struct a {
2682                 long    fdes;
2683                 long    ctl;            /* struct strbuf * */
2684                 long    data;           /* struct strbuf * */
2685                 long    pri;
2686                 long    flags;
2687         } *uap = (struct a *)clwp->lwp_ap;

2689         fd = (uint32_t)uap->fdes;
2690         pri  = (uint32_t)uap->pri;
2691         flags  = (uint32_t)uap->flags;

2693         au_uwrite(au_to_arg32(1, "fd", fd));

2695         if ((fp = getf(fd)) != NULL) {
2696                 fad = F2A(fp);

2698                 fad->fad_flags |= FAD_WRITE;

2700                 /* add path name to audit record */
2701                 if (fad->fad_aupath != NULL) {
2702                         au_uwrite(au_to_path(fad->fad_aupath));
2703                 }
2704                 audit_attributes(fp->f_vnode);

2706                 releasef(fd);
2707         }


2710         au_uwrite(au_to_arg32(4, "pri", pri));
2711         au_uwrite(au_to_arg32(5, "flags", flags));
2712 }
```



```
2714 /*ARGSUSED*/
2715 static void
2716 aus_getmsg(struct t_audit_data *tad)
2717 {
2718         klwp_t *clwp = ttolwp(curthread);
2719         uint32_t fd, pri;
2720         struct file *fp;
2721         struct f_audit_data *fad;

2723         struct a {
2724                 long    fdes;
2725                 long    ctl;            /* struct strbuf * */
2726                 long    data;           /* struct strbuf * */
2727                 long    pri;
2728         } *uap = (struct a *)clwp->lwp_ap;

2730         fd  = (uint32_t)uap->fdes;
2731         pri = (uint32_t)uap->pri;

2733         au_uwrite(au_to_arg32(1, "fd", fd));

2735         if ((fp = getf(fd)) != NULL) {
2736                 fad = F2A(fp);

2738                 /*
2739                  * read operation on this object
2740                  */
2741                 fad->fad_flags |= FAD_READ;

2743                 /* add path name to audit record */
2744                 if (fad->fad_aupath != NULL) {
2745                         au_uwrite(au_to_path(fad->fad_aupath));
2746                 }
2747                 audit_attributes(fp->f_vnode);

2749                 releasef(fd);
2750         }

2752         au_uwrite(au_to_arg32(4, "pri", pri));
2753 }

2755 /*ARGSUSED*/
2756 static void
2757 aus_getpmsg(struct t_audit_data *tad)
2758 {
2759         klwp_t *clwp = ttolwp(curthread);
2760         uint32_t fd;
2761         struct file *fp;
2762         struct f_audit_data *fad;

2764         struct a {
2765                 long    fdes;
2766                 long    ctl;            /* struct strbuf * */
2767                 long    data;           /* struct strbuf * */
2768                 long    pri;
2769                 long    flags;
2770         } *uap = (struct a *)clwp->lwp_ap;

2772         fd = (uint32_t)uap->fdes;

2774         au_uwrite(au_to_arg32(1, "fd", fd));

2776         if ((fp = getf(fd)) != NULL) {
2777                 fad = F2A(fp);

2779                 /*
```

```
2780                 * read operation on this object
2781                 */
2782                fad->fad_flags |= FAD_READ;

2784                /* add path name to audit record */
2785                if (fad->fad_aupath != NULL) {
2786                        au_uwrite(au_to_path(fad->fad_aupath));
2787                }
2788                audit_attributes(fp->f_vnode);

2790                releasef(fd);
2791        }
2792 }

2794 static au_event_t
2795 aui_labelsys(au_event_t e)
2796 {
2797        klwp_t *clwp = ttolwp(curthread);
2798        uint32_t code;
2799        uint32_t cmd;

2801        struct a {
2802                long    code;
2803                long    cmd;
2804        } *uap = (struct a *)clwp->lwp_ap;

2806        code = (uint32_t)uap->code;
2807        cmd = (uint32_t)uap->cmd;

2809        /* not security relevant if not changing kernel cache */
2810        if (cmd == TNDB_GET)
2811                return (AUE_NULL);

2813        switch (code) {
2814        case TSOL_TNRH:
2815                e = AUE_LABELSYS_TNRH;
2816                break;
2817        case TSOL_TNRHTP:
2818                e = AUE_LABELSYS_TNRHTP;
2819                break;
2820        case TSOL_TNMLP:
2821                e = AUE_LABELSYS_TNMLP;
2822                break;
2823        default:
2824                e = AUE_NULL;
2825                break;
2826        }

2828        return (e);

2830 }

2832 static void
2833 aus_labelsys(struct t_audit_data *tad)
2834 {
2835        klwp_t *clwp = ttolwp(curthread);
2836        uint32_t cmd;
2837        uintptr_t a2;

2839        struct a {
2840                long    code;
2841                long    cmd;
2842                long    a2;
2843        } *uap = (struct a *)clwp->lwp_ap;

2845        cmd = (uint32_t)uap->cmd;
```



```
2846        a2 = (uintptr_t)uap->a2;

2848        switch (tad->tad_event) {
2849        case AUE_LABELSYS_TNRH:
2850        {
2851                tsol_rhent_t    *rhent;
2852                tnaddr_t        *rh_addr;

2854                au_uwrite(au_to_arg32(1, "cmd", cmd));

2856                /* Remaining args don't apply for FLUSH, so skip */
2857                if (cmd == TNDB_FLUSH)
2858                        break;

2860                rhent = kmem_alloc(sizeof (tsol_rhent_t), KM_SLEEP);
2861                if (copyin((caddr_t)a2, rhent, sizeof (tsol_rhent_t))) {
2862                        kmem_free(rhent, sizeof (tsol_rhent_t));
2863                        return;
2864                }

2866                rh_addr = &rhent->rh_address;
2867                if (rh_addr->ta_family == AF_INET) {
2868                        struct in_addr  *ipaddr;

2870                        ipaddr = &(rh_addr->ta_addr_v4);
2871                        au_uwrite(au_to_in_addr(ipaddr));
2872                } else if (rh_addr->ta_family == AF_INET6) {
2873                        int32_t         *ipaddr;

2875                        ipaddr = (int32_t *)&(rh_addr->ta_addr_v6);
2876                        au_uwrite(au_to_in_addr_ex(ipaddr));
2877                }
2878                au_uwrite(au_to_arg32(2, "prefix len", rhent->rh_prefix));

2880                kmem_free(rhent, sizeof (tsol_rhent_t));

2882                break;
2883        }
2884        case AUE_LABELSYS_TNRHTP:
2885        {
2886                tsol_tpent_t    *tpent;

2888                au_uwrite(au_to_arg32(1, "cmd", cmd));

2890                /* Remaining args don't apply for FLUSH, so skip */
2891                if (cmd == TNDB_FLUSH)
2892                        break;

2894                tpent = kmem_alloc(sizeof (tsol_tpent_t), KM_SLEEP);
2895                if (copyin((caddr_t)a2, tpent, sizeof (tsol_tpent_t))) {
2896                        kmem_free(tpent, sizeof (tsol_tpent_t));
2897                        return;
2898                }

2900                /* Make sure that the template name is null-terminated. */
2901                *(tpent->name + TNTNAMSIZ - 1) = '\0';

2903                au_uwrite(au_to_text(tpent->name));
2904                kmem_free(tpent, sizeof (tsol_tpent_t));

2906                break;
2907        }
2908        case AUE_LABELSYS_TNMLP:
2909        {
2910                tsol_mlpent_t   *mlpent;
```

```
2912                 au_uwrite(au_to_arg32(1, "cmd", cmd));

2914                 mlpent = kmem_alloc(sizeof (tsol_mlpent_t), KM_SLEEP);
2915                 if (copyin((caddr_t)a2, mlpent, sizeof (tsol_mlpent_t))) {
2916                         kmem_free(mlpent, sizeof (tsol_mlpent_t));
2917                         return;
2918                 }

2920                 if (mlpent->tsme_flags & TSOL_MEF_SHARED) {
2921                         au_uwrite(au_to_text("shared"));
2922                 } else {
2923                         zone_t  *zone;

2925                         zone = zone_find_by_id(mlpent->tsme_zoneid);
2926                         if (zone != NULL) {
2927                                 au_uwrite(au_to_text(zone->zone_name));
2928                                 zone_rele(zone);
2929                         }
2930                 }

2932                 /* Remaining args don't apply for FLUSH, so skip */
2933                 if (cmd == TNDB_FLUSH) {
2934                         kmem_free(mlpent, sizeof (tsol_mlpent_t));
2935                         break;
2936                 }

2938                 au_uwrite(au_to_arg32(2, "proto num",
2939                     (uint32_t)mlpent->tsme_mlp.mlp_ipp));
2940                 au_uwrite(au_to_arg32(2, "mlp_port",
2941                     (uint32_t)mlpent->tsme_mlp.mlp_port));

2943                 if (mlpent->tsme_mlp.mlp_port_upper != 0)
2944                         au_uwrite(au_to_arg32(2, "mlp_port_upper",
2945                             (uint32_t)mlpent->tsme_mlp.mlp_port_upper));

2947                 kmem_free(mlpent, sizeof (tsol_mlpent_t));

2949                 break;
2950         }
2951         default:
2952                 break;
2953         }
2954 }


2957 static au_event_t
2958 aui_auditsys(au_event_t e)
2959 {
2960         klwp_t *clwp = ttolwp(curthread);
2961         uint32_t code;

2963         struct a {
2964                 long    code;
2965                 long    a1;
2966                 long    a2;
2967                 long    a3;
2968                 long    a4;
2969                 long    a5;
2970                 long    a6;
2971                 long    a7;
2972         } *uap = (struct a *)clwp->lwp_ap;

2974         code = (uint32_t)uap->code;

2976         switch (code) {
```



```
2978         case BSM_GETAUID:
2979                 e = AUE_GETAUID;
2980                 break;
2981         case BSM_SETAUID:
2982                 e = AUE_SETAUID;
2983                 break;
2984         case BSM_GETAUDIT:
2985                 e = AUE_GETAUDIT;
2986                 break;
2987         case BSM_GETAUDIT_ADDR:
2988                 e = AUE_GETAUDIT_ADDR;
2989                 break;
2990         case BSM_SETAUDIT:
2991                 e = AUE_SETAUDIT;
2992                 break;
2993         case BSM_SETAUDIT_ADDR:
2994                 e = AUE_SETAUDIT_ADDR;
2995                 break;
2996         case BSM_AUDIT:
2997                 e = AUE_AUDIT;
2998                 break;
2999         case BSM_AUDITCTL:
3000                 switch ((uint_t)uap->a1) {

3002                 case A_GETPOLICY:
3003                         e = AUE_AUDITON_GPOLICY;
3004                         break;
3005                 case A_SETPOLICY:
3006                         e = AUE_AUDITON_SPOLICY;
3007                         break;
3008                 case A_GETAMASK:
3009                         e = AUE_AUDITON_GETAMASK;
3010                         break;
3011                 case A_SETAMASK:
3012                         e = AUE_AUDITON_SETAMASK;
3013                         break;
3014                 case A_GETKMASK:
3015                         e = AUE_AUDITON_GETKMASK;
3016                         break;
3017                 case A_SETKMASK:
3018                         e = AUE_AUDITON_SETKMASK;
3019                         break;
3020                 case A_GETQCTRL:
3021                         e = AUE_AUDITON_GQCTRL;
3022                         break;
3023                 case A_SETQCTRL:
3024                         e = AUE_AUDITON_SQCTRL;
3025                         break;
3026                 case A_GETCWD:
3027                         e = AUE_AUDITON_GETCWD;
3028                         break;
3029                 case A_GETCAR:
3030                         e = AUE_AUDITON_GETCAR;
3031                         break;
3032                 case A_GETSTAT:
3033                         e = AUE_AUDITON_GETSTAT;
3034                         break;
3035                 case A_SETSTAT:
3036                         e = AUE_AUDITON_SETSTAT;
3037                         break;
3038                 case A_SETUMASK:
3039                         e = AUE_AUDITON_SETUMASK;
3040                         break;
3041                 case A_SETSMASK:
3042                         e = AUE_AUDITON_SETSMASK;
3043                         break;
```

```
3044                 case A_GETCOND:
3045                         e = AUE_AUDITON_GETCOND;
3046                         break;
3047                 case A_SETCOND:
3048                         e = AUE_AUDITON_SETCOND;
3049                         break;
3050                 case A_GETCLASS:
3051                         e = AUE_AUDITON_GETCLASS;
3052                         break;
3053                 case A_SETCLASS:
3054                         e = AUE_AUDITON_SETCLASS;
3055                         break;
3056                 default:
3057                         e = AUE_NULL;
3058                         break;
3059                 }
3060                 break;
3061         default:
3062                 e = AUE_NULL;
3063                 break;
3064         }

3066         return (e);

3068 }       /* AUI_AUDITSYS */


3071 static void
3072 aus_auditsys(struct t_audit_data *tad)
3073 {
3074         klwp_t *clwp = ttolwp(curthread);
3075         uintptr_t a1, a2;
3076         STRUCT_DECL(auditinfo, ainfo);
3077         STRUCT_DECL(auditinfo_addr, ainfo_addr);
3078         au_evclass_map_t event;
3079         au_mask_t mask;
3080         int auditstate, policy;
3081         au_id_t auid;


3084         struct a {
3085                 long    code;
3086                 long    a1;
3087                 long    a2;
3088                 long    a3;
3089                 long    a4;
3090                 long    a5;
3091                 long    a6;
3092                 long    a7;
3093         } *uap = (struct a *)clwp->lwp_ap;

3095         a1   = (uintptr_t)uap->a1;
3096         a2   = (uintptr_t)uap->a2;

3098         switch (tad->tad_event) {
3099         case AUE_SETAUID:
3100                 if (copyin((caddr_t)a1, &auid, sizeof (au_id_t)))
3101                                 return;
3102                 au_uwrite(au_to_arg32(2, "setauid", auid));
3103                 break;
3104         case AUE_SETAUDIT:
3105                 STRUCT_INIT(ainfo, get_udatamodel());
3106                 if (copyin((caddr_t)a1, STRUCT_BUF(ainfo),
3107                     STRUCT_SIZE(ainfo))) {
3108                                 return;
3109                 }
```



```
3110                 au_uwrite(au_to_arg32((char)1, "setaudit:auid",
3111                     (uint32_t)STRUCT_FGET(ainfo, ai_auid)));
3112 #ifdef _LP64
3113                 au_uwrite(au_to_arg64((char)1, "setaudit:port",
3114                     (uint64_t)STRUCT_FGET(ainfo, ai_termid.port)));
3115 #else
3116                 au_uwrite(au_to_arg32((char)1, "setaudit:port",
3117                     (uint32_t)STRUCT_FGET(ainfo, ai_termid.port)));
3118 #endif
3119                 au_uwrite(au_to_arg32((char)1, "setaudit:machine",
3120                     (uint32_t)STRUCT_FGET(ainfo, ai_termid.machine)));
3121                 au_uwrite(au_to_arg32((char)1, "setaudit:as_success",
3122                     (uint32_t)STRUCT_FGET(ainfo, ai_mask.as_success)));
3123                 au_uwrite(au_to_arg32((char)1, "setaudit:as_failure",
3124                     (uint32_t)STRUCT_FGET(ainfo, ai_mask.as_failure)));
3125                 au_uwrite(au_to_arg32((char)1, "setaudit:asid",
3126                     (uint32_t)STRUCT_FGET(ainfo, ai_asid)));
3127                 break;
3128         case AUE_SETAUDIT_ADDR:
3129                 STRUCT_INIT(ainfo_addr, get_udatamodel());
3130                 if (copyin((caddr_t)a1, STRUCT_BUF(ainfo_addr),
3131                     STRUCT_SIZE(ainfo_addr))) {
3132                                 return;
3133                 }
3134                 au_uwrite(au_to_arg32((char)1, "auid",
3135                     (uint32_t)STRUCT_FGET(ainfo_addr, ai_auid)));
3136 #ifdef _LP64
3137                 au_uwrite(au_to_arg64((char)1, "port",
3138                     (uint64_t)STRUCT_FGET(ainfo_addr, ai_termid.at_port)));
3139 #else
3140                 au_uwrite(au_to_arg32((char)1, "port",
3141                     (uint32_t)STRUCT_FGET(ainfo_addr, ai_termid.at_port)));
3142 #endif
3143                 au_uwrite(au_to_arg32((char)1, "type",
3144                     (uint32_t)STRUCT_FGET(ainfo_addr, ai_termid.at_type)));
3145                 if ((uint32_t)STRUCT_FGET(ainfo_addr, ai_termid.at_type) ==
3146                     AU_IPv4) {
3147                         au_uwrite(au_to_in_addr(
3148                             (struct in_addr *)STRUCT_FGETP(ainfo_addr,
3149                             ai_termid.at_addr)));
3150                 } else {
3151                         au_uwrite(au_to_in_addr_ex(
3152                             (int32_t *)STRUCT_FGETP(ainfo_addr,
3153                             ai_termid.at_addr)));
3154                 }
3155                 au_uwrite(au_to_arg32((char)1, "as_success",
3156                     (uint32_t)STRUCT_FGET(ainfo_addr, ai_mask.as_success)));
3157                 au_uwrite(au_to_arg32((char)1, "as_failure",
3158                     (uint32_t)STRUCT_FGET(ainfo_addr, ai_mask.as_failure)));
3159                 au_uwrite(au_to_arg32((char)1, "asid",
3160                     (uint32_t)STRUCT_FGET(ainfo_addr, ai_asid)));
3161                 break;
3162         case AUE_AUDITON_SETAMASK:
3163                 if (copyin((caddr_t)a2, &mask, sizeof (au_mask_t)))
3164                                 return;
3165                 au_uwrite(au_to_arg32(
3166                     2, "setamask:as_success", (uint32_t)mask.as_success));
3167                 au_uwrite(au_to_arg32(
3168                     2, "setamask:as_failure", (uint32_t)mask.as_failure));
3169                 break;
3170         case AUE_AUDITON_SETKMASK:
3171                 if (copyin((caddr_t)a2, &mask, sizeof (au_mask_t)))
3172                                 return;
3173                 au_uwrite(au_to_arg32(
3174                     2, "setkmask:as_success", (uint32_t)mask.as_success));
3175                 au_uwrite(au_to_arg32(
```

```
3176                    2, "setkmask:as_failure", (uint32_t)mask.as_failure));
3177                break;
3178        case AUE_AUDITON_SPOLICY:
3179                if (copyin((caddr_t)a2, &policy, sizeof (int)))
3180                        return;
3181                au_uwrite(au_to_arg32(3, "setpolicy", (uint32_t)policy));
3182                break;
3183        case AUE_AUDITON_SQCTRL: {
3184                STRUCT_DECL(au_qctrl, qctrl);
3185                model_t model;

3187                model = get_udatamodel();
3188                STRUCT_INIT(qctrl, model);
3189                if (copyin((caddr_t)a2, STRUCT_BUF(qctrl), STRUCT_SIZE(qctrl)))
3190                        return;
3191                if (model == DATAMODEL_ILP32) {
3192                        au_uwrite(au_to_arg32(
3193                            3, "setqctrl:aq_hiwater",
3194                            (uint32_t)STRUCT_FGET(qctrl, aq_hiwater)));
3195                        au_uwrite(au_to_arg32(
3196                            3, "setqctrl:aq_lowater",
3197                            (uint32_t)STRUCT_FGET(qctrl, aq_lowater)));
3198                        au_uwrite(au_to_arg32(
3199                            3, "setqctrl:aq_bufsz",
3200                            (uint32_t)STRUCT_FGET(qctrl, aq_bufsz)));
3201                        au_uwrite(au_to_arg32(
3202                            3, "setqctrl:aq_delay",
3203                            (uint32_t)STRUCT_FGET(qctrl, aq_delay)));
3204                } else {
3205                        au_uwrite(au_to_arg64(
3206                            3, "setqctrl:aq_hiwater",
3207                            (uint64_t)STRUCT_FGET(qctrl, aq_hiwater)));
3208                        au_uwrite(au_to_arg64(
3209                            3, "setqctrl:aq_lowater",
3210                            (uint64_t)STRUCT_FGET(qctrl, aq_lowater)));
3211                        au_uwrite(au_to_arg64(
3212                            3, "setqctrl:aq_bufsz",
3213                            (uint64_t)STRUCT_FGET(qctrl, aq_bufsz)));
3214                        au_uwrite(au_to_arg64(
3215                            3, "setqctrl:aq_delay",
3216                            (uint64_t)STRUCT_FGET(qctrl, aq_delay)));
3217                }
3218                break;
3219        }
3220        case AUE_AUDITON_SETUMASK:
3221                STRUCT_INIT(ainfo, get_udatamodel());
3222                if (copyin((caddr_t)uap->a2, STRUCT_BUF(ainfo),
3223                    STRUCT_SIZE(ainfo))) {
3224                        return;
3225                }
3226                au_uwrite(au_to_arg32(3, "setumask:as_success",
3227                    (uint32_t)STRUCT_FGET(ainfo, ai_mask.as_success)));
3228                au_uwrite(au_to_arg32(3, "setumask:as_failure",
3229                    (uint32_t)STRUCT_FGET(ainfo, ai_mask.as_failure)));
3230                break;
3231        case AUE_AUDITON_SETSMASK:
3232                STRUCT_INIT(ainfo, get_udatamodel());
3233                if (copyin((caddr_t)uap->a2, STRUCT_BUF(ainfo),
3234                    STRUCT_SIZE(ainfo))) {
3235                        return;
3236                }
3237                au_uwrite(au_to_arg32(3, "setsmask:as_success",
3238                    (uint32_t)STRUCT_FGET(ainfo, ai_mask.as_success)));
3239                au_uwrite(au_to_arg32(3, "setsmask:as_failure",
3240                    (uint32_t)STRUCT_FGET(ainfo, ai_mask.as_failure)));
3241                break;
```



```
3242        case AUE_AUDITON_SETCOND:
3243                if (copyin((caddr_t)a2, &auditstate, sizeof (int)))
3244                        return;
3245                au_uwrite(au_to_arg32(3, "setcond", (uint32_t)auditstate));
3246                break;
3247        case AUE_AUDITON_SETCLASS:
3248                if (copyin((caddr_t)a2, &event, sizeof (au_evclass_map_t)))
3249                        return;
3250                au_uwrite(au_to_arg32(
3251                    2, "setclass:ec_event", (uint32_t)event.ec_number));
3252                au_uwrite(au_to_arg32(
3253                    3, "setclass:ec_class", (uint32_t)event.ec_class));
3254                break;
3255        case AUE_GETAUID:
3256        case AUE_GETAUDIT:
3257        case AUE_GETAUDIT_ADDR:
3258        case AUE_AUDIT:
3259        case AUE_AUDITON_GPOLICY:
3260        case AUE_AUDITON_GQCTRL:
3261        case AUE_AUDITON_GETAMASK:
3262        case AUE_AUDITON_GETKMASK:
3263        case AUE_AUDITON_GETCWD:
3264        case AUE_AUDITON_GETCAR:
3265        case AUE_AUDITON_GETSTAT:
3266        case AUE_AUDITON_SETSTAT:
3267        case AUE_AUDITON_GETCOND:
3268        case AUE_AUDITON_GETCLASS:
3269                break;
3270        default:
3271                break;
3272        }

3274 }      /* AUS_AUDITSYS */


3277 /* only audit privileged operations for systeminfo(2) system call */
3278 static au_event_t
3279 aui_sysinfo(au_event_t e)
3280 {
3281        klwp_t *clwp = ttolwp(curthread);
3282        uint32_t command;

3284        struct a {
3285                long    command;
3286                long    buf;            /* char * */
3287                long    count;
3288        } *uap = (struct a *)clwp->lwp_ap;

3290        command = (uint32_t)uap->command;

3292        switch (command) {
3293        case SI_SET_HOSTNAME:
3294        case SI_SET_SRPC_DOMAIN:
3295                e = (au_event_t)AUE_SYSINFO;
3296                break;
3297        default:
3298                e = (au_event_t)AUE_NULL;
3299                break;
3300        }
3301        return (e);
3302 }

3304 /*ARGSUSED*/
3305 static void
3306 aus_sysinfo(struct t_audit_data *tad)
3307 {
```

```
3308         klwp_t *clwp = ttolwp(curthread);
3309         uint32_t command;
3310         size_t len, maxlen;
3311         char *name;
3312         uintptr_t buf;

3314         struct a {
3315                 long    command;
3316                 long    buf;            /* char * */
3317                 long    count;
3318         } *uap = (struct a *)clwp->lwp_ap;

3320         command = (uint32_t)uap->command;
3321         buf = (uintptr_t)uap->buf;

3323         au_uwrite(au_to_arg32(1, "cmd", command));

3325         switch (command) {
3326         case SI_SET_HOSTNAME:
3327         {
3328                 if (secpolicy_sys_config(CRED(), B_TRUE) != 0)
3329                         return;

3331                 maxlen = SYS_NMLN;
3332                 name = kmem_alloc(maxlen, KM_SLEEP);
3333                 if (copyinstr((caddr_t)buf, name, SYS_NMLN, &len))
3334                         break;

3336                 /*
3337                  * Must be non-NULL string and string
3338                  * must be less than SYS_NMLN chars.
3339                  */
3340                 if (len < 2 || (len == SYS_NMLN && name[SYS_NMLN - 1] != '\0'))
3341                         break;

3343                 au_uwrite(au_to_text(name));
3344                 break;
3345         }

3347         case SI_SET_SRPC_DOMAIN:
3348         {
3349                 if (secpolicy_sys_config(CRED(), B_TRUE) != 0)
3350                         return;

3352                 maxlen = SYS_NMLN;
3353                 name = kmem_alloc(maxlen, KM_SLEEP);
3354                 if (copyinstr((caddr_t)buf, name, SYS_NMLN, &len))
3355                         break;

3357                 /*
3358                  * If string passed in is longer than length
3359                  * allowed for domain name, fail.
3360                  */
3361                 if (len == SYS_NMLN && name[SYS_NMLN - 1] != '\0')
3362                         break;

3364                 au_uwrite(au_to_text(name));
3365                 break;
3366         }

3368         default:
3369                 return;
3370         }

3372         kmem_free(name, maxlen);
3373 }
```



```
3375 static au_event_t
3376 aui_modctl(au_event_t e)
3377 {
3378         klwp_t *clwp = ttolwp(curthread);
3379         uint_t cmd;

3381         struct a {
3382                 long    cmd;
3383         } *uap = (struct a *)clwp->lwp_ap;

3385         cmd = (uint_t)uap->cmd;

3387         switch (cmd) {
3388         case MODLOAD:
3389                 e = AUE_MODLOAD;
3390                 break;
3391         case MODUNLOAD:
3392                 e = AUE_MODUNLOAD;
3393                 break;
3394         case MODADDMAJBIND:
3395                 e = AUE_MODADDMAJ;
3396                 break;
3397         case MODSETDEVPOLICY:
3398                 e = AUE_MODDEVPLCY;
3399                 break;
3400         case MODALLOCPRIV:
3401                 e = AUE_MODADDPRIV;
3402                 break;
3403         default:
3404                 e = AUE_NULL;
3405                 break;
3406         }
3407         return (e);
3408 }


3411 /*ARGSUSED*/
3412 static void
3413 aus_modctl(struct t_audit_data *tad)
3414 {
3415         klwp_t *clwp = ttolwp(curthread);
3416         void *a = clwp->lwp_ap;
3417         uint_t use_path;

3419         switch (tad->tad_event) {
3420         case AUE_MODLOAD: {
3421                 typedef struct {
3422                         long    cmd;
3423                         long    use_path;
3424                         long    filename;               /* char * */
3425                 } modloada_t;

3427                 char *filenamep;
3428                 uintptr_t fname;
3429                 extern char *default_path;

3431                 fname = (uintptr_t)((modloada_t *)a)->filename;
3432                 use_path = (uint_t)((modloada_t *)a)->use_path;

3434                         /* space to hold path */
3435                 filenamep = kmem_alloc(MOD_MAXPATH, KM_SLEEP);
3436                         /* get string */
3437                 if (copyinstr((caddr_t)fname, filenamep, MOD_MAXPATH, 0)) {
3438                                 /* free allocated path */
3439                         kmem_free(filenamep, MOD_MAXPATH);
```

```
3440                        return;
3441                }
3442                        /* ensure it's null terminated */
3443                filenamep[MOD_MAXPATH - 1] = 0;

3445                if (use_path)
3446                        au_uwrite(au_to_text(default_path));
3447                au_uwrite(au_to_text(filenamep));

3449                        /* release temporary memory */
3450                kmem_free(filenamep, MOD_MAXPATH);
3451                break;
3452        }
3453        case AUE_MODUNLOAD: {
3454                typedef struct {
3455                        long    cmd;
3456                        long    id;
3457                } modunloada_t;

3459                uint32_t id = (uint32_t)((modunloada_t *)a)->id;

3461                au_uwrite(au_to_arg32(1, "id", id));
3462                break;
3463        }
3464        case AUE_MODADDMAJ: {
3465                STRUCT_DECL(modconfig, mc);
3466                typedef struct {
3467                        long    cmd;
3468                        long    subcmd;
3469                        long    data;           /* int * */
3470                } modconfiga_t;

3472                STRUCT_DECL(aliases, alias);
3473                caddr_t ap;
3474                int i, num_aliases;
3475                char *drvname, *mc_drvname;
3476                char *name;
3477                extern char *ddi_major_to_name(major_t);
3478                model_t model;

3480                uintptr_t data = (uintptr_t)((modconfiga_t *)a)->data;

3482                model = get_udatamodel();
3483                STRUCT_INIT(mc, model);
3484                        /* sanitize buffer */
3485                bzero((caddr_t)STRUCT_BUF(mc), STRUCT_SIZE(mc));
3486                        /* get user arguments */
3487                if (copyin((caddr_t)data, (caddr_t)STRUCT_BUF(mc),
3488                    STRUCT_SIZE(mc)) != 0)
3489                        return;

3491                mc_drvname = STRUCT_FGET(mc, drvname);
3492                if ((drvname = ddi_major_to_name(
3493                    (major_t)STRUCT_FGET(mc, major))) != NULL &&
3494                    strncmp(drvname, mc_drvname, MAXMODCONFNAME) != 0) {
3495                                /* safety */
3496                        if (mc_drvname[0] != '\0') {
3497                                mc_drvname[MAXMODCONFNAME-1] = '\0';
3498                                au_uwrite(au_to_text(mc_drvname));
3499                        }
3500                                /* drvname != NULL from test above */
3501                        au_uwrite(au_to_text(drvname));
3502                        return;
3503                }

3505                if (mc_drvname[0] != '\0') {
```



```
3506                                /* safety */
3507                        mc_drvname[MAXMODCONFNAME-1] = '\0';
3508                        au_uwrite(au_to_text(mc_drvname));
3509                } else
3510                        au_uwrite(au_to_text("no drvname"));

3512                num_aliases = STRUCT_FGET(mc, num_aliases);
3513                au_uwrite(au_to_arg32(5, "", (uint32_t)num_aliases));
3514                ap = (caddr_t)STRUCT_FGETP(mc, ap);
3515                name = kmem_alloc(MAXMODCONFNAME, KM_SLEEP);
3516                STRUCT_INIT(alias, model);
3517                for (i = 0; i < num_aliases; i++) {
3518                        bzero((caddr_t)STRUCT_BUF(alias),
3519                            STRUCT_SIZE(alias));
3520                        if (copyin((caddr_t)ap, (caddr_t)STRUCT_BUF(alias),
3521                            STRUCT_SIZE(alias)) != 0)
3522                                break;
3523                        if (copyinstr(STRUCT_FGETP(alias, a_name), name,
3524                            MAXMODCONFNAME, NULL) != 0) {
3525                                break;
3526                        }

3528                        au_uwrite(au_to_text(name));
3529                        ap = (caddr_t)STRUCT_FGETP(alias, a_next);
3530                }
3531                kmem_free(name, MAXMODCONFNAME);
3532                break;
3533        }
3534        default:
3535                break;
3536        }
3537 }


3540 /*ARGSUSED*/
3541 static void
3542 auf_accept(
3543        struct t_audit_data *tad,
3544        int     error,
3545        rval_t  *rval)
3546 {
3547        uint32_t scid;
3548        uint32_t sy_flags;
3549        int fd;
3550        struct sonode *so;
3551        char so_laddr[sizeof (struct sockaddr_in6)];
3552        char so_faddr[sizeof (struct sockaddr_in6)];
3553        int err;
3554        short so_family, so_type;
3555        int add_sock_token = 0;

3557        /* need to determine type of executing binary */
3558        scid = tad->tad_scid;
3559 #ifdef _SYSCALL32_IMPL
3560        if (lwp_getdatamodel(ttolwp(curthread)) == DATAMODEL_NATIVE)
3561                sy_flags = sysent[scid].sy_flags & SE_RVAL_MASK;
3562        else
3563                sy_flags = sysent32[scid].sy_flags & SE_RVAL_MASK;
3564 #else
3565        sy_flags = sysent[scid].sy_flags & SE_RVAL_MASK;
3566 #endif
3567        switch (sy_flags) {
3568        case SE_32RVAL1:
3569                /* FALLTHRU */
3570        case SE_32RVAL2|SE_32RVAL1:
3571                fd = rval->r_val1;
```

```
3572                break;
3573        case SE_64RVAL:
3574                fd = (int)rval->r_vals;
3575                break;
3576        default:
3577                /*
3578                 * should never happen, seems to be an internal error
3579                 * in sysent => no fd, nothing to audit here, returning
3580                 */
3581                return;
3582        }

3584        if (error) {
3585                /* can't trust socket contents. Just return */
3586                au_uwrite(au_to_arg32(1, "so", (uint32_t)fd));
3587                return;
3588        }

3590        if ((so = getsonode(fd, &err, NULL)) == NULL) {
3591                /*
3592                 * not security relevant if doing a accept from non socket
3593                 * so no extra tokens. Should probably turn off audit record
3594                 * generation here.
3595                 */
3596                return;
3597        }

3599        so_family = so->so_family;
3600        so_type   = so->so_type;

3602        switch (so_family) {
3603        case AF_INET:
3604        case AF_INET6:
3605                /*
3606                 * XXX - what about other socket types for AF_INET (e.g. DGRAM)
3607                 */
3608                if (so->so_type == SOCK_STREAM) {
3609                        socklen_t len;

3611                        bzero((void *)so_laddr, sizeof (so_laddr));
3612                        bzero((void *)so_faddr, sizeof (so_faddr));

3614                        len = sizeof (so_laddr);
3615                        (void) socket_getsockname(so,
3616                            (struct sockaddr *)so_laddr, &len, CRED());
3617                        len = sizeof (so_faddr);
3618                        (void) socket_getpeername(so,
3619                            (struct sockaddr *)so_faddr, &len, B_FALSE, CRED());

3621                        add_sock_token = 1;
3622                }
3623                break;

3625        default:
3626                /* AF_UNIX, AF_ROUTE, AF_KEY do not support accept */
3627                break;
3628        }

3630        releasef(fd);

3632        au_uwrite(au_to_arg32(1, "so", (uint32_t)fd));

3634        if (add_sock_token == 0) {
3635                au_uwrite(au_to_arg32(0, "family", (uint32_t)(so_family)));
3636                au_uwrite(au_to_arg32(0, "type", (uint32_t)(so_type)));
3637                return;
```



```
3638        }

3640        au_uwrite(au_to_socket_ex(so_family, so_type, so_laddr, so_faddr));

3642 }

3644 /*ARGSUSED*/
3645 static void
3646 auf_bind(struct t_audit_data *tad, int error, rval_t *rvp)
3647 {
3648        struct a {
3649                long    fd;
3650                long    addr;
3651                long    len;
3652        } *uap = (struct a *)ttolwp(curthread)->lwp_ap;

3654        struct sonode *so;
3655        char so_laddr[sizeof (struct sockaddr_in6)];
3656        char so_faddr[sizeof (struct sockaddr_in6)];
3657        int err, fd;
3658        socklen_t len;
3659        short so_family, so_type;
3660        int add_sock_token = 0;

3662        fd = (int)uap->fd;

3664        /*
3665         * bind failed, then nothing extra to add to audit record.
3666         */
3667        if (error) {
3668                au_uwrite(au_to_arg32(1, "so", (uint32_t)fd));
3669                /* XXX may want to add failed address some day */
3670                return;
3671        }

3673        if ((so = getsonode(fd, &err, NULL)) == NULL) {
3674                /*
3675                 * not security relevant if doing a bind from non socket
3676                 * so no extra tokens. Should probably turn off audit record
3677                 * generation here.
3678                 */
3679                return;
3680        }

3682        so_family = so->so_family;
3683        so_type   = so->so_type;

3685        switch (so_family) {
3686        case AF_INET:
3687        case AF_INET6:

3689                bzero(so_faddr, sizeof (so_faddr));
3690                len = sizeof (so_faddr);

3692                (void) socket_getpeername(so,
3693                    (struct sockaddr *)so_faddr, &len, B_FALSE, CRED());
3694                add_sock_token = 1;

3696                break;

3698        case AF_UNIX:
3699                /* token added by lookup */
3700                break;
3701        default:
3702                /* AF_ROUTE, AF_KEY do not support accept */
3703                break;
```

```
3704         }

3706         releasef(fd);

3708         au_uwrite(au_to_arg32(1, "so", (uint32_t)fd));

3710         if (add_sock_token == 0) {
3711                 au_uwrite(au_to_arg32(1, "family", (uint32_t)(so_family)));
3712                 au_uwrite(au_to_arg32(1, "type", (uint32_t)(so_type)));
3713                 return;
3714         }

3716         au_uwrite(au_to_socket_ex(so_family, so_type, so_laddr, so_faddr));

3718 }

3720 /*ARGSUSED*/
3721 static void
3722 auf_connect(struct t_audit_data *tad, int error, rval_t *rval)
3723 {
3724         struct a {
3725                 long    fd;
3726                 long    addr;
3727                 long    len;
3728         } *uap = (struct a *)ttolwp(curthread)->lwp_ap;

3730         struct sonode *so;
3731         char so_laddr[sizeof (struct sockaddr_in6)];
3732         char so_faddr[sizeof (struct sockaddr_in6)];
3733         int err, fd;
3734         socklen_t len;
3735         short so_family, so_type;
3736         int add_sock_token = 0;

3738         fd = (int)uap->fd;


3741         if ((so = getsonode(fd, &err, NULL)) == NULL) {
3742                 /*
3743                  * not security relevant if doing a connect from non socket
3744                  * so no extra tokens. Should probably turn off audit record
3745                  * generation here.
3746                  */
3747                 return;
3748         }

3750         so_family = so->so_family;
3751         so_type   = so->so_type;

3753         switch (so_family) {
3754         case AF_INET:
3755         case AF_INET6:

3757                 bzero(so_laddr, sizeof (so_laddr));
3758                 bzero(so_faddr, sizeof (so_faddr));

3760                 len = sizeof (so_laddr);
3761                 (void) socket_getsockname(so, (struct sockaddr *)so_laddr,
3762                     &len, CRED());
3763                 if (error) {
3764                         if (uap->addr == NULL)
3765                                 break;
3766                         if (uap->len <= 0)
3767                                 break;
3768                         len = min(uap->len, sizeof (so_faddr));
3769                         if (copyin((caddr_t)(uap->addr), so_faddr, len) != 0)
```



```
3770                                 break;
3771 #ifdef NOTYET
3772                         au_uwrite(au_to_data(AUP_HEX, AUR_CHAR, len, so_faddr));
3773 #endif
3774                 } else {
3775                         /* sanity check on length */
3776                         len = sizeof (so_faddr);
3777                         (void) socket_getpeername(so,
3778                             (struct sockaddr *)so_faddr, &len, B_FALSE, CRED());
3779                 }

3781                 add_sock_token = 1;

3783                 break;

3785         case AF_UNIX:
3786                 /* does a lookup on name */
3787                 break;

3789         default:
3790                 /* AF_ROUTE, AF_KEY do not support accept */
3791                 break;
3792         }

3794         releasef(fd);

3796         au_uwrite(au_to_arg32(1, "so", (uint32_t)fd));

3798         if (add_sock_token == 0) {
3799                 au_uwrite(au_to_arg32(1, "family", (uint32_t)(so_family)));
3800                 au_uwrite(au_to_arg32(1, "type", (uint32_t)(so_type)));
3801                 return;
3802         }

3804         au_uwrite(au_to_socket_ex(so_family, so_type, so_laddr, so_faddr));

3806 }

3808 /*ARGSUSED*/
3809 static void
3810 aus_shutdown(struct t_audit_data *tad)
3811 {
3812         struct a {
3813                 long    fd;
3814                 long    how;
3815         } *uap = (struct a *)ttolwp(curthread)->lwp_ap;

3817         struct sonode *so;
3818         char so_laddr[sizeof (struct sockaddr_in6)];
3819         char so_faddr[sizeof (struct sockaddr_in6)];
3820         int err, fd;
3821         socklen_t len;
3822         short so_family, so_type;
3823         int add_sock_token = 0;
3824         file_t *fp;                             /* unix domain sockets */
3825         struct f_audit_data *fad;               /* unix domain sockets */

3827         fd = (int)uap->fd;

3829         if ((so = getsonode(fd, &err, &fp)) == NULL) {
3830                 /*
3831                  * not security relevant if doing a shutdown using non socket
3832                  * so no extra tokens. Should probably turn off audit record
3833                  * generation here.
3834                  */
3835                 return;
```

```
3836         }

3838         so_family = so->so_family;
3839         so_type   = so->so_type;

3841         switch (so_family) {
3842         case AF_INET:
3843         case AF_INET6:

3845                 bzero(so_laddr, sizeof (so_laddr));
3846                 bzero(so_faddr, sizeof (so_faddr));

3848                 len = sizeof (so_laddr);
3849                 (void) socket_getsockname(so,
3850                     (struct sockaddr *)so_laddr, &len, CRED());
3851                 len = sizeof (so_faddr);
3852                 (void) socket_getpeername(so,
3853                     (struct sockaddr *)so_faddr, &len, B_FALSE, CRED());

3855                 add_sock_token = 1;

3857                 break;

3859         case AF_UNIX:

3861                 /* get path from file struct here */
3862                 fad = F2A(fp);
3863                 ASSERT(fad);

3865                 if (fad->fad_aupath != NULL) {
3866                         au_uwrite(au_to_path(fad->fad_aupath));
3867                 } else {
3868                         au_uwrite(au_to_arg32(1, "no path: fd", fd));
3869                 }

3871                 audit_attributes(fp->f_vnode);

3873                 break;

3875         default:
3876                 /*
3877                  * AF_KEY and AF_ROUTE support shutdown. No socket token
3878                  * added.
3879                  */
3880                 break;
3881         }

3883         releasef(fd);

3885         au_uwrite(au_to_arg32(1, "so", (uint32_t)fd));

3887         if (add_sock_token == 0) {
3888                 au_uwrite(au_to_arg32(1, "family", (uint32_t)(so_family)));
3889                 au_uwrite(au_to_arg32(1, "type", (uint32_t)(so_type)));
3890                 au_uwrite(au_to_arg32(2, "how", (uint32_t)(uap->how)));
3891                 return;
3892         }

3894         au_uwrite(au_to_arg32(2, "how", (uint32_t)(uap->how)));

3896         au_uwrite(au_to_socket_ex(so_family, so_type, so_laddr, so_faddr));

3898 }

3900 /*ARGSUSED*/
3901 static void
```



```
3902 auf_setsockopt(struct t_audit_data *tad, int error, rval_t *rval)
3903 {
3904         struct a {
3905                 long    fd;
3906                 long    level;
3907                 long    optname;
3908                 long    *optval;
3909                 long    optlen;
3910         } *uap = (struct a *)ttolwp(curthread)->lwp_ap;

3912         struct sonode   *so;
3913         char so_laddr[sizeof (struct sockaddr_in6)];
3914         char so_faddr[sizeof (struct sockaddr_in6)];
3915         char            val[AU_BUFSIZE];
3916         int             err, fd;
3917         socklen_t       len;
3918         short so_family, so_type;
3919         int             add_sock_token = 0;
3920         file_t *fp;                             /* unix domain sockets */
3921         struct f_audit_data *fad;               /* unix domain sockets */

3923         fd = (int)uap->fd;

3925         if (error) {
3926                 au_uwrite(au_to_arg32(1, "so", (uint32_t)fd));
3927                 au_uwrite(au_to_arg32(2, "level", (uint32_t)uap->level));
3928                 /* XXX may want to include other arguments */
3929                 return;
3930         }

3932         if ((so = getsonode(fd, &err, &fp)) == NULL) {
3933                 /*
3934                  * not security relevant if doing a setsockopt from non socket
3935                  * so no extra tokens. Should probably turn off audit record
3936                  * generation here.
3937                  */
3938                 return;
3939         }

3941         so_family = so->so_family;
3942         so_type   = so->so_type;

3944         switch (so_family) {
3945         case AF_INET:
3946         case AF_INET6:
3947                 bzero((void *)so_laddr, sizeof (so_laddr));
3948                 bzero((void *)so_faddr, sizeof (so_faddr));

3950                 /* get local and foreign addresses */
3951                 len = sizeof (so_laddr);
3952                 (void) socket_getsockname(so, (struct sockaddr *)so_laddr,
3953                     &len, CRED());
3954                 len = sizeof (so_faddr);
3955                 (void) socket_getpeername(so, (struct sockaddr *)so_faddr,
3956                     &len, B_FALSE, CRED());

3958                 add_sock_token = 1;

3960                 break;

3962         case AF_UNIX:

3964                 /* get path from file struct here */
3965                 fad = F2A(fp);
3966                 ASSERT(fad);
```

```
3968                 if (fad->fad_aupath != NULL) {
3969                         au_uwrite(au_to_path(fad->fad_aupath));
3970                 } else {
3971                         au_uwrite(au_to_arg32(1, "no path: fd", fd));
3972                 }

3974                 audit_attributes(fp->f_vnode);

3976                 break;

3978         default:
3979                 /*
3980                  * AF_KEY and AF_ROUTE support setsockopt. No socket token
3981                  * added.
3982                  */
3983                 break;
3984         }

3986         releasef(fd);

3988         au_uwrite(au_to_arg32(1, "so", (uint32_t)fd));

3990         if (add_sock_token == 0) {
3991                 au_uwrite(au_to_arg32(1, "family", (uint32_t)(so_family)));
3992                 au_uwrite(au_to_arg32(1, "type", (uint32_t)(so_type)));
3993         }
3994         au_uwrite(au_to_arg32(2, "level", (uint32_t)(uap->level)));
3995         au_uwrite(au_to_arg32(3, "optname", (uint32_t)(uap->optname)));

3997         bzero(val, sizeof (val));
3998         len = min(uap->optlen, sizeof (val));
3999         if ((len > 0) &&
4000             (copyin((caddr_t)(uap->optval), (caddr_t)val, len) == 0)) {
4001                 au_uwrite(au_to_arg32(5, "optlen", (uint32_t)(uap->optlen)));
4002                 au_uwrite(au_to_data(AUP_HEX, AUR_BYTE, len, val));
4003         }

4005         if (add_sock_token == 0)
4006                 return;

4008         au_uwrite(au_to_socket_ex(so_family, so_type, so_laddr, so_faddr));

4010 }

4012 /*ARGSUSED*/
4013 static void
4014 aus_sockconfig(tad)
4015         struct t_audit_data *tad;
4016 {
4017         struct a {
4018                 long    cmd;
4019                 long    arg1;
4020                 long    arg2;
4021                 long    arg3;
4022                 long    arg4;
4023         } *uap = (struct a *)ttolwp(curthread)->lwp_ap;

4025         char    *buf;
4026         int     buflen;
4027         size_t  size;

4029         au_uwrite(au_to_arg32(1, "cmd", (uint_t)uap->cmd));
4030         switch (uap->cmd) {
4031         case SOCKCONFIG_ADD_SOCK:
4032         case SOCKCONFIG_REMOVE_SOCK:
4033                 au_uwrite(au_to_arg32(2, "domain", (uint32_t)uap->arg1));
```



```
4034                 au_uwrite(au_to_arg32(3, "type", (uint32_t)uap->arg2));
4035                 au_uwrite(au_to_arg32(4, "protocol", (uint32_t)uap->arg3));

4037                 if (uap->arg4 == 0) {
4038                         au_uwrite(au_to_arg32(5, "devpath", (uint32_t)0));
4039                 } else {
4040                         buflen = MAXPATHLEN + 1;
4041                         buf = kmem_alloc(buflen, KM_SLEEP);
4042                         if (copyinstr((caddr_t)uap->arg4, buf, buflen,
4043                             &size)) {
4044                                 kmem_free(buf, buflen);
4045                                 return;
4046                         }

4048                         if (size > MAXPATHLEN) {
4049                                 kmem_free(buf, buflen);
4050                                 return;
4051                         }

4053                         au_uwrite(au_to_text(buf));
4054                         kmem_free(buf, buflen);
4055                 }
4056                 break;
4057         case SOCKCONFIG_ADD_FILTER:
4058         case SOCKCONFIG_REMOVE_FILTER:
4059                 buflen = FILNAME_MAX;
4060                 buf = kmem_alloc(buflen, KM_SLEEP);

4062                 if (copyinstr((caddr_t)uap->arg1, buf, buflen, &size)) {
4063                         kmem_free(buf, buflen);
4064                         return;
4065                 }

4067                 au_uwrite(au_to_text(buf));
4068                 kmem_free(buf, buflen);
4069                 break;
4070         default:
4071                 break;
4072         }
4073 }

4075 /*
4076  * only audit recvmsg when the system call represents the creation of a new
4077  * circuit. This effectively occurs for all UDP packets and may occur for
4078  * special TCP situations where the local host has not set a local address
4079  * in the socket structure.
4080  */
4081 /*ARGSUSED*/
4082 static void
4083 auf_recvmsg(
4084         struct t_audit_data *tad,
4085         int error,
4086         rval_t *rvp)
4087 {
4088         struct a {
4089                 long    fd;
4090                 long    msg;    /* struct msghdr */
4091                 long    flags;
4092         } *uap = (struct a *)ttolwp(curthread)->lwp_ap;

4094         struct sonode   *so;
4095         STRUCT_DECL(msghdr, msg);
4096         caddr_t msg_name;
4097         socklen_t msg_namelen;
4098         int fd;
4099         int err;
```

```
4100         char so_laddr[sizeof (struct sockaddr_in6)];
4101         char so_faddr[sizeof (struct sockaddr_in6)];
4102         socklen_t len;
4103         file_t *fp;                             /* unix domain sockets */
4104         struct f_audit_data *fad;               /* unix domain sockets */
4105         short so_family, so_type;
4106         int add_sock_token = 0;
4107         au_kcontext_t   *kctx = GET_KCTX_PZ;

4109         fd = (int)uap->fd;

4111         /* bail if an error */
4112         if (error) {
4113                 au_uwrite(au_to_arg32(1, "so", (uint32_t)fd));
4114                 au_uwrite(au_to_arg32(3, "flags", (uint32_t)(uap->flags)));
4115                 return;
4116         }

4118         if ((so = getsonode(fd, &err, &fp)) == NULL) {
4119                 /*
4120                  * not security relevant if doing a recvmsg from non socket
4121                  * so no extra tokens. Should probably turn off audit record
4122                  * generation here.
4123                  */
4124                 return;
4125         }

4127         so_family = so->so_family;
4128         so_type   = so->so_type;

4130         /*
4131          * only putout SOCKET_EX token if INET/INET6 family.
4132          * XXX - what do we do about other families?
4133          */

4135         switch (so_family) {
4136         case AF_INET:
4137         case AF_INET6:

4139                 /*
4140                  * if datagram type socket, then just use what is in
4141                  * socket structure for local address.
4142                  * XXX - what do we do for other types?
4143                  */
4144                 if ((so->so_type == SOCK_DGRAM) ||
4145                     (so->so_type == SOCK_RAW)) {
4146                         add_sock_token = 1;

4148                         bzero((void *)so_laddr, sizeof (so_laddr));
4149                         bzero((void *)so_faddr, sizeof (so_faddr));

4151                         /* get local address */
4152                         len = sizeof (so_laddr);
4153                         (void) socket_getsockname(so,
4154                             (struct sockaddr *)so_laddr, &len, CRED());

4156                         /* get peer address */
4157                         STRUCT_INIT(msg, get_udatamodel());

4159                         if (copyin((caddr_t)(uap->msg),
4160                             (caddr_t)STRUCT_BUF(msg), STRUCT_SIZE(msg)) != 0) {
4161                                 break;
4162                         }
4163                         msg_name = (caddr_t)STRUCT_FGETP(msg, msg_name);
4164                         if (msg_name == NULL) {
4165                                 break;
```



```
4166                         }

4168                         /* length is value from recvmsg - sanity check */
4169                         msg_namelen = (socklen_t)STRUCT_FGET(msg, msg_namelen);
4170                         if (msg_namelen == 0) {
4171                                 break;
4172                         }
4173                         if (copyin(msg_name, so_faddr,
4174                             sizeof (so_faddr)) != 0) {
4175                                 break;
4176                         }

4178                 } else if (so->so_type == SOCK_STREAM) {

4180                         /* get path from file struct here */
4181                         fad = F2A(fp);
4182                         ASSERT(fad);

4184                         /*
4185                          * already processed this file for read attempt
4186                          */
4187                         if (fad->fad_flags & FAD_READ) {
4188                                 /* don't want to audit every recvmsg attempt */
4189                                 tad->tad_flag = 0;
4190                                 /* free any residual audit data */
4191                                 au_close(kctx, &(u_ad), 0, 0, 0, NULL);
4192                                 releasef(fd);
4193                                 return;
4194                         }
4195                         /*
4196                          * mark things so we know what happened and don't
4197                          * repeat things
4198                          */
4199                         fad->fad_flags |= FAD_READ;

4201                         bzero((void *)so_laddr, sizeof (so_laddr));
4202                         bzero((void *)so_faddr, sizeof (so_faddr));

4204                         /* get local and foreign addresses */
4205                         len = sizeof (so_laddr);
4206                         (void) socket_getsockname(so,
4207                             (struct sockaddr *)so_laddr, &len, CRED());
4208                         len = sizeof (so_faddr);
4209                         (void) socket_getpeername(so,
4210                             (struct sockaddr *)so_faddr, &len, B_FALSE, CRED());

4212                         add_sock_token = 1;
4213                 }

4215                 /* XXX - what about SOCK_RDM/SOCK_SEQPACKET ??? */

4217                 break;

4219         case AF_UNIX:
4220                 /*
4221                  * first check if this is first time through. Too much
4222                  * duplicate code to put this in an aui_ routine.
4223                  */

4225                 /* get path from file struct here */
4226                 fad = F2A(fp);
4227                 ASSERT(fad);

4229                 /*
4230                  * already processed this file for read attempt
4231                  */
```

```
4232                if (fad->fad_flags & FAD_READ) {
4233                        releasef(fd);
4234                        /* don't want to audit every recvmsg attempt */
4235                        tad->tad_flag = 0;
4236                        /* free any residual audit data */
4237                        au_close(kctx, &(u_ad), 0, 0, 0, NULL);
4238                        return;
4239                }
4240                /*
4241                 * mark things so we know what happened and don't
4242                 * repeat things
4243                 */
4244                fad->fad_flags |= FAD_READ;

4246                if (fad->fad_aupath != NULL) {
4247                        au_uwrite(au_to_path(fad->fad_aupath));
4248                } else {
4249                        au_uwrite(au_to_arg32(1, "no path: fd", fd));
4250                }

4252                audit_attributes(fp->f_vnode);

4254                releasef(fd);

4256                return;

4258        default:
4259                break;

4261        }

4263        releasef(fd);

4265        au_uwrite(au_to_arg32(1, "so", (uint32_t)fd));

4267        if (add_sock_token == 0) {
4268                au_uwrite(au_to_arg32(1, "family", (uint32_t)so_family));
4269                au_uwrite(au_to_arg32(1, "type", (uint32_t)so_type));
4270                au_uwrite(au_to_arg32(3, "flags", (uint32_t)(uap->flags)));
4271                return;
4272        }

4274        au_uwrite(au_to_arg32(3, "flags", (uint32_t)(uap->flags)));

4276        au_uwrite(au_to_socket_ex(so_family, so_type, so_laddr, so_faddr));

4278 }

4280 /*ARGSUSED*/
4281 static void
4282 auf_recvfrom(
4283        struct t_audit_data *tad,
4284        int error,
4285        rval_t *rvp)
4286 {

4288        struct a {
4289                long    fd;
4290                long    msg;    /* char */
4291                long    len;
4292                long    flags;
4293                long    from;   /* struct sockaddr */
4294                long    fromlen;
4295        } *uap = (struct a *)ttolwp(curthread)->lwp_ap;

4297        socklen_t       fromlen;
```



```
4298        struct sonode   *so;
4299        char so_laddr[sizeof (struct sockaddr_in6)];
4300        char so_faddr[sizeof (struct sockaddr_in6)];
4301        int             fd;
4302        short so_family, so_type;
4303        int add_sock_token = 0;
4304        socklen_t len;
4305        int err;
4306        struct file *fp;
4307        struct f_audit_data *fad;               /* unix domain sockets */
4308        au_kcontext_t   *kctx = GET_KCTX_PZ;

4310        fd = (int)uap->fd;

4312        /* bail if an error */
4313        if (error) {
4314                au_uwrite(au_to_arg32(1, "so", (uint32_t)fd));
4315                au_uwrite(au_to_arg32(3, "flags", (uint32_t)(uap->flags)));
4316                return;
4317        }

4319        if ((so = getsonode(fd, &err, &fp)) == NULL) {
4320                /*
4321                 * not security relevant if doing a recvmsg from non socket
4322                 * so no extra tokens. Should probably turn off audit record
4323                 * generation here.
4324                 */
4325                return;
4326        }

4328        so_family = so->so_family;
4329        so_type   = so->so_type;

4331        /*
4332         * only putout SOCKET_EX token if INET/INET6 family.
4333         * XXX - what do we do about other families?
4334         */

4336        switch (so_family) {
4337        case AF_INET:
4338        case AF_INET6:

4340                /*
4341                 * if datagram type socket, then just use what is in
4342                 * socket structure for local address.
4343                 * XXX - what do we do for other types?
4344                 */
4345                if ((so->so_type == SOCK_DGRAM) ||
4346                    (so->so_type == SOCK_RAW)) {
4347                        add_sock_token = 1;

4349                        /* get local address */
4350                        len = sizeof (so_laddr);
4351                        (void) socket_getsockname(so,
4352                            (struct sockaddr *)so_laddr, &len, CRED());

4354                        /* get peer address */
4355                        bzero((void *)so_faddr, sizeof (so_faddr));

4357                        /* sanity check */
4358                        if (uap->from == NULL)
4359                                break;

4361                        /* sanity checks */
4362                        if (uap->fromlen == 0)
4363                                break;
```

```
4365                        if (copyin((caddr_t)(uap->fromlen), (caddr_t)&fromlen,
4366                            sizeof (fromlen)) != 0)
4367                                break;
4368
4369                        if (fromlen == 0)
4370                                break;
4371
4372                        /* enforce maximum size */
4373                        if (fromlen > sizeof (so_faddr))
4374                                fromlen = sizeof (so_faddr);
4375
4376                        if (copyin((caddr_t)(uap->from), so_faddr,
4377                            fromlen) != 0)
4378                                break;
4379
4380                } else if (so->so_type == SOCK_STREAM) {
4381
4382                        /* get path from file struct here */
4383                        fad = F2A(fp);
4384                        ASSERT(fad);
4385
4386                        /*
4387                         * already processed this file for read attempt
4388                         */
4389                        if (fad->fad_flags & FAD_READ) {
4390                                /* don't want to audit every recvfrom attempt */
4391                                tad->tad_flag = 0;
4392                                /* free any residual audit data */
4393                                au_close(kctx, &(u_ad), 0, 0, 0, NULL);
4394                                releasef(fd);
4395                                return;
4396                        }
4397                        /*
4398                         * mark things so we know what happened and don't
4399                         * repeat things
4400                         */
4401                        fad->fad_flags |= FAD_READ;
4402
4403                        bzero((void *)so_laddr, sizeof (so_laddr));
4404                        bzero((void *)so_faddr, sizeof (so_faddr));
4405
4406                        /* get local and foreign addresses */
4407                        len = sizeof (so_laddr);
4408                        (void) socket_getsockname(so,
4409                            (struct sockaddr *)so_laddr, &len, CRED());
4410                        len = sizeof (so_faddr);
4411                        (void) socket_getpeername(so,
4412                            (struct sockaddr *)so_faddr, &len, B_FALSE, CRED());
4413
4414                        add_sock_token = 1;
4415                }
4416
4417                /* XXX - what about SOCK_RDM/SOCK_SEQPACKET ??? */
4418
4419                break;
4420
4421        case AF_UNIX:
4422                /*
4423                 * first check if this is first time through. Too much
4424                 * duplicate code to put this in an aui_ routine.
4425                 */
4426
4427                /* get path from file struct here */
4428                fad = F2A(fp);
4429                ASSERT(fad);
```



```
4431                /*
4432                 * already processed this file for read attempt
4433                 */
4434                if (fad->fad_flags & FAD_READ) {
4435                        /* don't want to audit every recvfrom attempt */
4436                        tad->tad_flag = 0;
4437                        /* free any residual audit data */
4438                        au_close(kctx, &(u_ad), 0, 0, 0, NULL);
4439                        releasef(fd);
4440                        return;
4441                }
4442                /*
4443                 * mark things so we know what happened and don't
4444                 * repeat things
4445                 */
4446                fad->fad_flags |= FAD_READ;
4447
4448                if (fad->fad_aupath != NULL) {
4449                        au_uwrite(au_to_path(fad->fad_aupath));
4450                } else {
4451                        au_uwrite(au_to_arg32(1, "no path: fd", fd));
4452                }
4453
4454                audit_attributes(fp->f_vnode);
4455
4456                releasef(fd);
4457
4458                return;
4459
4460        default:
4461                break;
4462
4463        }
4464
4465        releasef(fd);
4466
4467        au_uwrite(au_to_arg32(1, "so", (uint32_t)fd));
4468
4469        if (add_sock_token == 0) {
4470                au_uwrite(au_to_arg32(1, "family", (uint32_t)so_family));
4471                au_uwrite(au_to_arg32(1, "type", (uint32_t)so_type));
4472                au_uwrite(au_to_arg32(3, "flags", (uint32_t)(uap->flags)));
4473                return;
4474        }
4475
4476        au_uwrite(au_to_arg32(3, "flags", (uint32_t)(uap->flags)));
4477
4478        au_uwrite(au_to_socket_ex(so_family, so_type, so_laddr, so_faddr));
4479 }
4480
4481 /*ARGSUSED*/
4482 static void
4483 auf_sendmsg(struct t_audit_data *tad, int error, rval_t *rval)
4484 {
4485        struct a {
4486                long    fd;
4487                long    msg;    /* struct msghdr */
4488                long    flags;
4489        } *uap = (struct a *)ttolwp(curthread)->lwp_ap;
4490
4491        struct sonode   *so;
4492        char so_laddr[sizeof (struct sockaddr_in6)];
4493        char so_faddr[sizeof (struct sockaddr_in6)];
4494        int             err;
4495        int             fd;
```

```
4496         short so_family, so_type;
4497         int             add_sock_token = 0;
4498         socklen_t       len;
4499         struct file     *fp;
4500         struct f_audit_data *fad;
4501         caddr_t         msg_name;
4502         socklen_t       msg_namelen;
4503         STRUCT_DECL(msghdr, msg);
4504         au_kcontext_t   *kctx = GET_KCTX_PZ;

4506         fd = (int)uap->fd;

4508         /* bail if an error */
4509         if (error) {
4510                 /* XXX include destination address from system call arguments */
4511                 au_uwrite(au_to_arg32(1, "so", (uint32_t)fd));
4512                 au_uwrite(au_to_arg32(3, "flags", (uint32_t)(uap->flags)));
4513                 return;
4514         }

4516         if ((so = getsonode(fd, &err, &fp)) == NULL) {
4517                 /*
4518                  * not security relevant if doing a sendmsg from non socket
4519                  * so no extra tokens. Should probably turn off audit record
4520                  * generation here.
4521                  */
4522                 return;
4523         }

4525         so_family = so->so_family;
4526         so_type   = so->so_type;

4528         switch (so_family) {
4529         case AF_INET:
4530         case AF_INET6:
4531                 /*
4532                  * if datagram type socket, then just use what is in
4533                  * socket structure for local address.
4534                  * XXX - what do we do for other types?
4535                  */
4536                 if ((so->so_type == SOCK_DGRAM) ||
4537                     (so->so_type == SOCK_RAW)) {

4539                         bzero((void *)so_laddr, sizeof (so_laddr));
4540                         bzero((void *)so_faddr, sizeof (so_faddr));

4542                         /* get local address */
4543                         len = sizeof (so_laddr);
4544                         (void) socket_getsockname(so,
4545                             (struct sockaddr *)so_laddr, &len, CRED());

4547                         /* get peer address */
4548                         STRUCT_INIT(msg, get_udatamodel());

4550                         if (copyin((caddr_t)(uap->msg),
4551                             (caddr_t)STRUCT_BUF(msg), STRUCT_SIZE(msg)) != 0) {
4552                                 break;
4553                         }
4554                         msg_name = (caddr_t)STRUCT_FGETP(msg, msg_name);
4555                         if (msg_name == NULL)
4556                                 break;

4558                         msg_namelen = (socklen_t)STRUCT_FGET(msg, msg_namelen);
4559                         /* length is value from recvmsg - sanity check */
4560                         if (msg_namelen == 0)
4561                                 break;
```



```
4563                         if (copyin(msg_name, so_faddr,
4564                             sizeof (so_faddr)) != 0)
4565                                 break;

4567                         add_sock_token = 1;

4569                 } else if (so->so_type == SOCK_STREAM) {

4571                         /* get path from file struct here */
4572                         fad = F2A(fp);
4573                         ASSERT(fad);

4575                         /*
4576                          * already processed this file for write attempt
4577                          */
4578                         if (fad->fad_flags & FAD_WRITE) {
4579                                 releasef(fd);
4580                                 /* don't want to audit every sendmsg attempt */
4581                                 tad->tad_flag = 0;
4582                                 /* free any residual audit data */
4583                                 au_close(kctx, &(u_ad), 0, 0, 0, NULL);
4584                                 return;
4585                         }

4587                         /*
4588                          * mark things so we know what happened and don't
4589                          * repeat things
4590                          */
4591                         fad->fad_flags |= FAD_WRITE;

4593                         bzero((void *)so_laddr, sizeof (so_laddr));
4594                         bzero((void *)so_faddr, sizeof (so_faddr));

4596                         /* get local and foreign addresses */
4597                         len = sizeof (so_laddr);
4598                         (void) socket_getsockname(so,
4599                             (struct sockaddr *)so_laddr, &len, CRED());
4600                         len = sizeof (so_faddr);
4601                         (void) socket_getpeername(so,
4602                             (struct sockaddr *)so_faddr, &len, B_FALSE, CRED());

4604                         add_sock_token = 1;
4605                 }

4607                 /* XXX - what about SOCK_RAW/SOCK_RDM/SOCK_SEQPACKET ??? */

4609                 break;

4611         case AF_UNIX:
4612                 /*
4613                  * first check if this is first time through. Too much
4614                  * duplicate code to put this in an aui_ routine.
4615                  */

4617                 /* get path from file struct here */
4618                 fad = F2A(fp);
4619                 ASSERT(fad);

4621                 /*
4622                  * already processed this file for write attempt
4623                  */
4624                 if (fad->fad_flags & FAD_WRITE) {
4625                         releasef(fd);
4626                         /* don't want to audit every sendmsg attempt */
4627                         tad->tad_flag = 0;
```

```
4628                        /* free any residual audit data */
4629                        au_close(kctx, &(u_ad), 0, 0, 0, NULL);
4630                        return;
4631                }
4632                /*
4633                 * mark things so we know what happened and don't
4634                 * repeat things
4635                 */
4636                fad->fad_flags |= FAD_WRITE;

4638                if (fad->fad_aupath != NULL) {
4639                        au_uwrite(au_to_path(fad->fad_aupath));
4640                } else {
4641                        au_uwrite(au_to_arg32(1, "no path: fd", fd));
4642                }

4644                audit_attributes(fp->f_vnode);

4646                releasef(fd);

4648                return;

4650        default:
4651                break;
4652        }

4654        releasef(fd);

4656        au_uwrite(au_to_arg32(1, "so", (uint32_t)fd));

4658        if (add_sock_token == 0) {
4659                au_uwrite(au_to_arg32(1, "family", (uint32_t)so_family));
4660                au_uwrite(au_to_arg32(1, "type", (uint32_t)so_type));
4661                au_uwrite(au_to_arg32(3, "flags", (uint32_t)(uap->flags)));
4662                return;
4663        }

4665        au_uwrite(au_to_arg32(3, "flags", (uint32_t)(uap->flags)));

4667        au_uwrite(au_to_socket_ex(so_family, so_type, so_laddr, so_faddr));
4668 }

4670 /*ARGSUSED*/
4671 static void
4672 auf_sendto(struct t_audit_data *tad, int error, rval_t *rval)
4673 {
4674        struct a {
4675                long    fd;
4676                long    msg;    /* char */
4677                long    len;
4678                long    flags;
4679                long    to;     /* struct sockaddr */
4680                long    tolen;
4681        } *uap = (struct a *)ttolwp(curthread)->lwp_ap;

4683        struct sonode   *so;
4684        char so_laddr[sizeof (struct sockaddr_in6)];
4685        char so_faddr[sizeof (struct sockaddr_in6)];
4686        socklen_t       tolen;
4687        int             err;
4688        int             fd;
4689        socklen_t       len;
4690        short so_family, so_type;
4691        int             add_sock_token = 0;
4692        struct file     *fp;
4693        struct f_audit_data *fad;
```



```
4694        au_kcontext_t   *kctx = GET_KCTX_PZ;

4696        fd = (int)uap->fd;

4698        /* bail if an error */
4699        if (error) {
4700                au_uwrite(au_to_arg32(1, "so", (uint32_t)fd));
4701                au_uwrite(au_to_arg32(3, "flags", (uint32_t)(uap->flags)));
4702                /* XXX include destination address from system call arguments */
4703                return;
4704        }

4706        if ((so = getsonode(fd, &err, &fp)) == NULL) {
4707                /*
4708                 * not security relevant if doing a sendto using non socket
4709                 * so no extra tokens. Should probably turn off audit record
4710                 * generation here.
4711                 */
4712                return;
4713        }

4715        so_family = so->so_family;
4716        so_type   = so->so_type;

4718        /*
4719         * only putout SOCKET_EX token if INET/INET6 family.
4720         * XXX - what do we do about other families?
4721         */

4723        switch (so_family) {
4724        case AF_INET:
4725        case AF_INET6:

4727                /*
4728                 * if datagram type socket, then just use what is in
4729                 * socket structure for local address.
4730                 * XXX - what do we do for other types?
4731                 */
4732                if ((so->so_type == SOCK_DGRAM) ||
4733                    (so->so_type == SOCK_RAW)) {

4735                        bzero((void *)so_laddr, sizeof (so_laddr));
4736                        bzero((void *)so_faddr, sizeof (so_faddr));

4738                        /* get local address */
4739                        len = sizeof (so_laddr);
4740                        (void) socket_getsockname(so,
4741                            (struct sockaddr *)so_laddr, &len, CRED());

4743                        /* get peer address */

4745                        /* sanity check */
4746                        if (uap->to == NULL)
4747                                break;

4749                        /* sanity checks */
4750                        if (uap->tolen == 0)
4751                                break;

4753                        tolen = (socklen_t)uap->tolen;

4755                        /* enforce maximum size */
4756                        if (tolen > sizeof (so_faddr))
4757                                tolen = sizeof (so_faddr);

4759                        if (copyin((caddr_t)(uap->to), so_faddr, tolen) != 0)
```

```
4760                                    break;
4762                            add_sock_token = 1;
4763                    } else {
4764                            /*
4765                             * check if this is first time through.
4766                             */
4768                            /* get path from file struct here */
4769                            fad = F2A(fp);
4770                            ASSERT(fad);
4772                            /*
4773                             * already processed this file for write attempt
4774                             */
4775                            if (fad->fad_flags & FAD_WRITE) {
4776                                    /* don't want to audit every sendto attempt */
4777                                    tad->tad_flag = 0;
4778                                    /* free any residual audit data */
4779                                    au_close(kctx, &(u_ad), 0, 0, 0, NULL);
4780                                    releasef(fd);
4781                                    return;
4782                            }
4783                            /*
4784                             * mark things so we know what happened and don't
4785                             * repeat things
4786                             */
4787                            fad->fad_flags |= FAD_WRITE;
4789                            bzero((void *)so_laddr, sizeof (so_laddr));
4790                            bzero((void *)so_faddr, sizeof (so_faddr));
4792                            /* get local and foreign addresses */
4793                            len = sizeof (so_laddr);
4794                            (void) socket_getsockname(so,
4795                                (struct sockaddr *)so_laddr, &len, CRED());
4796                            len = sizeof (so_faddr);
4797                            (void) socket_getpeername(so,
4798                                (struct sockaddr *)so_faddr, &len, B_FALSE, CRED());
4800                            add_sock_token = 1;
4801                    }
4803                    /* XXX - what about SOCK_RDM/SOCK_SEQPACKET ??? */
4805                    break;
4807            case AF_UNIX:
4808                    /*
4809                     * first check if this is first time through. Too much
4810                     * duplicate code to put this in an aui_ routine.
4811                     */
4813                    /* get path from file struct here */
4814                    fad = F2A(fp);
4815                    ASSERT(fad);
4817                    /*
4818                     * already processed this file for write attempt
4819                     */
4820                    if (fad->fad_flags & FAD_WRITE) {
4821                            /* don't want to audit every sendto attempt */
4822                            tad->tad_flag = 0;
4823                            /* free any residual audit data */
4824                            au_close(kctx, &(u_ad), 0, 0, 0, NULL);
4825                            releasef(fd);
```



```
4826                            return;
4827                    }
4828                    /*
4829                     * mark things so we know what happened and don't
4830                     * repeat things
4831                     */
4832                    fad->fad_flags |= FAD_WRITE;
4834                    if (fad->fad_aupath != NULL) {
4835                            au_uwrite(au_to_path(fad->fad_aupath));
4836                    } else {
4837                            au_uwrite(au_to_arg32(1, "no path: fd", fd));
4838                    }
4840                    audit_attributes(fp->f_vnode);
4842                    releasef(fd);
4844                    return;
4846            default:
4847                    break;
4849            }
4851            releasef(fd);
4853            au_uwrite(au_to_arg32(1, "so", (uint32_t)fd));
4855            if (add_sock_token == 0) {
4856                    au_uwrite(au_to_arg32(1, "family", (uint32_t)so_family));
4857                    au_uwrite(au_to_arg32(1, "type", (uint32_t)so_type));
4858                    au_uwrite(au_to_arg32(3, "flags", (uint32_t)(uap->flags)));
4859                    return;
4860            }
4862            au_uwrite(au_to_arg32(3, "flags", (uint32_t)(uap->flags)));
4864            au_uwrite(au_to_socket_ex(so_family, so_type, so_laddr, so_faddr));
4866    }
4868    /*
4869     * XXX socket(2) may be equivalent to open(2) on a unix domain
4870     * socket. This needs investigation.
4871     */
4873    /*ARGSUSED*/
4874    static void
4875    aus_socket(struct t_audit_data *tad)
4876    {
4877            struct a {
4878                    long    domain;
4879                    long    type;
4880                    long    protocol;
4881            } *uap = (struct a *)ttolwp(curthread)->lwp_ap;
4883            au_uwrite(au_to_arg32(1, "domain", (uint32_t)uap->domain));
4884            au_uwrite(au_to_arg32(2, "type", (uint32_t)uap->type));
4885            au_uwrite(au_to_arg32(3, "protocol", (uint32_t)uap->protocol));
4886    }
4888    /*ARGSUSED*/
4889    static void
4890    aus_sigqueue(struct t_audit_data *tad)
4891    {
```

```
4892         struct a {
4893                 long    pid;
4894                 long    signo;
4895                 long    *val;
4896         } *uap = (struct a *)ttolwp(curthread)->lwp_ap;
4897         struct proc *p;
4898         uid_t uid, ruid;
4899         gid_t gid, rgid;
4900         pid_t pid;
4901         const auditinfo_addr_t *ainfo;
4902         cred_t *cr;

4904         pid = (pid_t)uap->pid;

4906         au_uwrite(au_to_arg32(2, "signal", (uint32_t)uap->signo));
4907         if (pid > 0) {
4908                 mutex_enter(&pidlock);
4909                 if ((p = prfind(pid)) == (struct proc *)0) {
4910                         mutex_exit(&pidlock);
4911                         return;
4912                 }
4913                 mutex_enter(&p->p_lock); /* so process doesn't go away */
4914                 mutex_exit(&pidlock);

4916                 mutex_enter(&p->p_crlock);
4917                 crhold(cr = p->p_cred);
4918                 mutex_exit(&p->p_crlock);
4919                 mutex_exit(&p->p_lock);

4921                 ainfo = crgetauinfo(cr);
4922                 if (ainfo == NULL) {
4923                         crfree(cr);
4924                         return;
4925                 }

4927                 uid  = crgetuid(cr);
4928                 gid  = crgetgid(cr);
4929                 ruid = crgetruid(cr);
4930                 rgid = crgetrgid(cr);
4931                 au_uwrite(au_to_process(uid, gid, ruid, rgid, pid,
4932                     ainfo->ai_auid, ainfo->ai_asid, &ainfo->ai_termid));
4933                 crfree(cr);
4934         }
4935         else
4936                 au_uwrite(au_to_arg32(1, "process ID", (uint32_t)pid));
4937 }

4939 /*ARGSUSED*/
4940 static void
4941 aus_inst_sync(struct t_audit_data *tad)
4942 {
4943         struct a {
4944                 long    name;   /* char */
4945                 long    flags;
4946         } *uap = (struct a *)ttolwp(curthread)->lwp_ap;

4948         au_uwrite(au_to_arg32(2, "flags", (uint32_t)uap->flags));
4949 }

4951 /*ARGSUSED*/
4952 static void
4953 aus_brandsys(struct t_audit_data *tad)
4954 {
4955         klwp_t *clwp = ttolwp(curthread);

4957         struct a {
```



```
4958                 long    cmd;
4959                 long    arg1;
4960                 long    arg2;
4961                 long    arg3;
4962                 long    arg4;
4963                 long    arg5;
4964                 long    arg6;
4965         } *uap = (struct a *)clwp->lwp_ap;

4967         au_uwrite(au_to_arg32(1, "cmd", (uint_t)uap->cmd));
4968 #ifdef _LP64
4969         au_uwrite(au_to_arg64(2, "arg1", (uint64_t)uap->arg1));
4970         au_uwrite(au_to_arg64(3, "arg2", (uint64_t)uap->arg2));
4971         au_uwrite(au_to_arg64(4, "arg3", (uint64_t)uap->arg3));
4972         au_uwrite(au_to_arg64(5, "arg4", (uint64_t)uap->arg4));
4973         au_uwrite(au_to_arg64(6, "arg5", (uint64_t)uap->arg5));
4974         au_uwrite(au_to_arg64(7, "arg6", (uint64_t)uap->arg6));
4975 #else
4976         au_uwrite(au_to_arg32(2, "arg1", (uint32_t)uap->arg1));
4977         au_uwrite(au_to_arg32(3, "arg2", (uint32_t)uap->arg2));
4978         au_uwrite(au_to_arg32(4, "arg3", (uint32_t)uap->arg3));
4979         au_uwrite(au_to_arg32(5, "arg4", (uint32_t)uap->arg4));
4980         au_uwrite(au_to_arg32(6, "arg5", (uint32_t)uap->arg5));
4981         au_uwrite(au_to_arg32(7, "arg6", (uint32_t)uap->arg6));
4982 #endif
4983 }

4985 /*ARGSUSED*/
4986 static void
4987 aus_p_online(struct t_audit_data *tad)
4988 {
4989         struct a {
4990                 long    processor_id;
4991                 long    flag;
4992         } *uap = (struct a *)ttolwp(curthread)->lwp_ap;

4994         struct flags {
4995                         int     flag;
4996                         char    *cflag;
4997         } aflags[6] = {
4998                         { P_ONLINE, "P_ONLINE"},
4999                         { P_OFFLINE, "P_OFFLINE"},
5000                         { P_NOINTR, "P_NOINTR"},
5001                         { P_SPARE, "P_SPARE"},
5002                         { P_FAULTED, "P_FAULTED"},
5003                         { P_STATUS, "P_STATUS"}
5004         };
5005         int i;
5006         char *cflag;

5008         au_uwrite(au_to_arg32(1, "processor ID", (uint32_t)uap->processor_id));
5009         au_uwrite(au_to_arg32(2, "flag", (uint32_t)uap->flag));

5011         for (i = 0; i < 6; i++) {
5012                 if (aflags[i].flag == uap->flag)
5013                         break;
5014         }
5015         cflag = (i == 6) ? "bad flag":aflags[i].cflag;

5017         au_uwrite(au_to_text(cflag));
5018 }

5020 /*ARGSUSED*/
5021 static void
5022 aus_processor_bind(struct t_audit_data *tad)
5023 {
```

```
5024         struct a {
5025                 long    id_type;
5026                 long    id;
5027                 long    processor_id;
5028                 long    obind;
5029         } *uap = (struct a *)ttolwp(curthread)->lwp_ap;

5031         struct proc *p;
5032         int lwpcnt;
5033         uid_t uid, ruid;
5034         gid_t gid, rgid;
5035         pid_t pid;
5036         const auditinfo_addr_t *ainfo;
5037         cred_t *cr;

5039         au_uwrite(au_to_arg32(1, "ID type", (uint32_t)uap->id_type));
5040         au_uwrite(au_to_arg32(2, "ID", (uint32_t)uap->id));
5041         if (uap->processor_id == PBIND_NONE)
5042                 au_uwrite(au_to_text("PBIND_NONE"));
5043         else
5044                 au_uwrite(au_to_arg32(3, "processor_id",
5045                     (uint32_t)uap->processor_id));

5047         switch (uap->id_type) {
5048         case P_MYID:
5049         case P_LWPID:
5050                 mutex_enter(&pidlock);
5051                 p = ttoproc(curthread);
5052                 if (p == NULL || p->p_as == &kas) {
5053                         mutex_exit(&pidlock);
5054                         return;
5055                 }
5056                 mutex_enter(&p->p_lock);
5057                 mutex_exit(&pidlock);
5058                 lwpcnt = p->p_lwpcnt;
5059                 pid  = p->p_pid;

5061                 mutex_enter(&p->p_crlock);
5062                 crhold(cr = p->p_cred);
5063                 mutex_exit(&p->p_crlock);
5064                 mutex_exit(&p->p_lock);

5066                 ainfo = crgetauinfo(cr);
5067                 if (ainfo == NULL) {
5068                         crfree(cr);
5069                         return;
5070                 }

5072                 uid  = crgetuid(cr);
5073                 gid  = crgetgid(cr);
5074                 ruid = crgetruid(cr);
5075                 rgid = crgetrgid(cr);
5076                 au_uwrite(au_to_process(uid, gid, ruid, rgid, pid,
5077                     ainfo->ai_auid, ainfo->ai_asid, &ainfo->ai_termid));
5078                 crfree(cr);
5079                 break;
5080         case P_PID:
5081                 mutex_enter(&pidlock);
5082                 p = prfind(uap->id);
5083                 if (p == NULL || p->p_as == &kas) {
5084                         mutex_exit(&pidlock);
5085                         return;
5086                 }
5087                 mutex_enter(&p->p_lock);
5088                 mutex_exit(&pidlock);
5089                 lwpcnt = p->p_lwpcnt;
```



```
5090                 pid  = p->p_pid;

5092                 mutex_enter(&p->p_crlock);
5093                 crhold(cr = p->p_cred);
5094                 mutex_exit(&p->p_crlock);
5095                 mutex_exit(&p->p_lock);

5097                 ainfo = crgetauinfo(cr);
5098                 if (ainfo == NULL) {
5099                         crfree(cr);
5100                         return;
5101                 }

5103                 uid  = crgetuid(cr);
5104                 gid  = crgetgid(cr);
5105                 ruid = crgetruid(cr);
5106                 rgid = crgetrgid(cr);
5107                 au_uwrite(au_to_process(uid, gid, ruid, rgid, pid,
5108                     ainfo->ai_auid, ainfo->ai_asid, &ainfo->ai_termid));
5109                 crfree(cr);

5111                 break;
5112         default:
5113                 return;
5114         }

5116         if (uap->processor_id == PBIND_NONE &&
5117             (!(uap->id_type == P_LWPID && lwpcnt > 1)))
5118                 au_uwrite(au_to_text("PBIND_NONE for process"));
5119         else
5120                 au_uwrite(au_to_arg32(3, "processor_id",
5121                     (uint32_t)uap->processor_id));
5122 }

5124 /*ARGSUSED*/
5125 static au_event_t
5126 aui_doorfs(au_event_t e)
5127 {
5128         uint32_t code;

5130         struct a {              /* doorfs */
5131                 long    a1;
5132                 long    a2;
5133                 long    a3;
5134                 long    a4;
5135                 long    a5;
5136                 long    code;
5137         } *uap = (struct a *)ttolwp(curthread)->lwp_ap;

5139         /*
5140          *      audit formats for several of the
5141          *      door calls have not yet been determined
5142          */
5143         code = (uint32_t)uap->code;
5144         switch (code) {
5145         case DOOR_CALL:
5146                 e = AUE_DOORFS_DOOR_CALL;
5147                 break;
5148         case DOOR_RETURN:
5149                 e = AUE_NULL;
5150                 break;
5151         case DOOR_CREATE:
5152                 e = AUE_DOORFS_DOOR_CREATE;
5153                 break;
5154         case DOOR_REVOKE:
5155                 e = AUE_DOORFS_DOOR_REVOKE;
```

```
5156                    break;
5157            case DOOR_INFO:
5158                    e = AUE_NULL;
5159                    break;
5160            case DOOR_UCRED:
5161                    e = AUE_NULL;
5162                    break;
5163            case DOOR_BIND:
5164                    e = AUE_NULL;
5165                    break;
5166            case DOOR_UNBIND:
5167                    e = AUE_NULL;
5168                    break;
5169            case DOOR_GETPARAM:
5170                    e = AUE_NULL;
5171                    break;
5172            case DOOR_SETPARAM:
5173                    e = AUE_NULL;
5174                    break;
5175            default:        /* illegal system call */
5176                    e = AUE_NULL;
5177                    break;
5178            }

5180            return (e);
5181    }

5183    static door_node_t *
5184    au_door_lookup(int did)
5185    {
5186            vnode_t *vp;
5187            file_t *fp;

5189            if ((fp = getf(did)) == NULL)
5190                    return (NULL);
5191            /*
5192             * Use the underlying vnode (we may be namefs mounted)
5193             */
5194            if (VOP_REALVP(fp->f_vnode, &vp, NULL))
5195                    vp = fp->f_vnode;

5197            if (vp == NULL || vp->v_type != VDOOR) {
5198                    releasef(did);
5199                    return (NULL);
5200            }

5202            return (VTOD(vp));
5203    }

5205    /*ARGSUSED*/
5206    static void
5207    aus_doorfs(struct t_audit_data *tad)
5208    {

5210            struct a {              /* doorfs */
5211                    long    a1;
5212                    long    a2;
5213                    long    a3;
5214                    long    a4;
5215                    long    a5;
5216                    long    code;
5217            } *uap = (struct a *)ttolwp(curthread)->lwp_ap;

5219            door_node_t     *dp;
5220            struct proc     *p;
5221            uint32_t        did;
```



```
5222            uid_t uid, ruid;
5223            gid_t gid, rgid;
5224            pid_t pid;
5225            const auditinfo_addr_t *ainfo;
5226            cred_t *cr;

5228            did = (uint32_t)uap->a1;

5230            switch (tad->tad_event) {
5231            case AUE_DOORFS_DOOR_CALL:
5232                    au_uwrite(au_to_arg32(1, "door ID", (uint32_t)did));
5233                    if ((dp = au_door_lookup(did)) == NULL)
5234                            break;

5236                    if (DOOR_INVALID(dp)) {
5237                            releasef(did);
5238                            break;
5239                    }

5241                    if ((p = dp->door_target) == NULL) {
5242                            releasef(did);
5243                            break;
5244                    }
5245                    mutex_enter(&p->p_lock);
5246                    releasef(did);

5248                    pid  = p->p_pid;

5250                    mutex_enter(&p->p_crlock);
5251                    crhold(cr = p->p_cred);
5252                    mutex_exit(&p->p_crlock);
5253                    mutex_exit(&p->p_lock);

5255                    ainfo = crgetauinfo(cr);
5256                    if (ainfo == NULL) {
5257                            crfree(cr);
5258                            return;
5259                    }
5260                    uid  = crgetuid(cr);
5261                    gid  = crgetgid(cr);
5262                    ruid = crgetruid(cr);
5263                    rgid = crgetrgid(cr);
5264                    au_uwrite(au_to_process(uid, gid, ruid, rgid, pid,
5265                        ainfo->ai_auid, ainfo->ai_asid, &ainfo->ai_termid));
5266                    crfree(cr);
5267                    break;
5268            case AUE_DOORFS_DOOR_RETURN:
5269                    /*
5270                     * We may want to write information about
5271                     * all doors (if any) which will be copied
5272                     * by this call to the user space
5273                     */
5274                    break;
5275            case AUE_DOORFS_DOOR_CREATE:
5276                    au_uwrite(au_to_arg32(3, "door attr", (uint32_t)uap->a3));
5277                    break;
5278            case AUE_DOORFS_DOOR_REVOKE:
5279                    au_uwrite(au_to_arg32(1, "door ID", (uint32_t)did));
5280                    break;
5281            case AUE_DOORFS_DOOR_INFO:
5282                    break;
5283            case AUE_DOORFS_DOOR_CRED:
5284                    break;
5285            case AUE_DOORFS_DOOR_BIND:
5286                    break;
5287            case AUE_DOORFS_DOOR_UNBIND: {
```

```
5288                    break;
5289            }
5290            default:        /* illegal system call */
5291                    break;
5292            }
5293    }

5295    /*ARGSUSED*/
5296    static au_event_t
5297    aui_acl(au_event_t e)
5298    {
5299            struct a {
5300                    union {
5301                            long    name;   /* char */
5302                            long    fd;
5303                    }               obj;

5305                    long            cmd;
5306                    long            nentries;
5307                    long            arg;    /* aclent_t */
5308            } *uap = (struct a *)ttolwp(curthread)->lwp_ap;

5310            switch (uap->cmd) {
5311            case SETACL:
5312            case ACE_SETACL:
5313                    /*
5314                     * acl(SETACL/ACE_SETACL, ...) and facl(SETACL/ACE_SETACL, ...)
5315                     * are expected.
5316                     */
5317                    break;
5318            case GETACL:
5319            case GETACLCNT:
5320            case ACE_GETACL:
5321            case ACE_GETACLCNT:
5322                    /* do nothing for these four values. */
5323                    e = AUE_NULL;
5324                    break;
5325            default:
5326                    /* illegal system call */
5327                    break;
5328            }

5330            return (e);
5331    }

5333    static void
5334    au_acl(int cmd, int nentries, caddr_t bufp)
5335    {
5336            size_t          a_size;
5337            aclent_t        *aclbufp;
5338            ace_t           *acebufp;
5339            int             i;

5341            switch (cmd) {
5342            case GETACL:
5343            case GETACLCNT:
5344                    break;
5345            case SETACL:
5346                    if (nentries < 3)
5347                            break;

5349                    a_size = nentries * sizeof (aclent_t);

5351                    if ((aclbufp = kmem_alloc(a_size, KM_SLEEP)) == NULL)
5352                            break;
5353                    if (copyin(bufp, aclbufp, a_size)) {
```



```
5354                            kmem_free(aclbufp, a_size);
5355                            break;
5356                    }
5357                    for (i = 0; i < nentries; i++) {
5358                            au_uwrite(au_to_acl(aclbufp + i));
5359                    }
5360                    kmem_free(aclbufp, a_size);
5361                    break;

5363            case ACE_SETACL:
5364                    if (nentries < 1 || nentries > MAX_ACL_ENTRIES)
5365                            break;

5367                    a_size = nentries * sizeof (ace_t);
5368                    if ((acebufp = kmem_alloc(a_size, KM_SLEEP)) == NULL)
5369                            break;
5370                    if (copyin(bufp, acebufp, a_size)) {
5371                            kmem_free(acebufp, a_size);
5372                            break;
5373                    }
5374                    for (i = 0; i < nentries; i++) {
5375                            au_uwrite(au_to_ace(acebufp + i));
5376                    }
5377                    kmem_free(acebufp, a_size);
5378                    break;
5379            default:
5380                    break;
5381            }
5382    }

5384    /*ARGSUSED*/
5385    static void
5386    aus_acl(struct t_audit_data *tad)
5387    {
5388            struct a {
5389                    long    fname;
5390                    long    cmd;
5391                    long    nentries;
5392                    long    aclbufp;
5393            } *uap = (struct a *)ttolwp(curthread)->lwp_ap;

5395            au_uwrite(au_to_arg32(2, "cmd", (uint32_t)uap->cmd));
5396            au_uwrite(au_to_arg32(3, "nentries", (uint32_t)uap->nentries));

5398            au_acl(uap->cmd, uap->nentries, (caddr_t)uap->aclbufp);
5399    }

5401    /*ARGSUSED*/
5402    static void
5403    aus_facl(struct t_audit_data *tad)
5404    {
5405            struct a {
5406                    long    fd;
5407                    long    cmd;
5408                    long    nentries;
5409                    long    aclbufp;
5410            } *uap = (struct a *)ttolwp(curthread)->lwp_ap;
5411            struct file  *fp;
5412            struct vnode *vp;
5413            struct f_audit_data *fad;
5414            int fd;

5416            au_uwrite(au_to_arg32(2, "cmd", (uint32_t)uap->cmd));
5417            au_uwrite(au_to_arg32(3, "nentries", (uint32_t)uap->nentries));

5419            fd = (int)uap->fd;
```

```
5421         if ((fp = getf(fd)) == NULL)
5422                 return;

5424         /* get path from file struct here */
5425         fad = F2A(fp);
5426         if (fad->fad_aupath != NULL) {
5427                 au_uwrite(au_to_path(fad->fad_aupath));
5428         } else {
5429                 au_uwrite(au_to_arg32(1, "no path: fd", (uint32_t)fd));
5430         }

5432         vp = fp->f_vnode;
5433         audit_attributes(vp);

5435         /* decrement file descriptor reference count */
5436         releasef(fd);

5438         au_acl(uap->cmd, uap->nentries, (caddr_t)uap->aclbufp);
5439 }

5441 /*ARGSUSED*/
5442 static void
5443 auf_read(tad, error, rval)
5444         struct t_audit_data *tad;
5445         int error;
5446         rval_t *rval;
5447 {
5448         struct file *fp;
5449         struct f_audit_data *fad;
5450         int fd;
5451         register struct a {
5452                 long    fd;
5453         } *uap = (struct a *)ttolwp(curthread)->lwp_ap;
5454         au_kcontext_t   *kctx = GET_KCTX_PZ;

5456         fd = (int)uap->fd;

5458         /*
5459          * convert file pointer to file descriptor
5460          *   Note: fd ref count incremented here.
5461          */
5462         if ((fp = getf(fd)) == NULL)
5463                 return;

5465         /* get path from file struct here */
5466         fad = F2A(fp);
5467         ASSERT(fad);

5469         /*
5470          * already processed this file for read attempt
5471          *
5472          * XXX might be better to turn off auditing in a aui_read() routine.
5473          */
5474         if (fad->fad_flags & FAD_READ) {
5475                 /* don't really want to audit every read attempt */
5476                 tad->tad_flag = 0;
5477                 /* free any residual audit data */
5478                 au_close(kctx, &(u_ad), 0, 0, 0, NULL);
5479                 releasef(fd);
5480                 return;
5481         }
5482         /* mark things so we know what happened and don't repeat things */
5483         fad->fad_flags |= FAD_READ;

5485         if (fad->fad_aupath != NULL) {
```



```
5486                 au_uwrite(au_to_path(fad->fad_aupath));
5487         } else {
5488                 au_uwrite(au_to_arg32(1, "no path: fd", (uint32_t)fd));
5489         }

5491         /* include attributes */
5492         audit_attributes(fp->f_vnode);

5494         /* decrement file descriptor reference count */
5495         releasef(fd);
5496 }

5498 /*ARGSUSED*/
5499 static void
5500 auf_write(tad, error, rval)
5501         struct t_audit_data *tad;
5502         int error;
5503         rval_t *rval;
5504 {
5505         struct file *fp;
5506         struct f_audit_data *fad;
5507         int fd;
5508         register struct a {
5509                 long    fd;
5510         } *uap = (struct a *)ttolwp(curthread)->lwp_ap;
5511         au_kcontext_t   *kctx = GET_KCTX_PZ;

5513         fd = (int)uap->fd;

5515         /*
5516          * convert file pointer to file descriptor
5517          *   Note: fd ref count incremented here.
5518          */
5519         if ((fp = getf(fd)) == NULL)
5520                 return;

5522         /* get path from file struct here */
5523         fad = F2A(fp);
5524         ASSERT(fad);

5526         /*
5527          * already processed this file for write attempt
5528          *
5529          * XXX might be better to turn off auditing in a aus_write() routine.
5530          */
5531         if (fad->fad_flags & FAD_WRITE) {
5532                 /* don't really want to audit every write attempt */
5533                 tad->tad_flag = 0;
5534                 /* free any residual audit data */
5535                 au_close(kctx, &(u_ad), 0, 0, 0, NULL);
5536                 releasef(fd);
5537                 return;
5538         }
5539         /* mark things so we know what happened and don't repeat things */
5540         fad->fad_flags |= FAD_WRITE;

5542         if (fad->fad_aupath != NULL) {
5543                 au_uwrite(au_to_path(fad->fad_aupath));
5544         } else {
5545                 au_uwrite(au_to_arg32(1, "no path: fd", (uint32_t)fd));
5546         }

5548         /* include attributes */
5549         audit_attributes(fp->f_vnode);

5551         /* decrement file descriptor reference count */
```

```
5552         releasef(fd);
5553 }

5555 /*ARGSUSED*/
5556 static void
5557 auf_recv(tad, error, rval)
5558         struct t_audit_data *tad;
5559         int error;
5560         rval_t *rval;
5561 {
5562         struct sonode *so;
5563         char so_laddr[sizeof (struct sockaddr_in6)];
5564         char so_faddr[sizeof (struct sockaddr_in6)];
5565         struct file *fp;
5566         struct f_audit_data *fad;
5567         int fd;
5568         int err;
5569         socklen_t len;
5570         short so_family, so_type;
5571         register struct a {
5572                 long    fd;
5573         } *uap = (struct a *)ttolwp(curthread)->lwp_ap;
5574         au_kcontext_t   *kctx = GET_KCTX_PZ;

5576         /*
5577          * If there was an error, then nothing to do. Only generate
5578          * audit record on first successful recv.
5579          */
5580         if (error) {
5581                 /* Turn off audit record generation here. */
5582                 tad->tad_flag = 0;
5583                 /* free any residual audit data */
5584                 au_close(kctx, &(u_ad), 0, 0, 0, NULL);
5585                 return;
5586         }

5588         fd = (int)uap->fd;

5590         if ((so = getsonode(fd, &err, &fp)) == NULL) {
5591                 /* Turn off audit record generation here. */
5592                 tad->tad_flag = 0;
5593                 /* free any residual audit data */
5594                 au_close(kctx, &(u_ad), 0, 0, 0, NULL);
5595                 return;
5596         }

5598         /* get path from file struct here */
5599         fad = F2A(fp);
5600         ASSERT(fad);

5602         /*
5603          * already processed this file for read attempt
5604          */
5605         if (fad->fad_flags & FAD_READ) {
5606                 releasef(fd);
5607                 /* don't really want to audit every recv call */
5608                 tad->tad_flag = 0;
5609                 /* free any residual audit data */
5610                 au_close(kctx, &(u_ad), 0, 0, 0, NULL);
5611                 return;
5612         }

5614         /* mark things so we know what happened and don't repeat things */
5615         fad->fad_flags |= FAD_READ;

5617         so_family = so->so_family;
```



```
5618         so_type   = so->so_type;

5620         switch (so_family) {
5621         case AF_INET:
5622         case AF_INET6:
5623                 /*
5624                  * Only for connections.
5625                  * XXX - do we need to worry about SOCK_DGRAM or other types???
5626                  */
5627                 if (so->so_state & SS_ISBOUND) {

5629                         bzero((void *)so_laddr, sizeof (so_laddr));
5630                         bzero((void *)so_faddr, sizeof (so_faddr));

5632                         /* get local and foreign addresses */
5633                         len = sizeof (so_laddr);
5634                         (void) socket_getsockname(so,
5635                             (struct sockaddr *)so_laddr, &len, CRED());
5636                         len = sizeof (so_faddr);
5637                         (void) socket_getpeername(so,
5638                             (struct sockaddr *)so_faddr, &len, B_FALSE, CRED());

5640                         /*
5641                          * only way to drop out of switch. Note that we
5642                          * we release fd below.
5643                          */

5645                         break;
5646                 }

5648                 releasef(fd);

5650                 /* don't really want to audit every recv call */
5651                 tad->tad_flag = 0;
5652                 /* free any residual audit data */
5653                 au_close(kctx, &(u_ad), 0, 0, 0, NULL);

5655                 return;

5657         case AF_UNIX:

5659                 if (fad->fad_aupath != NULL) {
5660                         au_uwrite(au_to_path(fad->fad_aupath));
5661                 } else {
5662                         au_uwrite(au_to_arg32(1, "no path: fd", fd));
5663                 }

5665                 audit_attributes(fp->f_vnode);

5667                 releasef(fd);

5669                 return;

5671         default:
5672                 releasef(fd);

5674                 au_uwrite(au_to_arg32(1, "so", (uint32_t)fd));
5675                 au_uwrite(au_to_arg32(1, "family", (uint32_t)so_family));
5676                 au_uwrite(au_to_arg32(1, "type", (uint32_t)so_type));

5678                 return;
5679         }

5681         releasef(fd);

5683         au_uwrite(au_to_arg32(1, "so", (uint32_t)fd));
```

```
5685         au_uwrite(au_to_socket_ex(so_family, so_type, so_laddr, so_faddr));

5687 }

5689 /*ARGSUSED*/
5690 static void
5691 auf_send(tad, error, rval)
5692         struct t_audit_data *tad;
5693         int error;
5694         rval_t *rval;
5695 {
5696         struct sonode *so;
5697         char so_laddr[sizeof (struct sockaddr_in6)];
5698         char so_faddr[sizeof (struct sockaddr_in6)];
5699         struct file *fp;
5700         struct f_audit_data *fad;
5701         int fd;
5702         int err;
5703         socklen_t len;
5704         short so_family, so_type;
5705         register struct a {
5706                 long    fd;
5707         } *uap = (struct a *)ttolwp(curthread)->lwp_ap;
5708         au_kcontext_t   *kctx = GET_KCTX_PZ;

5710         fd = (int)uap->fd;

5712         /*
5713          * If there was an error, then nothing to do. Only generate
5714          * audit record on first successful send.
5715          */
5716         if (error != 0) {
5717                 /* Turn off audit record generation here. */
5718                 tad->tad_flag = 0;
5719                 /* free any residual audit data */
5720                 au_close(kctx, &(u_ad), 0, 0, 0, NULL);
5721                 return;
5722         }

5724         fd = (int)uap->fd;

5726         if ((so = getsonode(fd, &err, &fp)) == NULL) {
5727                 /* Turn off audit record generation here. */
5728                 tad->tad_flag = 0;
5729                 /* free any residual audit data */
5730                 au_close(kctx, &(u_ad), 0, 0, 0, NULL);
5731                 return;
5732         }

5734         /* get path from file struct here */
5735         fad = F2A(fp);
5736         ASSERT(fad);

5738         /*
5739          * already processed this file for write attempt
5740          */
5741         if (fad->fad_flags & FAD_WRITE) {
5742                 releasef(fd);
5743                 /* don't really want to audit every send call */
5744                 tad->tad_flag = 0;
5745                 /* free any residual audit data */
5746                 au_close(kctx, &(u_ad), 0, 0, 0, NULL);
5747                 return;
5748         }
```



```
5750         /* mark things so we know what happened and don't repeat things */
5751         fad->fad_flags |= FAD_WRITE;

5753         so_family = so->so_family;
5754         so_type   = so->so_type;

5756         switch (so_family) {
5757         case AF_INET:
5758         case AF_INET6:
5759                 /*
5760                  * Only for connections.
5761                  * XXX - do we need to worry about SOCK_DGRAM or other types???
5762                  */
5763                 if (so->so_state & SS_ISBOUND) {

5765                         bzero((void *)so_laddr, sizeof (so_laddr));
5766                         bzero((void *)so_faddr, sizeof (so_faddr));

5768                         /* get local and foreign addresses */
5769                         len = sizeof (so_laddr);
5770                         (void) socket_getsockname(so,
5771                             (struct sockaddr *)so_laddr, &len, CRED());
5772                         len = sizeof (so_faddr);
5773                         (void) socket_getpeername(so,
5774                             (struct sockaddr *)so_faddr, &len, B_FALSE, CRED());

5776                         /*
5777                          * only way to drop out of switch. Note that we
5778                          * we release fd below.
5779                          */

5781                         break;
5782                 }

5784                 releasef(fd);
5785                 /* don't really want to audit every send call */
5786                 tad->tad_flag = 0;
5787                 /* free any residual audit data */
5788                 au_close(kctx, &(u_ad), 0, 0, 0, NULL);

5790                 return;

5792         case AF_UNIX:

5794                 if (fad->fad_aupath != NULL) {
5795                         au_uwrite(au_to_path(fad->fad_aupath));
5796                 } else {
5797                         au_uwrite(au_to_arg32(1, "no path: fd", fd));
5798                 }

5800                 audit_attributes(fp->f_vnode);

5802                 releasef(fd);

5804                 return;

5806         default:
5807                 releasef(fd);

5809                 au_uwrite(au_to_arg32(1, "so", (uint32_t)fd));
5810                 au_uwrite(au_to_arg32(1, "family", (uint32_t)so_family));
5811                 au_uwrite(au_to_arg32(1, "type", (uint32_t)so_type));

5813                 return;
5814         }
```

```
5816         releasef(fd);

5818         au_uwrite(au_to_arg32(1, "so", (uint32_t)fd));

5820         au_uwrite(au_to_socket_ex(so_family, so_type, so_laddr, so_faddr));
5821 }

5823 static au_event_t
5824 aui_forksys(au_event_t e)
5825 {
5826         struct a {
5827                 long    subcode;
5828                 long    flags;
5829         } *uap = (struct a *)ttolwp(curthread)->lwp_ap;

5831         switch ((uint_t)uap->subcode) {
5832         case 0:
5833                 e = AUE_FORK1;
5834                 break;
5835         case 1:
5836                 e = AUE_FORKALL;
5837                 break;
5838         case 2:
5839                 e = AUE_VFORK;
5840                 break;
5841         default:
5842                 e = AUE_NULL;
5843                 break;
5844         }

5846         return (e);
5847 }

5849 /*ARGSUSED*/
5850 static au_event_t
5851 aui_portfs(au_event_t e)
5852 {
5853         struct a {              /* portfs */
5854                 long    a1;
5855                 long    a2;
5856                 long    a3;
5857         } *uap = (struct a *)ttolwp(curthread)->lwp_ap;

5859         /*
5860          * check opcode
5861          */
5862         switch (((uint_t)uap->a1) & PORT_CODE_MASK) {
5863         case PORT_ASSOCIATE:
5864                 /* check source */
5865                 if (((uint_t)uap->a3 == PORT_SOURCE_FILE) ||
5866                     ((uint_t)uap->a3 == PORT_SOURCE_FD)) {
5867                         e = AUE_PORTFS_ASSOCIATE;
5868                 } else {
5869                         e = AUE_NULL;
5870                 }
5871                 break;
5872         case PORT_DISSOCIATE:
5873                 /* check source */
5874                 if (((uint_t)uap->a3 == PORT_SOURCE_FILE) ||
5875                     ((uint_t)uap->a3 == PORT_SOURCE_FD)) {
5876                         e = AUE_PORTFS_DISSOCIATE;
5877                 } else {
5878                         e = AUE_NULL;
5879                 }
5880                 break;
5881         default:
```



```
5882                 e = AUE_NULL;
5883         }
5884         return (e);
5885 }
```

```
**********************************************************
   15523 Wed May 27 19:49:25 2015
new/usr/src/uts/common/c2/audit_kevents.h
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright (c) 1992, 2010, Oracle and/or its affiliates. All rights reserved.
 */

#ifndef _BSM_AUDIT_KEVENTS_H
#define _BSM_AUDIT_KEVENTS_H

#ifdef __cplusplus
extern "C" {
#endif

/*
 * Audit event numbers.
 *
 *      0               Reserved as an invalid event number.
 *      1 -   511       Allocated for Solaris kernel
 *      512 -  2047     (reserved but not allocated)
 *      2048 - 32767    Reserved for the Solaris TCB application.
 *      32768 - 65535   Available for third party applications.
 *
 *      NOTE:   libbsm/audit_event.txt must be updated elsewhere when changes
 *              are made to kernel events.
 */

#define AUE_NULL                0       /* =no indir system call */
#define AUE_EXIT                1       /* =ps exit(2) */
#define AUE_FORKALL             2       /* =ps forkall(2) */
#define AUE_OPEN                3       /* =no open(2): place holder */
#define AUE_CREAT               4       /* =no obsolete */
#define AUE_LINK                5       /* =fc link(2) */
#define AUE_UNLINK              6       /* =fd unlink(2) */
#define AUE_EXEC                7       /* =no obsolete */
#define AUE_CHDIR               8       /* =pm chdir(2) */
#define AUE_MKNOD               9       /* =fc mknod(2) */
#define AUE_CHMOD               10      /* =fm chmod(2) */
#define AUE_CHOWN               11      /* =fm chown(2) */
#define AUE_UMOUNT              12      /* =as umount(2): old version */
#define AUE_JUNK                13      /* =no non existant event */
```



```
#define AUE_ACCESS              14      /* =fa access(2) */
#define AUE_KILL                15      /* =pm kill(2) */
#define AUE_STAT                16      /* =fa stat(2) */
#define AUE_LSTAT               17      /* =fa lstat(2) */
#define AUE_ACCT                18      /* =as acct(2) */
#define AUE_MCTL                19      /* =no mctl(2) */
#define AUE_REBOOT              20      /* =no reboot(2) */
#define AUE_SYMLINK             21      /* =fc symlink(2) */
#define AUE_READLINK            22      /* =fr readlink(2) */
#define AUE_EXECVE              23      /* =ps,ex execve(2) */
#define AUE_CHROOT              24      /* =pm chroot(2) */
#define AUE_VFORK               25      /* =ps vfork(2) */
#define AUE_SETGROUPS           26      /* =pm setgroups(2) */
#define AUE_SETPGRP             27      /* =pm setpgrp(2) */
#define AUE_SWAPON              28      /* =no swapon(2) */
#define AUE_SETHOSTNAME         29      /* =no sethostname(2) */
#define AUE_FCNTL               30      /* =fm fcntl(2) */
#define AUE_SETPRIORITY         31      /* =no setpriority(2) */
#define AUE_CONNECT             32      /* =nt connect(2) */
#define AUE_ACCEPT              33      /* =nt accept(2) */
#define AUE_BIND                34      /* =nt bind(2) */
#define AUE_SETSOCKOPT          35      /* =nt setsockopt(2) */
#define AUE_VTRACE              36      /* =no vtrace(2) */
#define AUE_SETTIMEOFDAY        37      /* =no settimeofday(2) */
#define AUE_FCHOWN              38      /* =fm fchown(2) */
#define AUE_FCHMOD              39      /* =fm fchmod(2) */
#define AUE_SETREUID            40      /* =pm setreuid(2) */
#define AUE_SETREGID            41      /* =pm setregid(2) */
#define AUE_RENAME              42      /* =fc,fd rename(2) */
#define AUE_TRUNCATE            43      /* =no truncate(2) */
#define AUE_FTRUNCATE           44      /* =no ftruncate(2) */
#define AUE_FLOCK               45      /* =no flock(2) */
#define AUE_SHUTDOWN            46      /* =nt shutdown(2) */
#define AUE_MKDIR               47      /* =fc mkdir(2) */
#define AUE_RMDIR               48      /* =fd rmdir(2) */
#define AUE_UTIMES              49      /* =fm futimens(2), utimensat(2) */
#define AUE_ADJTIME             50      /* =as adjtime(2) */
#define AUE_SETRLIMIT           51      /* =ua setrlimit(2) */
#define AUE_KILLPG              52      /* =no killpg(2) */
#define AUE_NFS_SVC             53      /* =no nfs_svc(2) */
#define AUE_STATFS              54      /* =fa statfs(2) */
#define AUE_FSTATFS             55      /* =fa fstatfs(2) */
#define AUE_UNMOUNT             56      /* =no unmount(2) */
#define AUE_ASYNC_DAEMON        57      /* =no async_daemon(2) */
#define AUE_NFS_GETFH           58      /* =no nfs_getfh(2) */
#define AUE_SETDOMAINNAME       59      /* =no setdomainname(2) */
#define AUE_QUOTACTL            60      /* =no quotactl(2) */
#define AUE_EXPORTFS            61      /* =no exportfs(2) */
#define AUE_MOUNT               62      /* =as mount(2) */
#define AUE_SEMSYS              63      /* =no semsys(2): place holder */
#define AUE_MSGSYS              64      /* =no msgsys(2): place holder */
#define AUE_SHMSYS              65      /* =no shmsys(2): place holder */
#define AUE_BSMSYS              66      /* =no bsmsys(2): place holder */
#define AUE_RFSSYS              67      /* =no rfssys(2): place holder */
#define AUE_FCHDIR              68      /* =pm fchdir(2) */
#define AUE_FCHROOT             69      /* =pm fchroot(2) */
#define AUE_VPIXSYS             70      /* =no obsolete */
#define AUE_PATHCONF            71      /* =fa pathconf(2) */
#define AUE_OPEN_R              72      /* =fr open(2): read */
#define AUE_OPEN_RC             73      /* =fc,fr open(2): read,creat */
#define AUE_OPEN_RT             74      /* =fd,fr open(2): read,trunc */
#define AUE_OPEN_RTC            75      /* =fc,fd,fr open(2): rd,cr,tr */
#define AUE_OPEN_W              76      /* =fw open(2): write */
#define AUE_OPEN_WC             77      /* =fc,fw open(2): write,creat */
#define AUE_OPEN_WT             78      /* =fd,fw open(2): write,trunc */
#define AUE_OPEN_WTC            79      /* =fc,fd,fw open(2): wr,cr,tr */
```

```
125 #define AUE_OPEN_RW             80      /* =fr,fw open(2): read,write */
126 #define AUE_OPEN_RWC            81      /* =fc,fw,fr open(2): rd,wr,cr */
127 #define AUE_OPEN_RWT            82      /* =fd,fr,fw open(2): rd,wr,tr */
128 #define AUE_OPEN_RWTC           83      /* =fc,fd,fw,fr open(2): rd,wr,cr,tr */
129 #define AUE_MSGCTL              84      /* =ip msgctl(2): illegal command */
130 #define AUE_MSGCTL_RMID         85      /* =ip msgctl(2): IPC_RMID command */
131 #define AUE_MSGCTL_SET          86      /* =ip msgctl(2): IPC_SET command */
132 #define AUE_MSGCTL_STAT         87      /* =ip msgctl(2): IPC_STAT command */
133 #define AUE_MSGGET              88      /* =ip msgget(2) */
134 #define AUE_MSGRCV              89      /* =ip msgrcv(2) */
135 #define AUE_MSGSND              90      /* =ip msgsnd(2) */
136 #define AUE_SHMCTL              91      /* =ip shmctl(2): Illegal command */
137 #define AUE_SHMCTL_RMID         92      /* =ip shmctl(2): IPC_RMID command */
138 #define AUE_SHMCTL_SET          93      /* =ip shmctl(2): IPC_SET command */
139 #define AUE_SHMCTL_STAT         94      /* =ip shmctl(2): IPC_STAT command */
140 #define AUE_SHMGET              95      /* =ip shmget(2) */
141 #define AUE_SHMAT               96      /* =ip shmat(2) */
142 #define AUE_SHMDT               97      /* =ip shmdt(2) */
143 #define AUE_SEMCTL              98      /* =ip semctl(2): illegal command */
144 #define AUE_SEMCTL_RMID         99      /* =ip semctl(2): IPC_RMID command */
145 #define AUE_SEMCTL_SET          100     /* =ip semctl(2): IPC_SET command */
146 #define AUE_SEMCTL_STAT         101     /* =ip semctl(2): IPC_STAT command */
147 #define AUE_SEMCTL_GETNCNT      102     /* =ip semctl(2): GETNCNT command */
148 #define AUE_SEMCTL_GETPID       103     /* =ip semctl(2): GETPID command */
149 #define AUE_SEMCTL_GETVAL       104     /* =ip semctl(2): GETVAL command */
150 #define AUE_SEMCTL_GETALL       105     /* =ip semctl(2): GETALL command */
151 #define AUE_SEMCTL_GETZCNT      106     /* =ip semctl(2): GETZCNT command */
152 #define AUE_SEMCTL_SETVAL       107     /* =ip semctl(2): SETVAL command */
153 #define AUE_SEMCTL_SETALL       108     /* =ip semctl(2): SETALL command */
154 #define AUE_SEMGET              109     /* =ip semget(2) */
155 #define AUE_SEMOP               110     /* =ip semop(2) */
156 #define AUE_CORE                111     /* =fc process dumped core */
157 #define AUE_CLOSE               112     /* =cl close(2) */
158 #define AUE_SYSTEMBOOT          113     /* =na system booted */
159 #define AUE_ASYNC_DAEMON_EXIT   114     /* =no async_daemon(2) exited */
160 #define AUE_NFSSVC_EXIT         115     /* =no nfssvc(2) exited */
161 #define AUE_PFEXEC              116     /* =ps,ex,ua,as execve(2) w/ pfexec */
162 #define AUE_OPEN_S              117     /* =fr open(2): search */
163 #define AUE_OPEN_E              118     /* =fr open(2): exec */
164 /*
165  * 119 - 129 are available for future growth (old SunOS_CMW events
166  * that had no libbsm or praudit support or references)
167  */
168 #define AUE_GETAUID             130     /* =aa getauid(2) */
169 #define AUE_SETAUID             131     /* =aa setauid(2) */
170 #define AUE_GETAUDIT            132     /* =aa getaudit(2) */
171 #define AUE_SETAUDIT            133     /* =aa setaudit(2) */
172 /*                              134         OBSOLETE */
173 /*                              135         OBSOLETE */
174 #define AUE_AUDITSVC            136     /* =no obsolete */
175 /*                              137         OBSOLETE */
176 #define AUE_AUDITON             138     /* =no auditon(2) */
177 #define AUE_AUDITON_GTERMID     139     /* =no auditctl(2): GETTERMID */
178 #define AUE_AUDITON_STERMID     140     /* =no auditctl(2): SETTERMID */
179 #define AUE_AUDITON_GPOLICY     141     /* =aa auditctl(2): GETPOLICY */
180 #define AUE_AUDITON_SPOLICY     142     /* =as auditctl(2): SETPOLICY */
181 #define AUE_AUDITON_GESTATE     143     /* =no auditctl(2): GETESTATE */
182 #define AUE_AUDITON_SESTATE     144     /* =no auditctl(2): SETESTATE */
183 #define AUE_AUDITON_GQCTRL      145     /* =as auditctl(2): GETQCTRL */
184 #define AUE_AUDITON_SQCTRL      146     /* =as auditctl(2): SETQCTRL */
185 /*                              147         OBSOLETE */
186 /*                              148         OBSOLETE */
187 /*                              149         OBSOLETE */
188 /*                              150         OBSOLETE */
189 /*                              151         OBSOLETE */
190 /*                              152         OBSOLETE */
```



```
191 #define AUE_ENTERPROM           153     /* =na enter prom */
192 #define AUE_EXITPROM            154     /* =na exit prom */
193 /*                              155         OBSOLETE */
194 /*                              156         OBSOLETE */
195 /*                              157         OBSOLETE */
196 #define AUE_IOCTL               158     /* =io ioctl(2) */
197 /*                              159         OBSOLETE */
198 /*                              160         OBSOLETE */
199 /*                              161         OBSOLETE */
200 /*                              162         OBSOLETE */
201 /*                              163         OBSOLETE */
202 /*                              164         OBSOLETE */
203 /*                              165         OBSOLETE */
204 /*                              166         OBSOLETE */
205 /*                              167         OBSOLETE */
206 /*                              168         OBSOLETE */
207 /*                              169         OBSOLETE */
208 /*                              170         OBSOLETE */
209 /*                              171         OBSOLETE */
210 /*                              172         OBSOLETE */
211 #define AUE_ONESIDE             173     /* =no one-sided session record */
212 #define AUE_MSGGETL             174     /* =no msggetl(2) */
213 #define AUE_MSGRCVL             175     /* =no msgrcvl(2) */
214 #define AUE_MSGSNDL             176     /* =no msgsndl(2) */
215 #define AUE_SEMGETL             177     /* =no semgetl(2) */
216 #define AUE_SHMGETL             178     /* =no shmgetl(2) */
217 /*                              179         OBSOLETE */
218 /*                              180         OBSOLETE */
219 /*                              181         OBSOLETE */
220 /*                              182         OBSOLETE */
221 #define AUE_SOCKET              183     /* =nt socket(2) */
222 #define AUE_SENDTO              184     /* =nt sendto(2) */
223 #define AUE_PIPE                185     /* =no pipe(2) */
224 #define AUE_SOCKETPAIR          186     /* =no socketpair(2) */
225 #define AUE_SEND                187     /* =no send(2) */
226 #define AUE_SENDMSG             188     /* =nt sendmsg(2) */
227 #define AUE_RECV                189     /* =no recv(2) */
228 #define AUE_RECVMSG             190     /* =nt recvmsg(2) */
229 #define AUE_RECVFROM            191     /* =nt recvfrom(2) */
230 #define AUE_READ                192     /* =no read(2) */
231 #define AUE_GETDENTS            193     /* =no getdents(2) */
232 #define AUE_LSEEK               194     /* =no lseek(2) */
233 #define AUE_WRITE               195     /* =no write(2) */
234 #define AUE_WRITEV              196     /* =no writev(2) */
235 #define AUE_NFS                 197     /* =no NFS server */
236 #define AUE_READV               198     /* =no readv(2) */
237 #define AUE_OSTAT               199     /* =no obsolete */
238 #define AUE_SETUID              200     /* =pm old setuid(2) */
239 #define AUE_STIME               201     /* =as old stime(2) */
240 #define AUE_UTIME               202     /* =no obsolete */
241 #define AUE_NICE                203     /* =pm old nice(2) */
242 #define AUE_OSETPGRP            204     /* =no old setpgrp(2) */
243 #define AUE_SETGID              205     /* =pm old setgid(2) */
244 #define AUE_READL               206     /* =no readl(2) */
245 #define AUE_READVL              207     /* =no readvl(2) */
246 #define AUE_FSTAT               208     /* =no fstat(2) */
247 #define AUE_DUP2                209     /* =no obsolete */
248 #define AUE_MMAP                210     /* =no mmap(2) u-o-p */
249 #define AUE_AUDIT               211     /* =no audit(2) u-o-p */
250 #define AUE_PRIOCNTLSYS         212     /* =pm priocntlsys */
251 #define AUE_MUNMAP              213     /* =cl munmap(2) u-o-p */
252 #define AUE_SETEGID             214     /* =pm setegid(2) */
253 #define AUE_SETEUID             215     /* =pm seteuid(2) */
254 #define AUE_PUTMSG              216     /* =nt */
255 #define AUE_GETMSG              217     /* =nt */
256 #define AUE_PUTPMSG             218     /* =nt */
```

```
257 #define AUE_GETPMSG             219     /* =nt */
258 #define AUE_AUDITSYS            220     /* =no place holder */
259 #define AUE_AUDITON_GETKMASK    221     /* =aa */
260 #define AUE_AUDITON_SETKMASK    222     /* =as */
261 #define AUE_AUDITON_GETCWD      223     /* =aa,as */
262 #define AUE_AUDITON_GETCAR      224     /* =aa,as */
263 #define AUE_AUDITON_GETSTAT     225     /* =as */
264 #define AUE_AUDITON_SETSTAT     226     /* =as */
265 #define AUE_AUDITON_SETUMASK    227     /* =as */
266 #define AUE_AUDITON_SETSMASK    228     /* =as */
267 #define AUE_AUDITON_GETCOND     229     /* =aa */
268 #define AUE_AUDITON_SETCOND     230     /* =as */
269 #define AUE_AUDITON_GETCLASS    231     /* =aa,as */
270 #define AUE_AUDITON_SETCLASS    232     /* =as */
271 #define AUE_FUSERS              233     /* =fa */
272 #define AUE_STATVFS             234     /* =fa */
273 #define AUE_XSTAT               235     /* =no obsolete */
274 #define AUE_LXSTAT              236     /* =no obsolete */
275 #define AUE_LCHOWN              237     /* =fm */
276 #define AUE_MEMCNTL             238     /* =ot */
277 #define AUE_SYSINFO             239     /* =as */
278 #define AUE_XMKNOD              240     /* =no obsolete */
279 #define AUE_FORK1               241     /* =ps */
280 #define AUE_MODCTL              242     /* =no */
281 #define AUE_MODLOAD             243     /* =as */
282 #define AUE_MODUNLOAD           244     /* =as */
283 #define AUE_MODCONFIG           245     /* =no obsolete */
284 #define AUE_MODADDMAJ           246     /* =as */
285 #define AUE_SOCKACCEPT          247     /* =nt */
286 #define AUE_SOCKCONNECT         248     /* =nt */
287 #define AUE_SOCKSEND            249     /* =nt */
288 #define AUE_SOCKRECEIVE         250     /* =nt */
289 #define AUE_ACLSET              251     /* =fm */
290 #define AUE_FACLSET             252     /* =fm */
291 #define AUE_DOORFS              253     /* =no */
292 #define AUE_DOORFS_DOOR_CALL    254     /* =ip */
293 #define AUE_DOORFS_DOOR_RETURN  255     /* =ip */
294 #define AUE_DOORFS_DOOR_CREATE  256     /* =ip */
295 #define AUE_DOORFS_DOOR_REVOKE  257     /* =ip */
296 #define AUE_DOORFS_DOOR_INFO    258     /* =ip */
297 #define AUE_DOORFS_DOOR_CRED    259     /* =ip */
298 #define AUE_DOORFS_DOOR_BIND    260     /* =ip */
299 #define AUE_DOORFS_DOOR_UNBIND  261     /* =ip */
300 #define AUE_P_ONLINE            262     /* =as */
301 #define AUE_PROCESSOR_BIND      263     /* =as */
302 #define AUE_INST_SYNC           264     /* =as */
303 #define AUE_SOCKCONFIG          265     /* =nt */
304 #define AUE_SETAUDIT_ADDR       266     /* =aa setaudit_addr(2) */
305 #define AUE_GETAUDIT_ADDR       267     /* =aa getaudit_addr(2) */
306 #define AUE_UMOUNT2             268     /* =as umount2(2) */
307 #define AUE_FSAT                269     /* =no obsolete */
308 #define AUE_OPENAT_R            270     /* =no obsolete */
309 #define AUE_OPENAT_RC           271     /* =no obsolete */
310 #define AUE_OPENAT_RT           272     /* =no obsolete */
311 #define AUE_OPENAT_RTC          273     /* =no obsolete */
312 #define AUE_OPENAT_W            274     /* =no obsolete */
313 #define AUE_OPENAT_WC           275     /* =no obsolete */
314 #define AUE_OPENAT_WT           276     /* =no obsolete */
315 #define AUE_OPENAT_WTC          277     /* =no obsolete */
316 #define AUE_OPENAT_RW           278     /* =no obsolete */
317 #define AUE_OPENAT_RWC          279     /* =no obsolete */
318 #define AUE_OPENAT_RWT          280     /* =no obsolete */
319 #define AUE_OPENAT_RWTC         281     /* =no obsolete */
320 #define AUE_RENAMEAT            282     /* =no obsolete */
321 #define AUE_FSTATAT             283     /* =no obsolete */
322 #define AUE_FCHOWNAT            284     /* =no obsolete */
```



```
323 #define AUE_FUTIMESAT           285     /* =no obsolete */
324 #define AUE_UNLINKAT            286     /* =no obsolete */
325 #define AUE_CLOCK_SETTIME       287     /* =as clock_settime(3RT) */
326 #define AUE_NTP_ADJTIME         288     /* =as ntp_adjtime(2) */
327 #define AUE_SETPPRIV            289     /* =pm setppriv(2) */
328 #define AUE_MODDEVPLCY          290     /* =as modctl(2) */
329 #define AUE_MODADDPRIV          291     /* =as modctl(2) */
330 #define AUE_CRYPTOADM           292     /* =as kernel cryptographic framework */
331 #define AUE_CONFIGKSSL          293     /* =as kernel SSL */
332 #define AUE_BRANDSYS            294     /* =ot */
333 #define AUE_PF_POLICY_ADDRULE   295     /* =as Add IPsec policy rule */
334 #define AUE_PF_POLICY_DELRULE   296     /* =as Delete IPsec policy rule */
335 #define AUE_PF_POLICY_CLONE     297     /* =as Clone IPsec policy */
336 #define AUE_PF_POLICY_FLIP      298     /* =as Flip IPsec policy */
337 #define AUE_PF_POLICY_FLUSH     299     /* =as Flush IPsec policy rules */
338 #define AUE_PF_POLICY_ALGS      300     /* =as Update IPsec algorithms */
339 #define AUE_PORTFS              301     /* =no portfs(2) - place holder */
340 #define AUE_LABELSYS_TNRH       302     /* =as tnrh(2) */
341 #define AUE_LABELSYS_TNRHTP     303     /* =as tnrhtp(2) */
342 #define AUE_LABELSYS_TNMLP      304     /* =as tnmlp(2) */
343 #define AUE_PORTFS_ASSOCIATE    305     /* =fa portfs(2) - port associate */
344 #define AUE_PORTFS_DISSOCIATE   306     /* =fa portfs(2) - port disassociate */
345 #define AUE_SETSID              307     /* =pm setsid(2) */
346 #define AUE_SETPGID             308     /* =pm setpgid(2) */
347 #define AUE_FACCESSAT           309     /* =no obsolete */
348 #define AUE_AUDITON_GETAMASK    310     /* =aa */
349 #define AUE_AUDITON_SETAMASK    311     /* =as */
350 #define AUE_PSECFLAGS           312     /* =pm psecflags */


352 /* NOTE: update MAX_KEVENTS below if events are added. */
353 #define MAX_KEVENTS             312

355 #define MAX_KEVENTS             311


355 #ifdef __cplusplus
356 }
______unchanged_portion_omitted_
```

```
**********************************************************
   58041 Wed May 27 19:49:25 2015
new/usr/src/uts/common/exec/elf/elf.c
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright (c) 1989, 2010, Oracle and/or its affiliates. All rights reserved.
 */

/*      Copyright (c) 1984, 1986, 1987, 1988, 1989 AT&T */
/*        All Rights Reserved   */
/*
 * Copyright (c) 2013, Joyent, Inc.  All rights reserved.
 */

#include <sys/types.h>
#include <sys/param.h>
#include <sys/thread.h>
#include <sys/sysmacros.h>
#include <sys/signal.h>
#include <sys/cred.h>
#include <sys/user.h>
#include <sys/errno.h>
#include <sys/vnode.h>
#include <sys/mman.h>
#include <sys/kmem.h>
#include <sys/proc.h>
#include <sys/pathname.h>
#include <sys/cmn_err.h>
#include <sys/systm.h>
#include <sys/elf.h>
#include <sys/vmsystm.h>
#include <sys/debug.h>
#include <sys/auxv.h>
#include <sys/exec.h>
#include <sys/prsystm.h>
#include <vm/as.h>
#include <vm/rm.h>
#include <vm/seg.h>
#include <vm/seg_vn.h>
#include <sys/modctl.h>
#include <sys/systeminfo.h>
```



```
#include <sys/vmparam.h>
#include <sys/machelf.h>
#include <sys/shm_impl.h>
#include <sys/archsystm.h>
#include <sys/fasttrap.h>
#include <sys/brand.h>
#include "elf_impl.h"
#include <sys/sdt.h>
#include <sys/siginfo.h>
#include <sys/random.h>
#endif /* ! codereview */

extern int at_flags;
extern volatile size_t aslr_max_brk_skew;
#endif /* ! codereview */

#define ORIGIN_STR      "ORIGIN"
#define ORIGIN_STR_SIZE 6

static int getelfhead(vnode_t *, cred_t *, Ehdr *, int *, int *, int *);
static int getelfphdr(vnode_t *, cred_t *, const Ehdr *, int, caddr_t *,
    ssize_t *);
static int getelfshdr(vnode_t *, cred_t *, const Ehdr *, int, int, caddr_t *,
    ssize_t *, caddr_t *, ssize_t *);
static size_t elfsize(Ehdr *, int, caddr_t, uintptr_t *);
static int mapelfexec(vnode_t *, Ehdr *, int, caddr_t,
    Phdr **, Phdr **, Phdr **, Phdr **, Phdr *,
    caddr_t *, caddr_t *, intptr_t *, intptr_t *, size_t, long *, size_t *);

typedef enum {
        STR_CTF,
        STR_SYMTAB,
        STR_DYNSYM,
        STR_STRTAB,
        STR_DYNSTR,
        STR_SHSTRTAB,
        STR_NUM
} shstrtype_t;

static const char *shstrtab_data[] = {
        ".SUNW_ctf",
        ".symtab",
        ".dynsym",
        ".strtab",
        ".dynstr",
        ".shstrtab"
};

typedef struct shstrtab {
        int     sst_ndx[STR_NUM];
        int     sst_cur;
} shstrtab_t;

static void
shstrtab_init(shstrtab_t *s)
{
        bzero(&s->sst_ndx, sizeof (s->sst_ndx));
        s->sst_cur = 1;
}

static int
shstrtab_ndx(shstrtab_t *s, shstrtype_t type)
{
        int ret;

        if ((ret = s->sst_ndx[type]) != 0)
```

```
125                 return (ret);

127         ret = s->sst_ndx[type] = s->sst_cur;
128         s->sst_cur += strlen(shstrtab_data[type]) + 1;

130         return (ret);
131 }

133 static size_t
134 shstrtab_size(const shstrtab_t *s)
135 {
136         return (s->sst_cur);
137 }

139 static void
140 shstrtab_dump(const shstrtab_t *s, char *buf)
141 {
142         int i, ndx;

144         *buf = '\0';
145         for (i = 0; i < STR_NUM; i++) {
146                 if ((ndx = s->sst_ndx[i]) != 0)
147                         (void) strcpy(buf + ndx, shstrtab_data[i]);
148         }
149 }

151 static int
152 dtrace_safe_phdr(Phdr *phdrp, struct uarg *args, uintptr_t base)
153 {
154         ASSERT(phdrp->p_type == PT_SUNWDTRACE);

156         /*
157          * See the comment in fasttrap.h for information on how to safely
158          * update this program header.
159          */
160         if (phdrp->p_memsz < PT_SUNWDTRACE_SIZE ||
161             (phdrp->p_flags & (PF_R | PF_W | PF_X)) != (PF_R | PF_W | PF_X))
162                 return (-1);

164         args->thrptr = phdrp->p_vaddr + base;

166         return (0);
167 }

169 /*
170  * Map in the executable pointed to by vp. Returns 0 on success.
171  */
172 int
173 mapexec_brand(vnode_t *vp, uarg_t *args, Ehdr *ehdr, Addr *uphdr_vaddr,
174     intptr_t *voffset, caddr_t exec_file, int *interp, caddr_t *bssbase,
175     caddr_t *brkbase, size_t *brksize, uintptr_t *lddatap)
176 {
177         size_t          len;
178         struct vattr    vat;
179         caddr_t         phdrbase = NULL;
180         ssize_t         phdrsize;
181         int             nshdrs, shstrndx, nphdrs;
182         int             error = 0;
183         Phdr            *uphdr = NULL;
184         Phdr            *junk = NULL;
185         Phdr            *dynphdr = NULL;
186         Phdr            *dtrphdr = NULL;
187         uintptr_t       lddata;
188         long            execsz;
189         intptr_t        minaddr;
```



```
191         if (lddatap != NULL)
192                 *lddatap = NULL;

194         if (error = execpermissions(vp, &vat, args)) {
195                 uprintf("%s: Cannot execute %s\n", exec_file, args->pathname);
196                 return (error);
197         }

199         if ((error = getelfhead(vp, CRED(), ehdr, &nshdrs, &shstrndx,
200             &nphdrs)) != 0 ||
201             (error = getelfphdr(vp, CRED(), ehdr, nphdrs, &phdrbase,
202             &phdrsize)) != 0) {
203                 uprintf("%s: Cannot read %s\n", exec_file, args->pathname);
204                 return (error);
205         }

207         if ((len = elfsize(ehdr, nphdrs, phdrbase, &lddata)) == 0) {
208                 uprintf("%s: Nothing to load in %s", exec_file, args->pathname);
209                 kmem_free(phdrbase, phdrsize);
210                 return (ENOEXEC);
211         }
212         if (lddatap != NULL)
213                 *lddatap = lddata;

215         if (error = mapelfexec(vp, ehdr, nphdrs, phdrbase, &uphdr, &dynphdr,
216             &junk, &dtrphdr, NULL, bssbase, brkbase, voffset, &minaddr,
217             len, &execsz, brksize)) {
218                 uprintf("%s: Cannot map %s\n", exec_file, args->pathname);
219                 kmem_free(phdrbase, phdrsize);
220                 return (error);
221         }

223         /*
224          * Inform our caller if the executable needs an interpreter.
225          */
226         *interp = (dynphdr == NULL) ? 0 : 1;

228         /*
229          * If this is a statically linked executable, voffset should indicate
230          * the address of the executable itself (it normally holds the address
231          * of the interpreter).
232          */
233         if (ehdr->e_type == ET_EXEC && *interp == 0)
234                 *voffset = minaddr;

236         if (uphdr != NULL) {
237                 *uphdr_vaddr = uphdr->p_vaddr;
238         } else {
239                 *uphdr_vaddr = (Addr)-1;
240         }

242         kmem_free(phdrbase, phdrsize);
243         return (error);
244 }

246 /*ARGSUSED*/
247 int
248 elfexec(vnode_t *vp, execa_t *uap, uarg_t *args, intpdata_t *idatap,
249     int level, long *execsz, int setid, caddr_t exec_file, cred_t *cred,
250     int brand_action)
251 {
252         caddr_t         phdrbase = NULL;
253         caddr_t         bssbase = 0;
254         caddr_t         brkbase = 0;
255         size_t          brksize = 0;
256         ssize_t         dlnsize;
```

```
257         aux_entry_t     *aux;
258         int             error;
259         ssize_t         resid;
260         int             fd = -1;
261         intptr_t        voffset;
262         Phdr            *intphdr = NULL;
263         Phdr            *dynamicphdr = NULL;
 68         Phdr            *dyphdr = NULL;
264         Phdr            *stphdr = NULL;
265         Phdr            *uphdr = NULL;
266         Phdr            *junk = NULL;
267         size_t          len;
268         ssize_t         phdrsize;
269         int             postfixsize = 0;
270         int             i, hsize;
271         Phdr            *phdrp;
272         Phdr            *dataphdrp = NULL;
273         Phdr            *dtrphdr;
274         Phdr            *capphdr = NULL;
275         Cap             *cap = NULL;
276         ssize_t         capsize;
277         Dyn             *dyn = NULL;
278         ssize_t         dynsize;
279 #endif /* ! codereview */
280         int             hasu = 0;
281         int             hasauxv = 0;
282         int             hasintp = 0;
 82         int             hasdy = 0;
283         int             branded = 0;

285         struct proc *p = ttoproc(curthread);
286         struct user *up = PTOU(p);
287         struct bigwad {
288                 Ehdr    ehdr;
289                 aux_entry_t     elfargs[__KERN_NAUXV_IMPL];
290                 char            dl_name[MAXPATHLEN];
291                 char            pathbuf[MAXPATHLEN];
292                 struct vattr    vattr;
293                 struct execenv  exenv;
294         } *bigwad;      /* kmem_alloc this behemoth so we don't blow stack */
295         Ehdr            *ehdrp;
296         int             nshdrs, shstrndx, nphdrs;
297         char            *dlnp;
298         char            *pathbufp;
299         rlim64_t        limit;
300         rlim64_t        roundlimit;

302         ASSERT(p->p_model == DATAMODEL_ILP32 || p->p_model == DATAMODEL_LP64);

304         bigwad = kmem_alloc(sizeof (struct bigwad), KM_SLEEP);
305         ehdrp = &bigwad->ehdr;
306         dlnp = bigwad->dl_name;
307         pathbufp = bigwad->pathbuf;

309         /*
310          * Obtain ELF and program header information.
311          */
312         if ((error = getelfhead(vp, CRED(), ehdrp, &nshdrs, &shstrndx,
313             &nphdrs)) != 0 ||
314             (error = getelfphdr(vp, CRED(), ehdrp, nphdrs, &phdrbase,
315             &phdrsize)) != 0)
316                 goto out;

318         /*
319          * Prevent executing an ELF file that has no entry point.
320          */
```



```
321         if (ehdrp->e_entry == 0) {
322                 uprintf("%s: Bad entry point\n", exec_file);
323                 goto bad;
324         }

326         /*
327          * Put data model that we're exec-ing to into the args passed to
328          * exec_args(), so it will know what it is copying to on new stack.
329          * Now that we know whether we are exec-ing a 32-bit or 64-bit
330          * executable, we can set execsz with the appropriate NCARGS.
331          */
332 #ifdef  _LP64
333         if (ehdrp->e_ident[EI_CLASS] == ELFCLASS32) {
334                 args->to_model = DATAMODEL_ILP32;
335                 *execsz = btopr(SINCR) + btopr(SSIZE) + btopr(NCARGS32-1);
336         } else {
337                 args->to_model = DATAMODEL_LP64;
338                 args->stk_prot &= ~PROT_EXEC;
339 #if defined(__i386) || defined(__amd64)
340                 args->dat_prot &= ~PROT_EXEC;
341 #endif
342                 *execsz = btopr(SINCR) + btopr(SSIZE) + btopr(NCARGS64-1);
343         }
344 #else   /* _LP64 */
345         args->to_model = DATAMODEL_ILP32;
346         *execsz = btopr(SINCR) + btopr(SSIZE) + btopr(NCARGS-1);
347 #endif  /* _LP64 */

349         /*
350          * We delay invoking the brand callback until we've figured out
351          * what kind of elf binary we're trying to run, 32-bit or 64-bit.
352          * We do this because now the brand library can just check
353          * args->to_model to see if the target is 32-bit or 64-bit without
354          * having do duplicate all the code above.
355          */
356         if ((level < 2) &&
357             (brand_action != EBA_NATIVE) && (PROC_IS_BRANDED(p))) {
358                 error = BROP(p)->b_elfexec(vp, uap, args,
359                     idatap, level + 1, execsz, setid, exec_file, cred,
360                     brand_action);
361                 goto out;
362         }

364         /*
365          * Determine aux size now so that stack can be built
366          * in one shot (except actual copyout of aux image),
367          * determine any non-default stack protections,
368          * and still have this code be machine independent.
369          */
370         hsize = ehdrp->e_phentsize;
371         phdrp = (Phdr *)phdrbase;
372         for (i = nphdrs; i > 0; i--) {
373                 switch (phdrp->p_type) {
374                 case PT_INTERP:
375                         hasauxv = hasintp = 1;
175                         hasauxv = hasdy = 1;
376                         break;
377                 case PT_PHDR:
378                         hasu = 1;
379                         break;
380                 case PT_SUNWSTACK:
381                         args->stk_prot = PROT_USER;
382                         if (phdrp->p_flags & PF_R)
383                                 args->stk_prot |= PROT_READ;
384                         if (phdrp->p_flags & PF_W)
385                                 args->stk_prot |= PROT_WRITE;
```

```
386                         if (phdrp->p_flags & PF_X)
387                                 args->stk_prot |= PROT_EXEC;
388                         break;
389                 case PT_LOAD:
390                         dataphdrp = phdrp;
391                         break;
392                 case PT_SUNWCAP:
393                         capphdr = phdrp;
394                         break;
395                 case PT_DYNAMIC:
396                         dynamicphdr = phdrp;
397                         break;
398 #endif /* ! codereview */
399                 }
400                 phdrp = (Phdr *)((caddr_t)phdrp + hsize);
401         }

403         if (ehdrp->e_type != ET_EXEC) {
404                 dataphdrp = NULL;
405                 hasauxv = 1;
406         }

408         /* Copy BSS permissions to args->dat_prot */
409         if (dataphdrp != NULL) {
410                 args->dat_prot = PROT_USER;
411                 if (dataphdrp->p_flags & PF_R)
412                         args->dat_prot |= PROT_READ;
413                 if (dataphdrp->p_flags & PF_W)
414                         args->dat_prot |= PROT_WRITE;
415                 if (dataphdrp->p_flags & PF_X)
416                         args->dat_prot |= PROT_EXEC;
417         }

419         /*
420          * If a auxvector will be required - reserve the space for
421          * it now.  This may be increased by exec_args if there are
422          * ISA-specific types (included in __KERN_NAUXV_IMPL).
423          */
424         if (hasauxv) {
425                 /*
426                  * If a AUX vector is being built - the base AUX
427                  * entries are:
428                  *
429                  *      AT_BASE
430                  *      AT_FLAGS
431                  *      AT_PAGESZ
432                  *      AT_SUN_AUXFLAGS
433                  *      AT_SUN_HWCAP
434                  *      AT_SUN_HWCAP2
435                  *      AT_SUN_SECFLAGS
436 #endif /* ! codereview */
437                  *      AT_SUN_PLATFORM (added in stk_copyout)
438                  *      AT_SUN_EXECNAME (added in stk_copyout)
439                  *      AT_NULL
440                  *
441                  * total == 10
195                  * total == 9
442                  */
443                 if (hasintp && hasu) {
197                 if (hasdy && hasu) {
444                         /*
445                          * Has PT_INTERP & PT_PHDR - the auxvectors that
446                          * will be built are:
447                          *
448                          *      AT_PHDR
449                          *      AT_PHENT
```



```
450                          *      AT_PHNUM
451                          *      AT_ENTRY
452                          *      AT_LDDATA
453                          *
454                          * total = 5
455                          */
456                         args->auxsize = (10 + 5) * sizeof (aux_entry_t);
457                 } else if (hasintp) {
210                         args->auxsize = (9 + 5) * sizeof (aux_entry_t);
211                 } else if (hasdy) {
458                         /*
459                          * Has PT_INTERP but no PT_PHDR
460                          *
461                          *      AT_EXECFD
462                          *      AT_LDDATA
463                          *
464                          * total = 2
465                          */
466                         args->auxsize = (10 + 2) * sizeof (aux_entry_t);
220                         args->auxsize = (9 + 2) * sizeof (aux_entry_t);
467                 } else {
468                         args->auxsize = 10 * sizeof (aux_entry_t);
222                         args->auxsize = 9 * sizeof (aux_entry_t);
469                 }
470         } else {
471                 args->auxsize = 0;
472         }

474         /*
475          * If this binary is using an emulator, we need to add an
476          * AT_SUN_EMULATOR aux entry.
477          */
478         if (args->emulator != NULL)
479                 args->auxsize += sizeof (aux_entry_t);

481         if ((brand_action != EBA_NATIVE) && (PROC_IS_BRANDED(p))) {
482                 branded = 1;
483                 /*
484                  * We will be adding 4 entries to the aux vectors.  One for
485                  * the the brandname and 3 for the brand specific aux vectors.
486                  */
487                 args->auxsize += 4 * sizeof (aux_entry_t);
488         }

490         /* If the binary has an explicit ASLR flag, it must be honoured */
491         if (dynamicphdr != NULL) {
492                 Dyn *dp;

494                 dynsize = dynamicphdr->p_filesz;
495                 dyn = kmem_alloc(dynsize, KM_SLEEP);

497                 if ((error = vn_rdwr(UIO_READ, vp, (caddr_t)dyn, dynsize,
498                     (offset_t)dynamicphdr->p_offset, UIO_SYSSPACE, 0, (rlim64_t)
499                     CRED(), &resid)) != 0) {
500                         uprintf("%s: cannot read .dynamic section\n",
501                             exec_file);
502                         goto out;
503                 }

505                 if (resid != 0)
506                         goto out;

508                 dp = dyn;
509                 while (dp->d_tag != DT_NULL) {
510                         if (dp->d_tag == DT_SUNW_ASLR) {
511                                 if (dp->d_un.d_val != 0) {
```

```
512                                     curproc->p_secflags.psf_effective |=
513                                         PROC_SEC_ASLR;
514                                     curproc->p_secflags.psf_inherit |=
515                                         PROC_SEC_ASLR;
516
517                                 } else {
518                                     curproc->p_secflags.psf_effective &=
519                                         ~PROC_SEC_ASLR;
520                                     curproc->p_secflags.psf_inherit &=
521                                         ~PROC_SEC_ASLR;
522                                 }
523                         }
524                         dp++;
525                 }
526         }
527
528 #endif /* ! codereview */
529         /* Hardware/Software capabilities */
530         if (capphdr != NULL &&
531             (capsize = capphdr->p_filesz) > 0 &&
532             capsize <= 16 * sizeof (*cap)) {
533                 int ncaps = capsize / sizeof (*cap);
534                 Cap *cp;
535
536                 cap = kmem_alloc(capsize, KM_SLEEP);
537                 if ((error = vn_rdwr(UIO_READ, vp, (caddr_t)cap,
538                     capsize, (offset_t)capphdr->p_offset,
539                     UIO_SYSSPACE, 0, (rlim64_t)0, CRED(), &resid)) != 0) {
540                         uprintf("%s: Cannot read capabilities section\n",
541                             exec_file);
542                         goto out;
543                 }
544                 for (cp = cap; cp < cap + ncaps; cp++) {
545                         if (cp->c_tag == CA_SUNW_SF_1 &&
546                             (cp->c_un.c_val & SF1_SUNW_ADDR32)) {
547                                 if (args->to_model == DATAMODEL_LP64)
548                                         args->addr32 = 1;
549                                 break;
550                         }
551                 }
552         }
553
554         aux = bigwad->elfargs;
555         /*
556          * Move args to the user's stack.
557          * This can fill in the AT_SUN_PLATFORM and AT_SUN_EXECNAME aux entries.
558          */
559         if ((error = exec_args(uap, args, idatap, (void **)&aux)) != 0) {
560                 if (error == -1) {
561                         error = ENOEXEC;
562                         goto bad;
563                 }
564                 goto out;
565         }
566         /* we're single threaded after this point */
567
568         /*
569          * If this is an ET_DYN executable (shared object),
570          * determine its memory size so that mapelfexec() can load it.
571          */
572         if (ehdrp->e_type == ET_DYN)
573                 len = elfsize(ehdrp, nphdrs, phdrbase, NULL);
574         else
575                 len = 0;
576
577         dtrphdr = NULL;
```



```
578
579         if ((error = mapelfexec(vp, ehdrp, nphdrs, phdrbase, &uphdr, &intphdr,
244         if ((error = mapelfexec(vp, ehdrp, nphdrs, phdrbase, &uphdr, &dyphdr,
580             &stphdr, &dtrphdr, dataphdrp, &bssbase, &brkbase, &voffset, NULL,
581             len, execsz, &brksize)) != 0)
582                 goto bad;
583
584         if (uphdr != NULL && intphdr == NULL)
249         if (uphdr != NULL && dyphdr == NULL)
585                 goto bad;
586
587         if (dtrphdr != NULL && dtrace_safe_phdr(dtrphdr, args, voffset) != 0) {
588                 uprintf("%s: Bad DTrace phdr in %s\n", exec_file, exec_file);
589                 goto bad;
590         }
591
592         if (intphdr != NULL) {
257         if (dyphdr != NULL) {
593                 size_t          len;
594                 uintptr_t       lddata;
595                 char            *p;
596                 struct vnode    *nvp;
597
598                 dlnsize = intphdr->p_filesz;
263                 dlnsize = dyphdr->p_filesz;
599
600                 if (dlnsize > MAXPATHLEN || dlnsize <= 0)
601                         goto bad;
602
603                 /*
604                  * Read in "interpreter" pathname.
605                  */
606                 if ((error = vn_rdwr(UIO_READ, vp, dlnp, intphdr->p_filesz,
607                     (offset_t)intphdr->p_offset, UIO_SYSSPACE, 0, (rlim64_t)0,
271                 if ((error = vn_rdwr(UIO_READ, vp, dlnp, dyphdr->p_filesz,
272                     (offset_t)dyphdr->p_offset, UIO_SYSSPACE, 0, (rlim64_t)0,
608                     CRED(), &resid)) != 0) {
609                         uprintf("%s: Cannot obtain interpreter pathname\n",
610                             exec_file);
611                         goto bad;
612                 }
613
614                 if (resid != 0 || dlnp[dlnsize - 1] != '\0')
615                         goto bad;
616
617                 /*
618                  * Search for '$ORIGIN' token in interpreter path.
619                  * If found, expand it.
620                  */
621                 for (p = dlnp; p = strchr(p, '$'); ) {
622                         uint_t  len, curlen;
623                         char    *_ptr;
624
625                         if (strncmp(++p, ORIGIN_STR, ORIGIN_STR_SIZE))
626                                 continue;
627
628                         curlen = 0;
629                         len = p - dlnp - 1;
630                         if (len) {
631                                 bcopy(dlnp, pathbufp, len);
632                                 curlen += len;
633                         }
634                         if (_ptr = strrchr(args->pathname, '/')) {
635                                 len = _ptr - args->pathname;
636                                 if ((curlen + len) > MAXPATHLEN)
637                                         break;
```

```
639                             bcopy(args->pathname, &pathbufp[curlen], len);
640                             curlen += len;
641                     } else {
642                             /*
643                              * executable is a basename found in the
644                              * current directory.  So - just substitue
645                              * '.' for ORIGIN.
646                              */
647                             pathbufp[curlen] = '.';
648                             curlen++;
649                     }
650                     p += ORIGIN_STR_SIZE;
651                     len = strlen(p);

653                     if ((curlen + len) > MAXPATHLEN)
654                             break;
655                     bcopy(p, &pathbufp[curlen], len);
656                     curlen += len;
657                     pathbufp[curlen++] = '\0';
658                     bcopy(pathbufp, dlnp, curlen);
659             }

661             /*
662              * /usr/lib/ld.so.1 is known to be a symlink to /lib/ld.so.1
663              * (and /usr/lib/64/ld.so.1 is a symlink to /lib/64/ld.so.1).
664              * Just in case /usr is not mounted, change it now.
665              */
666             if (strcmp(dlnp, USR_LIB_RTLD) == 0)
667                     dlnp += 4;
668             error = lookupname(dlnp, UIO_SYSSPACE, FOLLOW, NULLVPP, &nvp);
669             if (error && dlnp != bigwad->dl_name) {
670                     /* new kernel, old user-level */
671                     error = lookupname(dlnp -= 4, UIO_SYSSPACE, FOLLOW,
672                         NULLVPP, &nvp);
673             }
674             if (error) {
675                     uprintf("%s: Cannot find %s\n", exec_file, dlnp);
676                     goto bad;
677             }

679             /*
680              * Setup the "aux" vector.
681              */
682             if (uphdr) {
683                     if (ehdrp->e_type == ET_DYN) {
684                             /* don't use the first page */
685                             bigwad->exenv.ex_brkbase = (caddr_t)PAGESIZE;
686                             bigwad->exenv.ex_bssbase = (caddr_t)PAGESIZE;
687                     } else {
688                             bigwad->exenv.ex_bssbase = bssbase;
689                             bigwad->exenv.ex_brkbase = brkbase;
690                     }
691                     bigwad->exenv.ex_brksize = brksize;
692                     bigwad->exenv.ex_magic = elfmagic;
693                     bigwad->exenv.ex_vp = vp;
694                     setexecenv(&bigwad->exenv);

696                     ADDAUX(aux, AT_PHDR, uphdr->p_vaddr + voffset)
697                     ADDAUX(aux, AT_PHENT, ehdrp->e_phentsize)
698                     ADDAUX(aux, AT_PHNUM, nphdrs)
699                     ADDAUX(aux, AT_ENTRY, ehdrp->e_entry + voffset)
700             } else {
701                     if ((error = execopen(&vp, &fd)) != 0) {
702                             VN_RELE(nvp);
703                             goto bad;
```



```
704                             }

706                             ADDAUX(aux, AT_EXECFD, fd)
707                     }

709                     if ((error = execpermissions(nvp, &bigwad->vattr, args)) != 0) {
710                             VN_RELE(nvp);
711                             uprintf("%s: Cannot execute %s\n", exec_file, dlnp);
712                             goto bad;
713                     }

715                     /*
716                      * Now obtain the ELF header along with the entire program
717                      * header contained in "nvp".
718                      */
719                     kmem_free(phdrbase, phdrsize);
720                     phdrbase = NULL;
721                     if ((error = getelfhead(nvp, CRED(), ehdrp, &nshdrs,
722                         &shstrndx, &nphdrs)) != 0 ||
723                         (error = getelfphdr(nvp, CRED(), ehdrp, nphdrs, &phdrbase,
724                         &phdrsize)) != 0) {
725                             VN_RELE(nvp);
726                             uprintf("%s: Cannot read %s\n", exec_file, dlnp);
727                             goto bad;
728                     }

730                     /*
731                      * Determine memory size of the "interpreter's" loadable
732                      * sections.  This size is then used to obtain the virtual
733                      * address of a hole, in the user's address space, large
734                      * enough to map the "interpreter".
735                      */
736                     if ((len = elfsize(ehdrp, nphdrs, phdrbase, &lddata)) == 0) {
737                             VN_RELE(nvp);
738                             uprintf("%s: Nothing to load in %s\n", exec_file, dlnp);
739                             goto bad;
740                     }

742                     dtrphdr = NULL;

744                     error = mapelfexec(nvp, ehdrp, nphdrs, phdrbase, &junk, &junk,
745                         &junk, &dtrphdr, NULL, NULL, NULL, &voffset, NULL, len,
746                         execsz, NULL);
747                     if (error || junk != NULL) {
748                             VN_RELE(nvp);
749                             uprintf("%s: Cannot map %s\n", exec_file, dlnp);
750                             goto bad;
751                     }

753                     /*
754                      * We use the DTrace program header to initialize the
755                      * architecture-specific user per-LWP location. The dtrace
756                      * fasttrap provider requires ready access to per-LWP scratch
757                      * space. We assume that there is only one such program header
758                      * in the interpreter.
759                      */
760                     if (dtrphdr != NULL &&
761                         dtrace_safe_phdr(dtrphdr, args, voffset) != 0) {
762                             VN_RELE(nvp);
763                             uprintf("%s: Bad DTrace phdr in %s\n", exec_file, dlnp);
764                             goto bad;
765                     }

767                     VN_RELE(nvp);
768                     ADDAUX(aux, AT_SUN_LDDATA, voffset + lddata)
769             }
```

```
771         if (hasauxv) {
772                 int auxf = AF_SUN_HWCAPVERIFY;
773                 /*
774                  * Note: AT_SUN_PLATFORM and AT_SUN_EXECNAME were filled in via
775                  * exec_args()
776                  */
777                 ADDAUX(aux, AT_BASE, voffset)
778                 ADDAUX(aux, AT_FLAGS, at_flags)
779                 ADDAUX(aux, AT_PAGESZ, PAGESIZE)
780                 /*
781                  * Linker flags. (security)
782                  * p_flag not yet set at this time.
783                  * We rely on gexec() to provide us with the information.
784                  * If the application is set-uid but this is not reflected
785                  * in a mismatch between real/effective uids/gids, then
786                  * don't treat this as a set-uid exec.  So we care about
787                  * the EXECSETID_UGIDS flag but not the ...SETID flag.
788                  */
789                 if ((setid &= ~EXECSETID_SETID) != 0)
790                         auxf |= AF_SUN_SETUGID;

792                 /*
793                  * If we're running a native process from within a branded
794                  * zone under pfexec then we clear the AF_SUN_SETUGID flag so
795                  * that the native ld.so.1 is able to link with the native
796                  * libraries instead of using the brand libraries that are
797                  * installed in the zone.  We only do this for processes
798                  * which we trust because we see they are already running
799                  * under pfexec (where uid != euid).  This prevents a
800                  * malicious user within the zone from crafting a wrapper to
801                  * run native suid commands with unsecure libraries interposed.
802                  */
803                 if ((brand_action == EBA_NATIVE) && (PROC_IS_BRANDED(p) &&
804                     (setid &= ~EXECSETID_SETID) != 0))
805                         auxf &= ~AF_SUN_SETUGID;

807                 /*
808                  * Record the user addr of the auxflags aux vector entry
809                  * since brands may optionally want to manipulate this field.
810                  */
811                 args->auxp_auxflags =
812                     (char *)((char *)args->stackend +
813                     ((char *)&aux->a_type -
814                     (char *)bigwad->elfargs));
815                 ADDAUX(aux, AT_SUN_AUXFLAGS, auxf);

817                 /*
818                  * Put the effective security-flags into the aux vector, for
819                  * the sake of flags that need partial (or complete)
820                  * implementation in userland.
821                  */
822                 ADDAUX(aux, AT_SUN_SECFLAGS, p->p_secflags.psf_effective);
823 #endif /* ! codereview */
824                 /*
825                  * Hardware capability flag word (performance hints)
826                  * Used for choosing faster library routines.
827                  * (Potentially different between 32-bit and 64-bit ABIs)
828                  */
829 #if defined(_LP64)
830                 if (args->to_model == DATAMODEL_NATIVE) {
831                         ADDAUX(aux, AT_SUN_HWCAP, auxv_hwcap)
832                         ADDAUX(aux, AT_SUN_HWCAP2, auxv_hwcap_2)
833                 } else {
834                         ADDAUX(aux, AT_SUN_HWCAP, auxv_hwcap32)
835                         ADDAUX(aux, AT_SUN_HWCAP2, auxv_hwcap32_2)
```



```
836                 }
837 #else
838                 ADDAUX(aux, AT_SUN_HWCAP, auxv_hwcap)
839                 ADDAUX(aux, AT_SUN_HWCAP2, auxv_hwcap_2)
840 #endif
841                 if (branded) {
842                         /*
843                          * Reserve space for the brand-private aux vectors,
844                          * and record the user addr of that space.
845                          */
846                         args->auxp_brand =
847                             (char *)((char *)args->stackend +
848                             ((char *)&aux->a_type -
849                             (char *)bigwad->elfargs));
850                         ADDAUX(aux, AT_SUN_BRAND_AUX1, 0)
851                         ADDAUX(aux, AT_SUN_BRAND_AUX2, 0)
852                         ADDAUX(aux, AT_SUN_BRAND_AUX3, 0)
853                 }

855                 ADDAUX(aux, AT_NULL, 0)
856                 postfixsize = (char *)aux - (char *)bigwad->elfargs;

858                 /*
859                  * We make assumptions above when we determine how many aux
860                  * vector entries we will be adding. However, if we have an
861                  * invalid elf file, it is possible that mapelfexec might
862                  * behave differently (but not return an error), in which case
863                  * the number of aux entries we actually add will be different.
864                  * We detect that now and error out.
865                  */
866                 if (postfixsize != args->auxsize) {
867                         DTRACE_PROBE2(elfexec_badaux, int, postfixsize,
868                             int, args->auxsize);
869                         goto bad;
870                 }
871                 ASSERT(postfixsize <= __KERN_NAUXV_IMPL * sizeof (aux_entry_t));
872         }

874         /*
875          * For the 64-bit kernel, the limit is big enough that rounding it up
876          * to a page can overflow the 64-bit limit, so we check for btopr()
877          * overflowing here by comparing it with the unrounded limit in pages.
878          * If it hasn't overflowed, compare the exec size with the rounded up
879          * limit in pages.  Otherwise, just compare with the unrounded limit.
880          */
881         limit = btop(p->p_vmem_ctl);
882         roundlimit = btopr(p->p_vmem_ctl);
883         if ((roundlimit > limit && *execsz > roundlimit) ||
884             (roundlimit < limit && *execsz > limit)) {
885                 mutex_enter(&p->p_lock);
886                 (void) rctl_action(rctlproc_legacy[RLIMIT_VMEM], p->p_rctls, p,
887                     RCA_SAFE);
888                 mutex_exit(&p->p_lock);
889                 error = ENOMEM;
890                 goto bad;
891         }

893         bzero(up->u_auxv, sizeof (up->u_auxv));
894         if (postfixsize) {
895                 int num_auxv;

897                 /*
898                  * Copy the aux vector to the user stack.
899                  */
900                 error = execpoststack(args, bigwad->elfargs, postfixsize);
901                 if (error)
```

```
902                         goto bad;

904                 /*
905                  * Copy auxv to the process's user structure for use by /proc.
906                  * If this is a branded process, the brand's exec routine will
907                  * copy it's private entries to the user structure later. It
908                  * relies on the fact that the blank entries are at the end.
909                  */
910                 num_auxv = postfixsize / sizeof (aux_entry_t);
911                 ASSERT(num_auxv <= sizeof (up->u_auxv) / sizeof (auxv_t));
912                 aux = bigwad->elfargs;
913                 for (i = 0; i < num_auxv; i++) {
914                         up->u_auxv[i].a_type = aux[i].a_type;
915                         up->u_auxv[i].a_un.a_val = (aux_val_t)aux[i].a_un.a_val;
916                 }
917         }

919         /*
920          * Pass back the starting address so we can set the program counter.
921          */
922         args->entry = (uintptr_t)(ehdrp->e_entry + voffset);

924         if (!uphdr) {
925                 if (ehdrp->e_type == ET_DYN) {
926                         /*
927                          * If we are executing a shared library which doesn't
928                          * have a interpreter (probably ld.so.1) then
929                          * we don't set the brkbase now.  Instead we
930                          * delay it's setting until the first call
931                          * via grow.c::brk().  This permits ld.so.1 to
932                          * initialize brkbase to the tail of the executable it
933                          * loads (which is where it needs to be).
934                          */
935                         bigwad->exenv.ex_brkbase = (caddr_t)0;
936                         bigwad->exenv.ex_bssbase = (caddr_t)0;
937                         bigwad->exenv.ex_brksize = 0;
938                 } else {
939                         bigwad->exenv.ex_brkbase = brkbase;
940                         bigwad->exenv.ex_bssbase = bssbase;
941                         bigwad->exenv.ex_brksize = brksize;
942                 }
943                 bigwad->exenv.ex_magic = elfmagic;
944                 bigwad->exenv.ex_vp = vp;
945                 setexecenv(&bigwad->exenv);
946         }

948         ASSERT(error == 0);
949         goto out;

951 bad:
952         if (fd != -1)           /* did we open the a.out yet */
953                 (void) execclose(fd);

955         psignal(p, SIGKILL);

957         if (error == 0)
958                 error = ENOEXEC;
959 out:
960         if (phdrbase != NULL)
961                 kmem_free(phdrbase, phdrsize);
962         if (cap != NULL)
963                 kmem_free(cap, capsize);
964         if (dyn != NULL)
965                 kmem_free(dyn, dynsize);
966 #endif /* ! codereview */
967         kmem_free(bigwad, sizeof (struct bigwad));
```



```
968         return (error);
969 }

971 /*
972  * Compute the memory size requirement for the ELF file.
973  */
974 static size_t
975 elfsize(Ehdr *ehdrp, int nphdrs, caddr_t phdrbase, uintptr_t *lddata)
976 {
977         size_t  len;
978         Phdr    *phdrp = (Phdr *)phdrbase;
979         int     hsize = ehdrp->e_phentsize;
980         int     first = 1;
981         int     dfirst = 1;     /* first data segment */
982         uintptr_t loaddr = 0;
983         uintptr_t hiaddr = 0;
984         uintptr_t lo, hi;
985         int     i;

987         for (i = nphdrs; i > 0; i--) {
988                 if (phdrp->p_type == PT_LOAD) {
989                         lo = phdrp->p_vaddr;
990                         hi = lo + phdrp->p_memsz;
991                         if (first) {
992                                 loaddr = lo;
993                                 hiaddr = hi;
994                                 first = 0;
995                         } else {
996                                 if (loaddr > lo)
997                                         loaddr = lo;
998                                 if (hiaddr < hi)
999                                         hiaddr = hi;
1000                        }

1002                        /*
1003                         * save the address of the first data segment
1004                         * of a object - used for the AT_SUNW_LDDATA
1005                         * aux entry.
1006                         */
1007                        if ((lddata != NULL) && dfirst &&
1008                            (phdrp->p_flags & PF_W)) {
1009                                *lddata = lo;
1010                                dfirst = 0;
1011                        }
1012                }
1013                phdrp = (Phdr *)((caddr_t)phdrp + hsize);
1014        }

1016        len = hiaddr - (loaddr & PAGEMASK);
1017        len = roundup(len, PAGESIZE);

1019        return (len);
1020 }

1022 /*
1023  * Read in the ELF header and program header table.
1024  * SUSV3 requires:
1025  *      ENOEXEC File format is not recognized
1026  *      EINVAL  Format recognized but execution not supported
1027  */
1028 static int
1029 getelfhead(vnode_t *vp, cred_t *credp, Ehdr *ehdr, int *nshdrs, int *shstrndx,
1030     int *nphdrs)
1031 {
1032         int error;
1033         ssize_t resid;
```

```
1035         /*
1036          * We got here by the first two bytes in ident,
1037          * now read the entire ELF header.
1038          */
1039         if ((error = vn_rdwr(UIO_READ, vp, (caddr_t)ehdr,
1040             sizeof (Ehdr), (offset_t)0, UIO_SYSSPACE, 0,
1041             (rlim64_t)0, credp, &resid)) != 0)
1042                 return (error);

1044         /*
1045          * Since a separate version is compiled for handling 32-bit and
1046          * 64-bit ELF executables on a 64-bit kernel, the 64-bit version
1047          * doesn't need to be able to deal with 32-bit ELF files.
1048          */
1049         if (resid != 0 ||
1050             ehdr->e_ident[EI_MAG2] != ELFMAG2 ||
1051             ehdr->e_ident[EI_MAG3] != ELFMAG3)
1052                 return (ENOEXEC);

1054         if ((ehdr->e_type != ET_EXEC && ehdr->e_type != ET_DYN) ||
1055 #if defined(_ILP32) || defined(_ELF32_COMPAT)
1056             ehdr->e_ident[EI_CLASS] != ELFCLASS32 ||
1057 #else
1058             ehdr->e_ident[EI_CLASS] != ELFCLASS64 ||
1059 #endif
1060             !elfheadcheck(ehdr->e_ident[EI_DATA], ehdr->e_machine,
1061             ehdr->e_flags))
1062                 return (EINVAL);

1064         *nshdrs = ehdr->e_shnum;
1065         *shstrndx = ehdr->e_shstrndx;
1066         *nphdrs = ehdr->e_phnum;

1068         /*
1069          * If e_shnum, e_shstrndx, or e_phnum is its sentinel value, we need
1070          * to read in the section header at index zero to acces the true
1071          * values for those fields.
1072          */
1073         if ((*nshdrs == 0 && ehdr->e_shoff != 0) ||
1074             *shstrndx == SHN_XINDEX || *nphdrs == PN_XNUM) {
1075                 Shdr shdr;

1077                 if (ehdr->e_shoff == 0)
1078                         return (EINVAL);

1080                 if ((error = vn_rdwr(UIO_READ, vp, (caddr_t)&shdr,
1081                     sizeof (shdr), (offset_t)ehdr->e_shoff, UIO_SYSSPACE, 0,
1082                     (rlim64_t)0, credp, &resid)) != 0)
1083                         return (error);

1085                 if (*nshdrs == 0)
1086                         *nshdrs = shdr.sh_size;
1087                 if (*shstrndx == SHN_XINDEX)
1088                         *shstrndx = shdr.sh_link;
1089                 if (*nphdrs == PN_XNUM && shdr.sh_info != 0)
1090                         *nphdrs = shdr.sh_info;
1091         }

1093         return (0);
1094 }

1096 #ifdef _ELF32_COMPAT
1097 extern size_t elf_nphdr_max;
1098 #else
1099 size_t elf_nphdr_max = 1000;
```



```
1100 #endif

1102 static int
1103 getelfphdr(vnode_t *vp, cred_t *credp, const Ehdr *ehdr, int nphdrs,
1104     caddr_t *phbasep, ssize_t *phsizep)
1105 {
1106         ssize_t resid, minsize;
1107         int err;

1109         /*
1110          * Since we're going to be using e_phentsize to iterate down the
1111          * array of program headers, it must be 8-byte aligned or else
1112          * a we might cause a misaligned access. We use all members through
1113          * p_flags on 32-bit ELF files and p_memsz on 64-bit ELF files so
1114          * e_phentsize must be at least large enough to include those
1115          * members.
1116          */
1117 #if !defined(_LP64) || defined(_ELF32_COMPAT)
1118         minsize = offsetof(Phdr, p_flags) + sizeof (((Phdr *)NULL)->p_flags);
1119 #else
1120         minsize = offsetof(Phdr, p_memsz) + sizeof (((Phdr *)NULL)->p_memsz);
1121 #endif
1122         if (ehdr->e_phentsize < minsize || (ehdr->e_phentsize & 3))
1123                 return (EINVAL);

1125         *phsizep = nphdrs * ehdr->e_phentsize;

1127         if (*phsizep > sizeof (Phdr) * elf_nphdr_max) {
1128                 if ((*phbasep = kmem_alloc(*phsizep, KM_NOSLEEP)) == NULL)
1129                         return (ENOMEM);
1130         } else {
1131                 *phbasep = kmem_alloc(*phsizep, KM_SLEEP);
1132         }

1134         if ((err = vn_rdwr(UIO_READ, vp, *phbasep, *phsizep,
1135             (offset_t)ehdr->e_phoff, UIO_SYSSPACE, 0, (rlim64_t)0,
1136             credp, &resid)) != 0) {
1137                 kmem_free(*phbasep, *phsizep);
1138                 *phbasep = NULL;
1139                 return (err);
1140         }

1142         return (0);
1143 }

1145 #ifdef _ELF32_COMPAT
1146 extern size_t elf_nshdr_max;
1147 extern size_t elf_shstrtab_max;
1148 #else
1149 size_t elf_nshdr_max = 10000;
1150 size_t elf_shstrtab_max = 100 * 1024;
1151 #endif


1154 static int
1155 getelfshdr(vnode_t *vp, cred_t *credp, const Ehdr *ehdr,
1156     int nshdrs, int shstrndx, caddr_t *shbasep, ssize_t *shsizep,
1157     char **shstrbasep, ssize_t *shstrsizep)
1158 {
1159         ssize_t resid, minsize;
1160         int err;
1161         Shdr *shdr;

1163         /*
1164          * Since we're going to be using e_shentsize to iterate down the
1165          * array of section headers, it must be 8-byte aligned or else
```

```
1166          * a we might cause a misaligned access. We use all members through
1167          * sh_entsize (on both 32- and 64-bit ELF files) so e_shentsize
1168          * must be at least large enough to include that member. The index
1169          * of the string table section must also be valid.
1170          */
1171         minsize = offsetof(Shdr, sh_entsize) + sizeof (shdr->sh_entsize);
1172         if (ehdr->e_shentsize < minsize || (ehdr->e_shentsize & 3) ||
1173             shstrndx >= nshdrs)
1174                 return (EINVAL);

1176         *shsizep = nshdrs * ehdr->e_shentsize;

1178         if (*shsizep > sizeof (Shdr) * elf_nshdr_max) {
1179                 if ((*shbasep = kmem_alloc(*shsizep, KM_NOSLEEP)) == NULL)
1180                         return (ENOMEM);
1181         } else {
1182                 *shbasep = kmem_alloc(*shsizep, KM_SLEEP);
1183         }

1185         if ((err = vn_rdwr(UIO_READ, vp, *shbasep, *shsizep,
1186             (offset_t)ehdr->e_shoff, UIO_SYSSPACE, 0, (rlim64_t)0,
1187             credp, &resid)) != 0) {
1188                 kmem_free(*shbasep, *shsizep);
1189                 return (err);
1190         }

1192         /*
1193          * Pull the section string table out of the vnode; fail if the size
1194          * is zero.
1195          */
1196         shdr = (Shdr *)(*shbasep + shstrndx * ehdr->e_shentsize);
1197         if ((*shstrsizep = shdr->sh_size) == 0) {
1198                 kmem_free(*shbasep, *shsizep);
1199                 return (EINVAL);
1200         }

1202         if (*shstrsizep > elf_shstrtab_max) {
1203                 if ((*shstrbasep = kmem_alloc(*shstrsizep,
1204                     KM_NOSLEEP)) == NULL) {
1205                         kmem_free(*shbasep, *shsizep);
1206                         return (ENOMEM);
1207                 }
1208         } else {
1209                 *shstrbasep = kmem_alloc(*shstrsizep, KM_SLEEP);
1210         }

1212         if ((err = vn_rdwr(UIO_READ, vp, *shstrbasep, *shstrsizep,
1213             (offset_t)shdr->sh_offset, UIO_SYSSPACE, 0, (rlim64_t)0,
1214             credp, &resid)) != 0) {
1215                 kmem_free(*shbasep, *shsizep);
1216                 kmem_free(*shstrbasep, *shstrsizep);
1217                 return (err);
1218         }

1220         /*
1221          * Make sure the strtab is null-terminated to make sure we
1222          * don't run off the end of the table.
1223          */
1224         (*shstrbasep)[*shstrsizep - 1] = '\0';

1226         return (0);
1227 }

1229 static int
1230 mapelfexec(
1231         vnode_t *vp,
```



```
1232         Ehdr *ehdr,
1233         int nphdrs,
1234         caddr_t phdrbase,
1235         Phdr **uphdr,
1236         Phdr **intphdr,
 481         Phdr **dyphdr,
1237         Phdr **stphdr,
1238         Phdr **dtphdr,
1239         Phdr *dataphdrp,
1240         caddr_t *bssbase,
1241         caddr_t *brkbase,
1242         intptr_t *voffset,
1243         intptr_t *minaddr,
1244         size_t len,
1245         long *execsz,
1246         size_t *brksize)
1247 {
1248         Phdr *phdr;
1249         int i, prot, error;
1250         caddr_t addr = NULL;
1251         size_t zfodsz;
1252         int ptload = 0;
1253         int page;
1254         off_t offset;
1255         int hsize = ehdr->e_phentsize;
1256         caddr_t mintmp = (caddr_t)-1;
1257         extern int use_brk_lpg;

1259         if (ehdr->e_type == ET_DYN) {
1260                 uint_t flags = 0;
1261 #endif /* ! codereview */
1262                 /*
1263                  * Obtain the virtual address of a hole in the
1264                  * address space to map the "interpreter".
1265                  */
1266                 if (secflag_enabled(curproc, PROC_SEC_ASLR))
1267                         flags |= _MAP_RANDOMIZE;

1269                 map_addr(&addr, len, (offset_t)0, 1, flags);
 505                 map_addr(&addr, len, (offset_t)0, 1, 0);
1270                 if (addr == NULL)
1271                         return (ENOMEM);
1272                 *voffset = (intptr_t)addr;

1274                 /*
1275                  * Calculate the minimum vaddr so it can be subtracted out.
1276                  * According to the ELF specification, since PT_LOAD sections
1277                  * must be sorted by increasing p_vaddr values, this is
1278                  * guaranteed to be the first PT_LOAD section.
1279                  */
1280                 phdr = (Phdr *)phdrbase;
1281                 for (i = nphdrs; i > 0; i--) {
1282                         if (phdr->p_type == PT_LOAD) {
1283                                 *voffset -= (uintptr_t)phdr->p_vaddr;
1284                                 break;
1285                         }
1286                         phdr = (Phdr *)((caddr_t)phdr + hsize);
1287                 }

1289         } else {
1290                 *voffset = 0;
1291         }
1292         phdr = (Phdr *)phdrbase;
1293         for (i = nphdrs; i > 0; i--) {
1294                 switch (phdr->p_type) {
1295                 case PT_LOAD:
```

```
1296                        if ((*intphdr != NULL) && (*uphdr == NULL))
 532                        if ((*dyphdr != NULL) && (*uphdr == NULL))
1297                                return (0);

1299                        ptload = 1;
1300                        prot = PROT_USER;
1301                        if (phdr->p_flags & PF_R)
1302                                prot |= PROT_READ;
1303                        if (phdr->p_flags & PF_W)
1304                                prot |= PROT_WRITE;
1305                        if (phdr->p_flags & PF_X)
1306                                prot |= PROT_EXEC;

1308                        addr = (caddr_t)((uintptr_t)phdr->p_vaddr + *voffset);

1310                        /*
1311                         * Keep track of the segment with the lowest starting
1312                         * address.
1313                         */
1314                        if (addr < mintmp)
1315                                mintmp = addr;

1317                        zfodsz = (size_t)phdr->p_memsz - phdr->p_filesz;

1319                        offset = phdr->p_offset;
1320                        if (((uintptr_t)offset & PAGEOFFSET) ==
1321                            ((uintptr_t)addr & PAGEOFFSET) &&
1322                            (!(vp->v_flag & VNOMAP))) {
1323                                page = 1;
1324                        } else {
1325                                page = 0;
1326                        }

1328                        /*
1329                         * Set the heap pagesize for OOB when the bss size
1330                         * is known and use_brk_lpg is not 0.
1331                         */
1332                        if (brksize != NULL && use_brk_lpg &&
1333                            zfodsz != 0 && phdr == dataphdrp &&
1334                            (prot & PROT_WRITE)) {
1335                                size_t tlen = P2NPHASE((uintptr_t)addr +
1336                                    phdr->p_filesz, PAGESIZE);

1338                                if (zfodsz > tlen) {
1339                                        curproc->p_brkpageszc =
1340                                            page_szc(map_pgsz(MAPPGSZ_HEAP,
1341                                            curproc, addr + phdr->p_filesz +
1342                                            tlen, zfodsz - tlen, 0));
1343                                }
1344                        }

1346                        if (curproc->p_brkpageszc != 0 && phdr == dataphdrp &&
1347                            (prot & PROT_WRITE)) {
1348                                uint_t  szc = curproc->p_brkpageszc;
1349                                size_t pgsz = page_get_pagesize(szc);
1350                                caddr_t ebss = addr + phdr->p_memsz;
1351                                /*
1352                                 * If we need extra space to keep the BSS an
1353                                 * integral number of pages in size, some of
1354                                 * that space may fall beyond p_brkbase, so we
1355                                 * need to set p_brksize to account for it
1356                                 * being (logically) part of the brk.
1357                                 */
1358 #endif /* ! codereview */
1359                                size_t extra_zfodsz;
```



```
1361                                ASSERT(pgsz > PAGESIZE);

1363                                extra_zfodsz = P2NPHASE((uintptr_t)ebss, pgsz);

1365                                if (error = execmap(vp, addr, phdr->p_filesz,
1366                                    zfodsz + extra_zfodsz, phdr->p_offset,
1367                                    prot, page, szc))
1368                                        goto bad;
1369                                if (brksize != NULL)
1370                                        *brksize = extra_zfodsz;
1371                        } else {
1372                                if (error = execmap(vp, addr, phdr->p_filesz,
1373                                    zfodsz, phdr->p_offset, prot, page, 0))
1374                                        goto bad;
1375                        }

1377                        if (bssbase != NULL && addr >= *bssbase &&
1378                            phdr == dataphdrp) {
1379                                *bssbase = addr + phdr->p_filesz;
1380                        }
1381                        if (brkbase != NULL && addr >= *brkbase) {
1382                                *brkbase = addr + phdr->p_memsz;
1383                        }

1385                        *execsz += btopr(phdr->p_memsz);
1386                        break;

1388                case PT_INTERP:
1389                        if (ptload)
1390                                goto bad;
1391                        *intphdr = phdr;
 587                        *dyphdr = phdr;
1392                        break;

1394                case PT_SHLIB:
1395                        *stphdr = phdr;
1396                        break;

1398                case PT_PHDR:
1399                        if (ptload)
1400                                goto bad;
1401                        *uphdr = phdr;
1402                        break;

1404                case PT_NULL:
1405                case PT_DYNAMIC:
1406                case PT_NOTE:
1407                        break;

1409                case PT_SUNWDTRACE:
1410                        if (dtphdr != NULL)
1411                                *dtphdr = phdr;
1412                        break;

1414                default:
1415                        break;
1416                }
1417                phdr = (Phdr *)((caddr_t)phdr + hsize);
1418        }

1420        if (minaddr != NULL) {
1421                ASSERT(mintmp != (caddr_t)-1);
1422                *minaddr = (intptr_t)mintmp;
1423        }

1425        if (brkbase != NULL && secflag_enabled(curproc, PROC_SEC_ASLR)) {
```

```
1426                 size_t off;
1427                 uintptr_t base = (uintptr_t)*brkbase;
1428                 uintptr_t oend = base + *brksize;

1430                 ASSERT(ISP2(aslr_max_brk_skew));

1432                 (void) random_get_pseudo_bytes((uint8_t *)&off, sizeof (off));
1433                 base += P2PHASE(off, aslr_max_brk_skew);
1434                 base = P2ROUNDUP(base, PAGESIZE);
1435                 *brkbase = (caddr_t)base;
1436                 /*
1437                  * Above, we set *brksize to account for the possibility we
1438                  * had to grow the 'brk' in padding out the BSS to a page
1439                  * boundary.
1440                  *
1441                  * We now need to adjust that based on where we now are
1442                  * actually putting the brk.
1443                  */
1444                 if (oend > base)
1445                         *brksize = oend - base;
1446                 else
1447                         *brksize = 0;
1448         }

1450 #endif /* ! codereview */
1451         return (0);
1452 bad:
1453         if (error == 0)
1454                 error = EINVAL;
1455         return (error);
1456 }

1458 int
1459 elfnote(vnode_t *vp, offset_t *offsetp, int type, int descsz, void *desc,
1460     rlim64_t rlimit, cred_t *credp)
1461 {
1462         Note note;
1463         int error;

1465         bzero(&note, sizeof (note));
1466         bcopy("CORE", note.name, 4);
1467         note.nhdr.n_type = type;
1468         /*
1469          * The System V ABI states that n_namesz must be the length of the
1470          * string that follows the Nhdr structure including the terminating
1471          * null. The ABI also specifies that sufficient padding should be
1472          * included so that the description that follows the name string
1473          * begins on a 4- or 8-byte boundary for 32- and 64-bit binaries
1474          * respectively. However, since this change was not made correctly
1475          * at the time of the 64-bit port, both 32- and 64-bit binaries
1476          * descriptions are only guaranteed to begin on a 4-byte boundary.
1477          */
1478         note.nhdr.n_namesz = 5;
1479         note.nhdr.n_descsz = roundup(descsz, sizeof (Word));

1481         if (error = core_write(vp, UIO_SYSSPACE, *offsetp, &note,
1482             sizeof (note), rlimit, credp))
1483                 return (error);

1485         *offsetp += sizeof (note);

1487         if (error = core_write(vp, UIO_SYSSPACE, *offsetp, desc,
1488             note.nhdr.n_descsz, rlimit, credp))
1489                 return (error);

1491         *offsetp += note.nhdr.n_descsz;
```



```
1492         return (0);
1493 }

1495 /*
1496  * Copy the section data from one vnode to the section of another vnode.
1497  */
1498 static void
1499 copy_scn(Shdr *src, vnode_t *src_vp, Shdr *dst, vnode_t *dst_vp, Off *doffset,
1500     void *buf, size_t size, cred_t *credp, rlim64_t rlimit)
1501 {
1502         ssize_t resid;
1503         size_t len, n = src->sh_size;
1504         offset_t off = 0;

1506         while (n != 0) {
1507                 len = MIN(size, n);
1508                 if (vn_rdwr(UIO_READ, src_vp, buf, len, src->sh_offset + off,
1509                     UIO_SYSSPACE, 0, (rlim64_t)0, credp, &resid) != 0 ||
1510                     resid >= len ||
1511                     core_write(dst_vp, UIO_SYSSPACE, *doffset + off,
1512                     buf, len - resid, rlimit, credp) != 0) {
1513                         dst->sh_size = 0;
1514                         dst->sh_offset = 0;
1515                         return;
1516                 }

1518                 ASSERT(n >= len - resid);

1520                 n -= len - resid;
1521                 off += len - resid;
1522         }

1524         *doffset += src->sh_size;
1525 }

1527 #ifdef _ELF32_COMPAT
1528 extern size_t elf_datasz_max;
1529 #else
1530 size_t elf_datasz_max = 1 * 1024 * 1024;
1531 #endif

1533 /*
1534  * This function processes mappings that correspond to load objects to
1535  * examine their respective sections for elfcore(). It's called once with
1536  * v set to NULL to count the number of sections that we're going to need
1537  * and then again with v set to some allocated buffer that we fill in with
1538  * all the section data.
1539  */
1540 static int
1541 process_scns(core_content_t content, proc_t *p, cred_t *credp, vnode_t *vp,
1542     Shdr *v, int nv, rlim64_t rlimit, Off *doffsetp, int *nshdrsp)
1543 {
1544         vnode_t *lastvp = NULL;
1545         struct seg *seg;
1546         int i, j;
1547         void *data = NULL;
1548         size_t datasz = 0;
1549         shstrtab_t shstrtab;
1550         struct as *as = p->p_as;
1551         int error = 0;

1553         if (v != NULL)
1554                 shstrtab_init(&shstrtab);

1556         i = 1;
1557         for (seg = AS_SEGFIRST(as); seg != NULL; seg = AS_SEGNEXT(as, seg)) {
```

```
1558                uint_t prot;
1559                vnode_t *mvp;
1560                void *tmp = NULL;
1561                caddr_t saddr = seg->s_base;
1562                caddr_t naddr;
1563                caddr_t eaddr;
1564                size_t segsize;

1566                Ehdr ehdr;
1567                int nshdrs, shstrndx, nphdrs;
1568                caddr_t shbase;
1569                ssize_t shsize;
1570                char *shstrbase;
1571                ssize_t shstrsize;

1573                Shdr *shdr;
1574                const char *name;
1575                size_t sz;
1576                uintptr_t off;

1578                int ctf_ndx = 0;
1579                int symtab_ndx = 0;

1581                /*
1582                 * Since we're just looking for text segments of load
1583                 * objects, we only care about the protection bits; we don't
1584                 * care about the actual size of the segment so we use the
1585                 * reserved size. If the segment's size is zero, there's
1586                 * something fishy going on so we ignore this segment.
1587                 */
1588                if (seg->s_ops != &segvn_ops ||
1589                    SEGOP_GETVP(seg, seg->s_base, &mvp) != 0 ||
1590                    mvp == lastvp || mvp == NULL || mvp->v_type != VREG ||
1591                    (segsize = pr_getsegsize(seg, 1)) == 0)
1592                        continue;

1594                eaddr = saddr + segsize;
1595                prot = pr_getprot(seg, 1, &tmp, &saddr, &naddr, eaddr);
1596                pr_getprot_done(&tmp);

1598                /*
1599                 * Skip this segment unless the protection bits look like
1600                 * what we'd expect for a text segment.
1601                 */
1602                if ((prot & (PROT_WRITE | PROT_EXEC)) != PROT_EXEC)
1603                        continue;

1605                if (getelfhead(mvp, credp, &ehdr, &nshdrs, &shstrndx,
1606                    &nphdrs) != 0 ||
1607                    getelfshdr(mvp, credp, &ehdr, nshdrs, shstrndx,
1608                    &shbase, &shsize, &shstrbase, &shstrsize) != 0)
1609                        continue;

1611                off = ehdr.e_shentsize;
1612                for (j = 1; j < nshdrs; j++, off += ehdr.e_shentsize) {
1613                        Shdr *symtab = NULL, *strtab;

1615                        shdr = (Shdr *)(shbase + off);

1617                        if (shdr->sh_name >= shstrsize)
1618                                continue;

1620                        name = shstrbase + shdr->sh_name;

1622                        if (strcmp(name, shstrtab_data[STR_CTF]) == 0) {
1623                                if ((content & CC_CONTENT_CTF) == 0 ||
```



```
1624                                    ctf_ndx != 0)
1625                                        continue;

1627                                if (shdr->sh_link > 0 &&
1628                                    shdr->sh_link < nshdrs) {
1629                                        symtab = (Shdr *)(shbase +
1630                                            shdr->sh_link * ehdr.e_shentsize);
1631                                }

1633                                if (v != NULL && i < nv - 1) {
1634                                        if (shdr->sh_size > datasz &&
1635                                            shdr->sh_size <= elf_datasz_max) {
1636                                                if (data != NULL)
1637                                                        kmem_free(data, datasz);

1639                                                datasz = shdr->sh_size;
1640                                                data = kmem_alloc(datasz,
1641                                                    KM_SLEEP);
1642                                        }

1644                                        v[i].sh_name = shstrtab_ndx(&shstrtab,
1645                                            STR_CTF);
1646                                        v[i].sh_addr = (Addr)(uintptr_t)saddr;
1647                                        v[i].sh_type = SHT_PROGBITS;
1648                                        v[i].sh_addralign = 4;
1649                                        *doffsetp = roundup(*doffsetp,
1650                                            v[i].sh_addralign);
1651                                        v[i].sh_offset = *doffsetp;
1652                                        v[i].sh_size = shdr->sh_size;
1653                                        if (symtab == NULL)  {
1654                                                v[i].sh_link = 0;
1655                                        } else if (symtab->sh_type ==
1656                                            SHT_SYMTAB &&
1657                                            symtab_ndx != 0) {
1658                                                v[i].sh_link =
1659                                                    symtab_ndx;
1660                                        } else {
1661                                                v[i].sh_link = i + 1;
1662                                        }

1664                                        copy_scn(shdr, mvp, &v[i], vp,
1665                                            doffsetp, data, datasz, credp,
1666                                            rlimit);
1667                                }

1669                                ctf_ndx = i++;

1671                                /*
1672                                 * We've already dumped the symtab.
1673                                 */
1674                                if (symtab != NULL &&
1675                                    symtab->sh_type == SHT_SYMTAB &&
1676                                    symtab_ndx != 0)
1677                                        continue;

1679                        } else if (strcmp(name,
1680                            shstrtab_data[STR_SYMTAB]) == 0) {
1681                                if ((content & CC_CONTENT_SYMTAB) == 0 ||
1682                                    symtab != 0)
1683                                        continue;

1685                                symtab = shdr;
1686                        }

1688                        if (symtab != NULL) {
1689                                if ((symtab->sh_type != SHT_DYNSYM &&
```

```
1690                                    symtab->sh_type != SHT_SYMTAB) ||
1691                                    symtab->sh_link == 0 ||
1692                                    symtab->sh_link >= nshdrs)
1693                                        continue;

1695                                strtab = (Shdr *)(shbase +
1696                                    symtab->sh_link * ehdr.e_shentsize);

1698                                if (strtab->sh_type != SHT_STRTAB)
1699                                        continue;

1701                                if (v != NULL && i < nv - 2) {
1702                                        sz = MAX(symtab->sh_size,
1703                                            strtab->sh_size);
1704                                        if (sz > datasz &&
1705                                            sz <= elf_datasz_max) {
1706                                                if (data != NULL)
1707                                                        kmem_free(data, datasz);

1709                                                datasz = sz;
1710                                                data = kmem_alloc(datasz,
1711                                                    KM_SLEEP);
1712                                        }

1714                                        if (symtab->sh_type == SHT_DYNSYM) {
1715                                                v[i].sh_name = shstrtab_ndx(
1716                                                    &shstrtab, STR_DYNSYM);
1717                                                v[i + 1].sh_name = shstrtab_ndx(
1718                                                    &shstrtab, STR_DYNSTR);
1719                                        } else {
1720                                                v[i].sh_name = shstrtab_ndx(
1721                                                    &shstrtab, STR_SYMTAB);
1722                                                v[i + 1].sh_name = shstrtab_ndx(
1723                                                    &shstrtab, STR_STRTAB);
1724                                        }

1726                                        v[i].sh_type = symtab->sh_type;
1727                                        v[i].sh_addr = symtab->sh_addr;
1728                                        if (ehdr.e_type == ET_DYN ||
1729                                            v[i].sh_addr == 0)
1730                                                v[i].sh_addr +=
1731                                                    (Addr)(uintptr_t)saddr;
1732                                        v[i].sh_addralign =
1733                                            symtab->sh_addralign;
1734                                        *doffsetp = roundup(*doffsetp,
1735                                            v[i].sh_addralign);
1736                                        v[i].sh_offset = *doffsetp;
1737                                        v[i].sh_size = symtab->sh_size;
1738                                        v[i].sh_link = i + 1;
1739                                        v[i].sh_entsize = symtab->sh_entsize;
1740                                        v[i].sh_info = symtab->sh_info;

1742                                        copy_scn(symtab, mvp, &v[i], vp,
1743                                            doffsetp, data, datasz, credp,
1744                                            rlimit);

1746                                        v[i + 1].sh_type = SHT_STRTAB;
1747                                        v[i + 1].sh_flags = SHF_STRINGS;
1748                                        v[i + 1].sh_addr = symtab->sh_addr;
1749                                        if (ehdr.e_type == ET_DYN ||
1750                                            v[i + 1].sh_addr == 0)
1751                                                v[i + 1].sh_addr +=
1752                                                    (Addr)(uintptr_t)saddr;
1753                                        v[i + 1].sh_addralign =
1754                                            strtab->sh_addralign;
1755                                        *doffsetp = roundup(*doffsetp,
```



```
1756                                            v[i + 1].sh_addralign);
1757                                        v[i + 1].sh_offset = *doffsetp;
1758                                        v[i + 1].sh_size = strtab->sh_size;

1760                                        copy_scn(strtab, mvp, &v[i + 1], vp,
1761                                            doffsetp, data, datasz, credp,
1762                                            rlimit);
1763                                }

1765                                if (symtab->sh_type == SHT_SYMTAB)
1766                                        symtab_ndx = i;
1767                                i += 2;
1768                        }
1769                }

1771                kmem_free(shstrbase, shstrsize);
1772                kmem_free(shbase, shsize);

1774                lastvp = mvp;
1775        }

1777        if (v == NULL) {
1778                if (i == 1)
1779                        *nshdrsp = 0;
1780                else
1781                        *nshdrsp = i + 1;
1782                goto done;
1783        }

1785        if (i != nv - 1) {
1786                cmn_err(CE_WARN, "elfcore: core dump failed for "
1787                    "process %d; address space is changing", p->p_pid);
1788                error = EIO;
1789                goto done;
1790        }

1792        v[i].sh_name = shstrtab_ndx(&shstrtab, STR_SHSTRTAB);
1793        v[i].sh_size = shstrtab_size(&shstrtab);
1794        v[i].sh_addralign = 1;
1795        *doffsetp = roundup(*doffsetp, v[i].sh_addralign);
1796        v[i].sh_offset = *doffsetp;
1797        v[i].sh_flags = SHF_STRINGS;
1798        v[i].sh_type = SHT_STRTAB;

1800        if (v[i].sh_size > datasz) {
1801                if (data != NULL)
1802                        kmem_free(data, datasz);

1804                datasz = v[i].sh_size;
1805                data = kmem_alloc(datasz,
1806                    KM_SLEEP);
1807        }

1809        shstrtab_dump(&shstrtab, data);

1811        if ((error = core_write(vp, UIO_SYSSPACE, *doffsetp,
1812            data, v[i].sh_size, rlimit, credp)) != 0)
1813                goto done;

1815        *doffsetp += v[i].sh_size;

1817 done:
1818        if (data != NULL)
1819                kmem_free(data, datasz);

1821        return (error);
```

```
1822 }

1824 int
1825 elfcore(vnode_t *vp, proc_t *p, cred_t *credp, rlim64_t rlimit, int sig,
1826     core_content_t content)
1827 {
1828         offset_t poffset, soffset;
1829         Off doffset;
1830         int error, i, nphdrs, nshdrs;
1831         int overflow = 0;
1832         struct seg *seg;
1833         struct as *as = p->p_as;
1834         union {
1835                 Ehdr ehdr;
1836                 Phdr phdr[1];
1837                 Shdr shdr[1];
1838         } *bigwad;
1839         size_t bigsize;
1840         size_t phdrsz, shdrsz;
1841         Ehdr *ehdr;
1842         Phdr *v;
1843         caddr_t brkbase;
1844         size_t brksize;
1845         caddr_t stkbase;
1846         size_t stksize;
1847         int ntries = 0;
1848         klwp_t *lwp = ttolwp(curthread);

1850 top:
1851         /*
1852          * Make sure we have everything we need (registers, etc.).
1853          * All other lwps have already stopped and are in an orderly state.
1854          */
1855         ASSERT(p == ttoproc(curthread));
1856         prstop(0, 0);

1858         AS_LOCK_ENTER(as, &as->a_lock, RW_WRITER);
1859         nphdrs = prnsegs(as, 0) + 2;            /* two CORE note sections */

1861         /*
1862          * Count the number of section headers we're going to need.
1863          */
1864         nshdrs = 0;
1865         if (content & (CC_CONTENT_CTF | CC_CONTENT_SYMTAB)) {
1866                 (void) process_scns(content, p, credp, NULL, NULL, NULL, 0,
1867                     NULL, &nshdrs);
1868         }
1869         AS_LOCK_EXIT(as, &as->a_lock);

1871         ASSERT(nshdrs == 0 || nshdrs > 1);

1873         /*
1874          * The core file contents may required zero section headers, but if
1875          * we overflow the 16 bits allotted to the program header count in
1876          * the ELF header, we'll need that program header at index zero.
1877          */
1878         if (nshdrs == 0 && nphdrs >= PN_XNUM)
1879                 nshdrs = 1;

1881         phdrsz = nphdrs * sizeof (Phdr);
1882         shdrsz = nshdrs * sizeof (Shdr);

1884         bigsize = MAX(sizeof (*bigwad), MAX(phdrsz, shdrsz));
1885         bigwad = kmem_alloc(bigsize, KM_SLEEP);

1887         ehdr = &bigwad->ehdr;
```



```
1888         bzero(ehdr, sizeof (*ehdr));

1890         ehdr->e_ident[EI_MAG0] = ELFMAG0;
1891         ehdr->e_ident[EI_MAG1] = ELFMAG1;
1892         ehdr->e_ident[EI_MAG2] = ELFMAG2;
1893         ehdr->e_ident[EI_MAG3] = ELFMAG3;
1894         ehdr->e_ident[EI_CLASS] = ELFCLASS;
1895         ehdr->e_type = ET_CORE;

1897 #if !defined(_LP64) || defined(_ELF32_COMPAT)

1899 #if defined(__sparc)
1900         ehdr->e_ident[EI_DATA] = ELFDATA2MSB;
1901         ehdr->e_machine = EM_SPARC;
1902 #elif defined(__i386) || defined(__i386_COMPAT)
1903         ehdr->e_ident[EI_DATA] = ELFDATA2LSB;
1904         ehdr->e_machine = EM_386;
1905 #else
1906 #error "no recognized machine type is defined"
1907 #endif

1909 #else   /* !defined(_LP64) || defined(_ELF32_COMPAT) */

1911 #if defined(__sparc)
1912         ehdr->e_ident[EI_DATA] = ELFDATA2MSB;
1913         ehdr->e_machine = EM_SPARCV9;
1914 #elif defined(__amd64)
1915         ehdr->e_ident[EI_DATA] = ELFDATA2LSB;
1916         ehdr->e_machine = EM_AMD64;
1917 #else
1918 #error "no recognized 64-bit machine type is defined"
1919 #endif

1921 #endif  /* !defined(_LP64) || defined(_ELF32_COMPAT) */

1923         /*
1924          * If the count of program headers or section headers or the index
1925          * of the section string table can't fit in the mere 16 bits
1926          * shortsightedly allotted to them in the ELF header, we use the
1927          * extended formats and put the real values in the section header
1928          * as index 0.
1929          */
1930         ehdr->e_version = EV_CURRENT;
1931         ehdr->e_ehsize = sizeof (Ehdr);

1933         if (nphdrs >= PN_XNUM)
1934                 ehdr->e_phnum = PN_XNUM;
1935         else
1936                 ehdr->e_phnum = (unsigned short)nphdrs;

1938         ehdr->e_phoff = sizeof (Ehdr);
1939         ehdr->e_phentsize = sizeof (Phdr);

1941         if (nshdrs > 0) {
1942                 if (nshdrs >= SHN_LORESERVE)
1943                         ehdr->e_shnum = 0;
1944                 else
1945                         ehdr->e_shnum = (unsigned short)nshdrs;

1947                 if (nshdrs - 1 >= SHN_LORESERVE)
1948                         ehdr->e_shstrndx = SHN_XINDEX;
1949                 else
1950                         ehdr->e_shstrndx = (unsigned short)(nshdrs - 1);

1952                 ehdr->e_shoff = ehdr->e_phoff + ehdr->e_phentsize * nphdrs;
1953                 ehdr->e_shentsize = sizeof (Shdr);
```

```
1954        }

1956        if (error = core_write(vp, UIO_SYSSPACE, (offset_t)0, ehdr,
1957            sizeof (Ehdr), rlimit, credp))
1958                goto done;

1960        poffset = sizeof (Ehdr);
1961        soffset = sizeof (Ehdr) + phdrsz;
1962        doffset = sizeof (Ehdr) + phdrsz + shdrsz;

1964        v = &bigwad->phdr[0];
1965        bzero(v, phdrsz);

1967        setup_old_note_header(&v[0], p);
1968        v[0].p_offset = doffset = roundup(doffset, sizeof (Word));
1969        doffset += v[0].p_filesz;

1971        setup_note_header(&v[1], p);
1972        v[1].p_offset = doffset = roundup(doffset, sizeof (Word));
1973        doffset += v[1].p_filesz;

1975        mutex_enter(&p->p_lock);

1977        brkbase = p->p_brkbase;
1978        brksize = p->p_brksize;

1980        stkbase = p->p_usrstack - p->p_stksize;
1981        stksize = p->p_stksize;

1983        mutex_exit(&p->p_lock);

1985        AS_LOCK_ENTER(as, &as->a_lock, RW_WRITER);
1986        i = 2;
1987        for (seg = AS_SEGFIRST(as); seg != NULL; seg = AS_SEGNEXT(as, seg)) {
1988                caddr_t eaddr = seg->s_base + pr_getsegsize(seg, 0);
1989                caddr_t saddr, naddr;
1990                void *tmp = NULL;
1991                extern struct seg_ops segspt_shmops;

1993                for (saddr = seg->s_base; saddr < eaddr; saddr = naddr) {
1994                        uint_t prot;
1995                        size_t size;
1996                        int type;
1997                        vnode_t *mvp;

1999                        prot = pr_getprot(seg, 0, &tmp, &saddr, &naddr, eaddr);
2000                        prot &= PROT_READ | PROT_WRITE | PROT_EXEC;
2001                        if ((size = (size_t)(naddr - saddr)) == 0)
2002                                continue;
2003                        if (i == nphdrs) {
2004                                overflow++;
2005                                continue;
2006                        }
2007                        v[i].p_type = PT_LOAD;
2008                        v[i].p_vaddr = (Addr)(uintptr_t)saddr;
2009                        v[i].p_memsz = size;
2010                        if (prot & PROT_READ)
2011                                v[i].p_flags |= PF_R;
2012                        if (prot & PROT_WRITE)
2013                                v[i].p_flags |= PF_W;
2014                        if (prot & PROT_EXEC)
2015                                v[i].p_flags |= PF_X;

2017                        /*
2018                         * Figure out which mappings to include in the core.
2019                         */
```



```
2020                        type = SEGOP_GETTYPE(seg, saddr);

2022                        if (saddr == stkbase && size == stksize) {
2023                                if (!(content & CC_CONTENT_STACK))
2024                                        goto exclude;

2026                        } else if (saddr == brkbase && size == brksize) {
2027                                if (!(content & CC_CONTENT_HEAP))
2028                                        goto exclude;

2030                        } else if (seg->s_ops == &segspt_shmops) {
2031                                if (type & MAP_NORESERVE) {
2032                                        if (!(content & CC_CONTENT_DISM))
2033                                                goto exclude;
2034                                } else {
2035                                        if (!(content & CC_CONTENT_ISM))
2036                                                goto exclude;
2037                                }

2039                        } else if (seg->s_ops != &segvn_ops) {
2040                                goto exclude;

2042                        } else if (type & MAP_SHARED) {
2043                                if (shmgetid(p, saddr) != SHMID_NONE) {
2044                                        if (!(content & CC_CONTENT_SHM))
2045                                                goto exclude;

2047                                } else if (SEGOP_GETVP(seg, seg->s_base,
2048                                    &mvp) != 0 || mvp == NULL ||
2049                                    mvp->v_type != VREG) {
2050                                        if (!(content & CC_CONTENT_SHANON))
2051                                                goto exclude;

2053                                } else {
2054                                        if (!(content & CC_CONTENT_SHFILE))
2055                                                goto exclude;
2056                                }

2058                        } else if (SEGOP_GETVP(seg, seg->s_base, &mvp) != 0 ||
2059                            mvp == NULL || mvp->v_type != VREG) {
2060                                if (!(content & CC_CONTENT_ANON))
2061                                        goto exclude;

2063                        } else if (prot == (PROT_READ | PROT_EXEC)) {
2064                                if (!(content & CC_CONTENT_TEXT))
2065                                        goto exclude;

2067                        } else if (prot == PROT_READ) {
2068                                if (!(content & CC_CONTENT_RODATA))
2069                                        goto exclude;

2071                        } else {
2072                                if (!(content & CC_CONTENT_DATA))
2073                                        goto exclude;
2074                        }

2076                        doffset = roundup(doffset, sizeof (Word));
2077                        v[i].p_offset = doffset;
2078                        v[i].p_filesz = size;
2079                        doffset += size;
2080 exclude:
2081                        i++;
2082                }
2083                ASSERT(tmp == NULL);
2084        }
2085        AS_LOCK_EXIT(as, &as->a_lock);
```

```
2087         if (overflow || i != nphdrs) {
2088                 if (ntries++ == 0) {
2089                         kmem_free(bigwad, bigsize);
2090                         overflow = 0;
2091                         goto top;
2092                 }
2093                 cmn_err(CE_WARN, "elfcore: core dump failed for "
2094                     "process %d; address space is changing", p->p_pid);
2095                 error = EIO;
2096                 goto done;
2097         }

2099         if ((error = core_write(vp, UIO_SYSSPACE, poffset,
2100             v, phdrsz, rlimit, credp)) != 0)
2101                 goto done;

2103         if ((error = write_old_elfnotes(p, sig, vp, v[0].p_offset, rlimit,
2104             credp)) != 0)
2105                 goto done;

2107         if ((error = write_elfnotes(p, sig, vp, v[1].p_offset, rlimit,
2108             credp, content)) != 0)
2109                 goto done;

2111         for (i = 2; i < nphdrs; i++) {
2112                 prkillinfo_t killinfo;
2113                 sigqueue_t *sq;
2114                 int sig, j;

2116                 if (v[i].p_filesz == 0)
2117                         continue;

2119                 /*
2120                  * If dumping out this segment fails, rather than failing
2121                  * the core dump entirely, we reset the size of the mapping
2122                  * to zero to indicate that the data is absent from the core
2123                  * file and or in the PF_SUNW_FAILURE flag to differentiate
2124                  * this from mappings that were excluded due to the core file
2125                  * content settings.
2126                  */
2127                 if ((error = core_seg(p, vp, v[i].p_offset,
2128                     (caddr_t)(uintptr_t)v[i].p_vaddr, v[i].p_filesz,
2129                     rlimit, credp)) == 0) {
2130                         continue;
2131                 }

2133                 if ((sig = lwp->lwp_cursig) == 0) {
2134                         /*
2135                          * We failed due to something other than a signal.
2136                          * Since the space reserved for the segment is now
2137                          * unused, we stash the errno in the first four
2138                          * bytes. This undocumented interface will let us
2139                          * understand the nature of the failure.
2140                          */
2141                         (void) core_write(vp, UIO_SYSSPACE, v[i].p_offset,
2142                             &error, sizeof (error), rlimit, credp);

2144                         v[i].p_filesz = 0;
2145                         v[i].p_flags |= PF_SUNW_FAILURE;
2146                         if ((error = core_write(vp, UIO_SYSSPACE,
2147                             poffset + sizeof (v[i]) * i, &v[i], sizeof (v[i]),
2148                             rlimit, credp)) != 0)
2149                                 goto done;

2151                         continue;
```



```
2152                 }

2154                 /*
2155                  * We took a signal.  We want to abort the dump entirely, but
2156                  * we also want to indicate what failed and why.  We therefore
2157                  * use the space reserved for the first failing segment to
2158                  * write our error (which, for purposes of compatability with
2159                  * older core dump readers, we set to EINTR) followed by any
2160                  * siginfo associated with the signal.
2161                  */
2162                 bzero(&killinfo, sizeof (killinfo));
2163                 killinfo.prk_error = EINTR;

2165                 sq = sig == SIGKILL ? curproc->p_killsqp : lwp->lwp_curinfo;

2167                 if (sq != NULL) {
2168                         bcopy(&sq->sq_info, &killinfo.prk_info,
2169                             sizeof (sq->sq_info));
2170                 } else {
2171                         killinfo.prk_info.si_signo = lwp->lwp_cursig;
2172                         killinfo.prk_info.si_code = SI_NOINFO;
2173                 }

2175 #if (defined(_SYSCALL32_IMPL) || defined(_LP64))
2176                 /*
2177                  * If this is a 32-bit process, we need to translate from the
2178                  * native siginfo to the 32-bit variant.  (Core readers must
2179                  * always have the same data model as their target or must
2180                  * be aware of -- and compensate for -- data model differences.)
2181                  */
2182                 if (curproc->p_model == DATAMODEL_ILP32) {
2183                         siginfo32_t si32;

2185                         siginfo_kto32((k_siginfo_t *)&killinfo.prk_info, &si32);
2186                         bcopy(&si32, &killinfo.prk_info, sizeof (si32));
2187                 }
2188 #endif

2190                 (void) core_write(vp, UIO_SYSSPACE, v[i].p_offset,
2191                     &killinfo, sizeof (killinfo), rlimit, credp);

2193                 /*
2194                  * For the segment on which we took the signal, indicate that
2195                  * its data now refers to a siginfo.
2196                  */
2197                 v[i].p_filesz = 0;
2198                 v[i].p_flags |= PF_SUNW_FAILURE | PF_SUNW_KILLED |
2199                     PF_SUNW_SIGINFO;

2201                 /*
2202                  * And for every other segment, indicate that its absence
2203                  * is due to a signal.
2204                  */
2205                 for (j = i + 1; j < nphdrs; j++) {
2206                         v[j].p_filesz = 0;
2207                         v[j].p_flags |= PF_SUNW_FAILURE | PF_SUNW_KILLED;
2208                 }

2210                 /*
2211                  * Finally, write out our modified program headers.
2212                  */
2213                 if ((error = core_write(vp, UIO_SYSSPACE,
2214                     poffset + sizeof (v[i]) * i, &v[i],
2215                     sizeof (v[i]) * (nphdrs - i), rlimit, credp)) != 0)
2216                         goto done;
```

```
2218                    break;
2219            }

2221            if (nshdrs > 0) {
2222                    bzero(&bigwad->shdr[0], shdrsz);

2224                    if (nshdrs >= SHN_LORESERVE)
2225                            bigwad->shdr[0].sh_size = nshdrs;

2227                    if (nshdrs - 1 >= SHN_LORESERVE)
2228                            bigwad->shdr[0].sh_link = nshdrs - 1;

2230                    if (nphdrs >= PN_XNUM)
2231                            bigwad->shdr[0].sh_info = nphdrs;

2233                    if (nshdrs > 1) {
2234                            AS_LOCK_ENTER(as, &as->a_lock, RW_WRITER);
2235                            if ((error = process_scns(content, p, credp, vp,
2236                                &bigwad->shdr[0], nshdrs, rlimit, &doffset,
2237                                NULL)) != 0) {
2238                                    AS_LOCK_EXIT(as, &as->a_lock);
2239                                    goto done;
2240                            }
2241                            AS_LOCK_EXIT(as, &as->a_lock);
2242                    }

2244                    if ((error = core_write(vp, UIO_SYSSPACE, soffset,
2245                        &bigwad->shdr[0], shdrsz, rlimit, credp)) != 0)
2246                            goto done;
2247            }

2249    done:
2250            kmem_free(bigwad, bigsize);
2251            return (error);
2252    }

2254    #ifndef _ELF32_COMPAT

2256    static struct execsw esw = {
2257    #ifdef  _LP64
2258            elf64magicstr,
2259    #else   /* _LP64 */
2260            elf32magicstr,
2261    #endif  /* _LP64 */
2262            0,
2263            5,
2264            elfexec,
2265            elfcore
2266    };

2268    static struct modlexec modlexec = {
2269            &mod_execops, "exec module for elf", &esw
2270    };

2272    #ifdef  _LP64
2273    extern int elf32exec(vnode_t *vp, execa_t *uap, uarg_t *args,
2274                            intpdata_t *idatap, int level, long *execsz,
2275                            int setid, caddr_t exec_file, cred_t *cred,
2276                            int brand_action);
2277    extern int elf32core(vnode_t *vp, proc_t *p, cred_t *credp,
2278                            rlim64_t rlimit, int sig, core_content_t content);

2280    static struct execsw esw32 = {
2281            elf32magicstr,
2282            0,
2283            5,
```



```
2284            elf32exec,
2285            elf32core
2286    };

2288    static struct modlexec modlexec32 = {
2289            &mod_execops, "32-bit exec module for elf", &esw32
2290    };
2291    #endif  /* _LP64 */

2293    static struct modlinkage modlinkage = {
2294            MODREV_1,
2295            (void *)&modlexec,
2296    #ifdef  _LP64
2297            (void *)&modlexec32,
2298    #endif  /* _LP64 */
2299            NULL
2300    };

2302    int
2303    _init(void)
2304    {
2305            return (mod_install(&modlinkage));
2306    }

2308    int
2309    _fini(void)
2310    {
2311            return (mod_remove(&modlinkage));
2312    }

2314    int
2315    _info(struct modinfo *modinfop)
2316    {
2317            return (mod_info(&modlinkage, modinfop));
2318    }

2320    #endif  /* !_ELF32_COMPAT */
```

```
**********************************************************
    3328 Wed May 27 19:49:26 2015
new/usr/src/uts/common/exec/elf/elf_impl.h
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
```

```
   1 /*
   2  * CDDL HEADER START
   3  *
   4  * The contents of this file are subject to the terms of the
   5  * Common Development and Distribution License (the "License").
   6  * You may not use this file except in compliance with the License.
   7  *
   8  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
   9  * or http://www.opensolaris.org/os/licensing.
  10  * See the License for the specific language governing permissions
  11  * and limitations under the License.
  12  *
  13  * When distributing Covered Code, include this CDDL HEADER in each
  14  * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  15  * If applicable, add the following below this CDDL HEADER, with the
  16  * fields enclosed by brackets "[]" replaced with your own identifying
  17  * information: Portions Copyright [yyyy] [name of copyright owner]
  18  *
  19  * CDDL HEADER END
  20  */
  21 /*
  22  * Copyright 2006 Sun Microsystems, Inc.  All rights reserved.
  23  * Use is subject to license terms.
  24  */

  26 #ifndef _ELF_ELF_IMPL_H
  27 #define _ELF_ELF_IMPL_H

  29 #pragma ident   "%Z%%M% %I%     %E% SMI"

  29 #ifdef __cplusplus
  30 extern "C" {
  31 #endif

  33 #if     !defined(_LP64) || defined(_ELF32_COMPAT)

  35 /*
  36  * Definitions for ELF32, native 32-bit or 32-bit compatibility mode.
  37  */
  38 #define ELFCLASS        ELFCLASS32
  39 typedef unsigned int    aux_val_t;
  40 typedef auxv32_t        aux_entry_t;

  42 #define USR_LIB_RTLD    "/usr/lib/ld.so.1"

  44 #else   /* !_LP64 || _ELF32_COMPAT */

  46 /*
  47  * Definitions for native 64-bit ELF
  48  */
  49 #define ELFCLASS        ELFCLASS64
  50 typedef unsigned long   aux_val_t;
  51 typedef auxv_t          aux_entry_t;

  53 /* put defines for 64-bit architectures here */
  54 #if defined(__sparcv9)
  55 #define USR_LIB_RTLD    "/usr/lib/sparcv9/ld.so.1"
  56 #endif
```



```
  58 #if defined(__amd64)
  59 #define USR_LIB_RTLD    "/usr/lib/amd64/ld.so.1"
  60 #endif

  62 #endif  /* !_LP64 || _ELF32_COMPAT */

  64 /*
  65  * Start of an ELF Note.
  66  */
  67 typedef struct {
  68         Nhdr    nhdr;
  69         char    name[8];
  70 } Note;
______unchanged_portion_omitted_
```

```
**********************************************************
  112763 Wed May 27 19:49:26 2015
new/usr/src/uts/common/fs/proc/prsubr.c
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
______unchanged_portion_omitted_

 855 /*
 856  * Return process/lwp status.
 857  * The u-block is mapped in by this routine and unmapped at the end.
 858  */
 859 void
 860 prgetstatus(proc_t *p, pstatus_t *sp, zone_t *zp)
 861 {
 862         kthread_t *t;

 864         ASSERT(MUTEX_HELD(&p->p_lock));

 866         t = prchoose(p);        /* returns locked thread */
 867         ASSERT(t != NULL);
 868         thread_unlock(t);

 870         /* just bzero the process part, prgetlwpstatus() does the rest */
 871         bzero(sp, sizeof (pstatus_t) - sizeof (lwpstatus_t));
 872         sp->pr_nlwp = p->p_lwpcnt;
 873         sp->pr_nzomb = p->p_zombcnt;
 874         prassignset(&sp->pr_sigpend, &p->p_sig);
 875         sp->pr_brkbase = (uintptr_t)p->p_brkbase;
 876         sp->pr_brksize = p->p_brksize;
 877         sp->pr_stkbase = (uintptr_t)prgetstackbase(p);
 878         sp->pr_stksize = p->p_stksize;
 879         sp->pr_pid = p->p_pid;
 880         if (curproc->p_zone->zone_id != GLOBAL_ZONEID &&
 881             (p->p_flag & SZONETOP)) {
 882                 ASSERT(p->p_zone->zone_id != GLOBAL_ZONEID);
 883                 /*
 884                  * Inside local zones, fake zsched's pid as parent pids for
 885                  * processes which reference processes outside of the zone.
 886                  */
 887                 sp->pr_ppid = curproc->p_zone->zone_zsched->p_pid;
 888         } else {
 889                 sp->pr_ppid = p->p_ppid;
 890         }
 891         sp->pr_pgid  = p->p_pgrp;
 892         sp->pr_sid   = p->p_sessp->s_sid;
 893         sp->pr_taskid = p->p_task->tk_tkid;
 894         sp->pr_projid = p->p_task->tk_proj->kpj_id;
 895         sp->pr_zoneid = p->p_zone->zone_id;
 896         bcopy(&p->p_secflags, &sp->pr_secflags, sizeof (psecflags_t));
 897 #endif /* ! codereview */
 898         hrt2ts(mstate_aggr_state(p, LMS_USER), &sp->pr_utime);
 899         hrt2ts(mstate_aggr_state(p, LMS_SYSTEM), &sp->pr_stime);
 900         TICK_TO_TIMESTRUC(p->p_cutime, &sp->pr_cutime);
 901         TICK_TO_TIMESTRUC(p->p_cstime, &sp->pr_cstime);
 902         prassignset(&sp->pr_sigtrace, &p->p_sigmask);
 903         prassignset(&sp->pr_flttrace, &p->p_fltmask);
 904         prassignset(&sp->pr_sysentry, &PTOU(p)->u_entrymask);
 905         prassignset(&sp->pr_sysexit, &PTOU(p)->u_exitmask);
 906         switch (p->p_model) {
 907         case DATAMODEL_ILP32:
 908                 sp->pr_dmodel = PR_MODEL_ILP32;
 909                 break;
 910         case DATAMODEL_LP64:
```



```
 911                 sp->pr_dmodel = PR_MODEL_LP64;
 912                 break;
 913         }
 914         if (p->p_agenttp)
 915                 sp->pr_agentid = p->p_agenttp->t_tid;

 917         /* get the chosen lwp's status */
 918         prgetlwpstatus(t, &sp->pr_lwp, zp);

 920         /* replicate the flags */
 921         sp->pr_flags = sp->pr_lwp.pr_flags;
 922 }

 924 #ifdef _SYSCALL32_IMPL
 925 void
 926 prgetlwpstatus32(kthread_t *t, lwpstatus32_t *sp, zone_t *zp)
 927 {
 928         proc_t *p = ttoproc(t);
 929         klwp_t *lwp = ttolwp(t);
 930         struct mstate *ms = &lwp->lwp_mstate;
 931         hrtime_t usr, sys;
 932         int flags;
 933         ulong_t instr;

 935         ASSERT(MUTEX_HELD(&p->p_lock));

 937         bzero(sp, sizeof (*sp));
 938         flags = 0L;
 939         if (t->t_state == TS_STOPPED) {
 940                 flags |= PR_STOPPED;
 941                 if ((t->t_schedflag & TS_PSTART) == 0)
 942                         flags |= PR_ISTOP;
 943         } else if (VSTOPPED(t)) {
 944                 flags |= PR_STOPPED|PR_ISTOP;
 945         }
 946         if (!(flags & PR_ISTOP) && (t->t_proc_flag & TP_PRSTOP))
 947                 flags |= PR_DSTOP;
 948         if (lwp->lwp_asleep)
 949                 flags |= PR_ASLEEP;
 950         if (t == p->p_agenttp)
 951                 flags |= PR_AGENT;
 952         if (!(t->t_proc_flag & TP_TWAIT))
 953                 flags |= PR_DETACH;
 954         if (t->t_proc_flag & TP_DAEMON)
 955                 flags |= PR_DAEMON;
 956         if (p->p_proc_flag & P_PR_FORK)
 957                 flags |= PR_FORK;
 958         if (p->p_proc_flag & P_PR_RUNLCL)
 959                 flags |= PR_RLC;
 960         if (p->p_proc_flag & P_PR_KILLCL)
 961                 flags |= PR_KLC;
 962         if (p->p_proc_flag & P_PR_ASYNC)
 963                 flags |= PR_ASYNC;
 964         if (p->p_proc_flag & P_PR_BPTADJ)
 965                 flags |= PR_BPTADJ;
 966         if (p->p_proc_flag & P_PR_PTRACE)
 967                 flags |= PR_PTRACE;
 968         if (p->p_flag & SMSACCT)
 969                 flags |= PR_MSACCT;
 970         if (p->p_flag & SMSFORK)
 971                 flags |= PR_MSFORK;
 972         if (p->p_flag & SVFWAIT)
 973                 flags |= PR_VFORKP;
 974         sp->pr_flags = flags;
 975         if (VSTOPPED(t)) {
 976                 sp->pr_why   = PR_REQUESTED;
```

```
977                 sp->pr_what  = 0;
978         } else {
979                 sp->pr_why   = t->t_whystop;
980                 sp->pr_what  = t->t_whatstop;
981         }
982         sp->pr_lwpid = t->t_tid;
983         sp->pr_cursig  = lwp->lwp_cursig;
984         prassignset(&sp->pr_lwppend, &t->t_sig);
985         schedctl_finish_sigblock(t);
986         prassignset(&sp->pr_lwphold, &t->t_hold);
987         if (t->t_whystop == PR_FAULTED) {
988                 siginfo_kto32(&lwp->lwp_siginfo, &sp->pr_info);
989                 if (t->t_whatstop == FLTPAGE)
990                         sp->pr_info.si_addr =
991                             (caddr32_t)(uintptr_t)lwp->lwp_siginfo.si_addr;
992         } else if (lwp->lwp_curinfo)
993                 siginfo_kto32(&lwp->lwp_curinfo->sq_info, &sp->pr_info);
994         if (SI_FROMUSER(&lwp->lwp_siginfo) && zp->zone_id != GLOBAL_ZONEID &&
995             sp->pr_info.si_zoneid != zp->zone_id) {
996                 sp->pr_info.si_pid = zp->zone_zsched->p_pid;
997                 sp->pr_info.si_uid = 0;
998                 sp->pr_info.si_ctid = -1;
999                 sp->pr_info.si_zoneid = zp->zone_id;
1000        }
1001        sp->pr_altstack.ss_sp =
1002            (caddr32_t)(uintptr_t)lwp->lwp_sigaltstack.ss_sp;
1003        sp->pr_altstack.ss_size = (size32_t)lwp->lwp_sigaltstack.ss_size;
1004        sp->pr_altstack.ss_flags = (int32_t)lwp->lwp_sigaltstack.ss_flags;
1005        prgetaction32(p, PTOU(p), lwp->lwp_cursig, &sp->pr_action);
1006        sp->pr_oldcontext = (caddr32_t)lwp->lwp_oldcontext;
1007        sp->pr_ustack = (caddr32_t)lwp->lwp_ustack;
1008        (void) strncpy(sp->pr_clname, sclass[t->t_cid].cl_name,
1009            sizeof (sp->pr_clname) - 1);
1010        if (flags & PR_STOPPED)
1011                hrt2ts32(t->t_stoptime, &sp->pr_tstamp);
1012        usr = ms->ms_acct[LMS_USER];
1013        sys = ms->ms_acct[LMS_SYSTEM] + ms->ms_acct[LMS_TRAP];
1014        scalehrtime(&usr);
1015        scalehrtime(&sys);
1016        hrt2ts32(usr, &sp->pr_utime);
1017        hrt2ts32(sys, &sp->pr_stime);

1019        /*
1020         * Fetch the current instruction, if not a system process.
1021         * We don't attempt this unless the lwp is stopped.
1022         */
1023        if ((p->p_flag & SSYS) || p->p_as == &kas)
1024                sp->pr_flags |= (PR_ISSYS|PR_PCINVAL);
1025        else if (!(flags & PR_STOPPED))
1026                sp->pr_flags |= PR_PCINVAL;
1027        else if (!prfetchinstr(lwp, &instr))
1028                sp->pr_flags |= PR_PCINVAL;
1029        else
1030                sp->pr_instr = (uint32_t)instr;

1032        /*
1033         * Drop p_lock while touching the lwp's stack.
1034         */
1035        mutex_exit(&p->p_lock);
1036        if (prisstep(lwp))
1037                sp->pr_flags |= PR_STEP;
1038        if ((flags & (PR_STOPPED|PR_ASLEEP)) && t->t_sysnum) {
1039                int i;

1041                sp->pr_syscall = get_syscall32_args(lwp,
1042                    (int *)sp->pr_sysarg, &i);
```



```
1043                sp->pr_nsysarg = (ushort_t)i;
1044        }
1045        if ((flags & PR_STOPPED) || t == curthread)
1046                prgetprregs32(lwp, sp->pr_reg);
1047        if ((t->t_state == TS_STOPPED && t->t_whystop == PR_SYSEXIT) ||
1048            (flags & PR_VFORKP)) {
1049                long r1, r2;
1050                user_t *up;
1051                auxv_t *auxp;
1052                int i;

1054                sp->pr_errno = prgetrvals(lwp, &r1, &r2);
1055                if (sp->pr_errno == 0) {
1056                        sp->pr_rval1 = (int32_t)r1;
1057                        sp->pr_rval2 = (int32_t)r2;
1058                        sp->pr_errpriv = PRIV_NONE;
1059                } else
1060                        sp->pr_errpriv = lwp->lwp_badpriv;

1062                if (t->t_sysnum == SYS_execve) {
1063                        up = PTOU(p);
1064                        sp->pr_sysarg[0] = 0;
1065                        sp->pr_sysarg[1] = (caddr32_t)up->u_argv;
1066                        sp->pr_sysarg[2] = (caddr32_t)up->u_envp;
1067                        for (i = 0, auxp = up->u_auxv;
1068                            i < sizeof (up->u_auxv) / sizeof (up->u_auxv[0]);
1069                            i++, auxp++) {
1070                                if (auxp->a_type == AT_SUN_EXECNAME) {
1071                                        sp->pr_sysarg[0] =
1072                                            (caddr32_t)
1073                                            (uintptr_t)auxp->a_un.a_ptr;
1074                                        break;
1075                                }
1076                        }
1077                }
1078        }
1079        if (prhasfp())
1080                prgetprfpregs32(lwp, &sp->pr_fpreg);
1081        mutex_enter(&p->p_lock);
1082 }

1084 void
1085 prgetstatus32(proc_t *p, pstatus32_t *sp, zone_t *zp)
1086 {
1087        kthread_t *t;

1089        ASSERT(MUTEX_HELD(&p->p_lock));

1091        t = prchoose(p);        /* returns locked thread */
1092        ASSERT(t != NULL);
1093        thread_unlock(t);

1095        /* just bzero the process part, prgetlwpstatus32() does the rest */
1096        bzero(sp, sizeof (pstatus32_t) - sizeof (lwpstatus32_t));
1097        sp->pr_nlwp = p->p_lwpcnt;
1098        sp->pr_nzomb = p->p_zombcnt;
1099        prassignset(&sp->pr_sigpend, &p->p_sig);
1100        sp->pr_brkbase = (uint32_t)(uintptr_t)p->p_brkbase;
1101        sp->pr_brksize = (uint32_t)p->p_brksize;
1102        sp->pr_stkbase = (uint32_t)(uintptr_t)prgetstackbase(p);
1103        sp->pr_stksize = (uint32_t)p->p_stksize;
1104        sp->pr_pid   = p->p_pid;
1105        if (curproc->p_zone->zone_id != GLOBAL_ZONEID &&
1106            (p->p_flag & SZONETOP)) {
1107                ASSERT(p->p_zone->zone_id != GLOBAL_ZONEID);
1108                /*
```

```
1109                 * Inside local zones, fake zsched's pid as parent pids for
1110                 * processes which reference processes outside of the zone.
1111                 */
1112                sp->pr_ppid = curproc->p_zone->zone_zsched->p_pid;
1113        } else {
1114                sp->pr_ppid = p->p_ppid;
1115        }
1116        sp->pr_pgid  = p->p_pgrp;
1117        sp->pr_sid   = p->p_sessp->s_sid;
1118        sp->pr_taskid = p->p_task->tk_tkid;
1119        sp->pr_projid = p->p_task->tk_proj->kpj_id;
1120        sp->pr_zoneid = p->p_zone->zone_id;
1121        bcopy(&p->p_secflags, &sp->pr_secflags, sizeof (psecflags_t));
1122 #endif /* ! codereview */
1123        hrt2ts32(mstate_aggr_state(p, LMS_USER), &sp->pr_utime);
1124        hrt2ts32(mstate_aggr_state(p, LMS_SYSTEM), &sp->pr_stime);
1125        TICK_TO_TIMESTRUC32(p->p_cutime, &sp->pr_cutime);
1126        TICK_TO_TIMESTRUC32(p->p_cstime, &sp->pr_cstime);
1127        prassignset(&sp->pr_sigtrace, &p->p_sigmask);
1128        prassignset(&sp->pr_flttrace, &p->p_fltmask);
1129        prassignset(&sp->pr_sysentry, &PTOU(p)->u_entrymask);
1130        prassignset(&sp->pr_sysexit, &PTOU(p)->u_exitmask);
1131        switch (p->p_model) {
1132        case DATAMODEL_ILP32:
1133                sp->pr_dmodel = PR_MODEL_ILP32;
1134                break;
1135        case DATAMODEL_LP64:
1136                sp->pr_dmodel = PR_MODEL_LP64;
1137                break;
1138        }
1139        if (p->p_agenttp)
1140                sp->pr_agentid = p->p_agenttp->t_tid;

1142        /* get the chosen lwp's status */
1143        prgetlwpstatus32(t, &sp->pr_lwp, zp);

1145        /* replicate the flags */
1146        sp->pr_flags = sp->pr_lwp.pr_flags;
1147 }
1148 #endif  /* _SYSCALL32_IMPL */

1150 /*
1151  * Return lwp status.
1152  */
1153 void
1154 prgetlwpstatus(kthread_t *t, lwpstatus_t *sp, zone_t *zp)
1155 {
1156        proc_t *p = ttoproc(t);
1157        klwp_t *lwp = ttolwp(t);
1158        struct mstate *ms = &lwp->lwp_mstate;
1159        hrtime_t usr, sys;
1160        int flags;
1161        ulong_t instr;

1163        ASSERT(MUTEX_HELD(&p->p_lock));

1165        bzero(sp, sizeof (*sp));
1166        flags = 0L;
1167        if (t->t_state == TS_STOPPED) {
1168                flags |= PR_STOPPED;
1169                if ((t->t_schedflag & TS_PSTART) == 0)
1170                        flags |= PR_ISTOP;
1171        } else if (VSTOPPED(t)) {
1172                flags |= PR_STOPPED|PR_ISTOP;
1173        }
1174        if (!(flags & PR_ISTOP) && (t->t_proc_flag & TP_PRSTOP))
```



```
1175                flags |= PR_DSTOP;
1176        if (lwp->lwp_asleep)
1177                flags |= PR_ASLEEP;
1178        if (t == p->p_agenttp)
1179                flags |= PR_AGENT;
1180        if (!(t->t_proc_flag & TP_TWAIT))
1181                flags |= PR_DETACH;
1182        if (t->t_proc_flag & TP_DAEMON)
1183                flags |= PR_DAEMON;
1184        if (p->p_proc_flag & P_PR_FORK)
1185                flags |= PR_FORK;
1186        if (p->p_proc_flag & P_PR_RUNLCL)
1187                flags |= PR_RLC;
1188        if (p->p_proc_flag & P_PR_KILLCL)
1189                flags |= PR_KLC;
1190        if (p->p_proc_flag & P_PR_ASYNC)
1191                flags |= PR_ASYNC;
1192        if (p->p_proc_flag & P_PR_BPTADJ)
1193                flags |= PR_BPTADJ;
1194        if (p->p_proc_flag & P_PR_PTRACE)
1195                flags |= PR_PTRACE;
1196        if (p->p_flag & SMSACCT)
1197                flags |= PR_MSACCT;
1198        if (p->p_flag & SMSFORK)
1199                flags |= PR_MSFORK;
1200        if (p->p_flag & SVFWAIT)
1201                flags |= PR_VFORKP;
1202        if (p->p_pgidp->pid_pgorphaned)
1203                flags |= PR_ORPHAN;
1204        if (p->p_pidflag & CLDNOSIGCHLD)
1205                flags |= PR_NOSIGCHLD;
1206        if (p->p_pidflag & CLDWAITPID)
1207                flags |= PR_WAITPID;
1208        sp->pr_flags = flags;
1209        if (VSTOPPED(t)) {
1210                sp->pr_why   = PR_REQUESTED;
1211                sp->pr_what  = 0;
1212        } else {
1213                sp->pr_why   = t->t_whystop;
1214                sp->pr_what  = t->t_whatstop;
1215        }
1216        sp->pr_lwpid = t->t_tid;
1217        sp->pr_cursig  = lwp->lwp_cursig;
1218        prassignset(&sp->pr_lwppend, &t->t_sig);
1219        schedctl_finish_sigblock(t);
1220        prassignset(&sp->pr_lwphold, &t->t_hold);
1221        if (t->t_whystop == PR_FAULTED)
1222                bcopy(&lwp->lwp_siginfo,
1223                    &sp->pr_info, sizeof (k_siginfo_t));
1224        else if (lwp->lwp_curinfo)
1225                bcopy(&lwp->lwp_curinfo->sq_info,
1226                    &sp->pr_info, sizeof (k_siginfo_t));
1227        if (SI_FROMUSER(&lwp->lwp_siginfo) && zp->zone_id != GLOBAL_ZONEID &&
1228            sp->pr_info.si_zoneid != zp->zone_id) {
1229                sp->pr_info.si_pid = zp->zone_zsched->p_pid;
1230                sp->pr_info.si_uid = 0;
1231                sp->pr_info.si_ctid = -1;
1232                sp->pr_info.si_zoneid = zp->zone_id;
1233        }
1234        sp->pr_altstack = lwp->lwp_sigaltstack;
1235        prgetaction(p, PTOU(p), lwp->lwp_cursig, &sp->pr_action);
1236        sp->pr_oldcontext = (uintptr_t)lwp->lwp_oldcontext;
1237        sp->pr_ustack = lwp->lwp_ustack;
1238        (void) strncpy(sp->pr_clname, sclass[t->t_cid].cl_name,
1239            sizeof (sp->pr_clname) - 1);
1240        if (flags & PR_STOPPED)
```

```
1241                hrt2ts(t->t_stoptime, &sp->pr_tstamp);
1242        usr = ms->ms_acct[LMS_USER];
1243        sys = ms->ms_acct[LMS_SYSTEM] + ms->ms_acct[LMS_TRAP];
1244        scalehrtime(&usr);
1245        scalehrtime(&sys);
1246        hrt2ts(usr, &sp->pr_utime);
1247        hrt2ts(sys, &sp->pr_stime);

1249        /*
1250         * Fetch the current instruction, if not a system process.
1251         * We don't attempt this unless the lwp is stopped.
1252         */
1253        if ((p->p_flag & SSYS) || p->p_as == &kas)
1254                sp->pr_flags |= (PR_ISSYS|PR_PCINVAL);
1255        else if (!(flags & PR_STOPPED))
1256                sp->pr_flags |= PR_PCINVAL;
1257        else if (!prfetchinstr(lwp, &instr))
1258                sp->pr_flags |= PR_PCINVAL;
1259        else
1260                sp->pr_instr = instr;

1262        /*
1263         * Drop p_lock while touching the lwp's stack.
1264         */
1265        mutex_exit(&p->p_lock);
1266        if (prisstep(lwp))
1267                sp->pr_flags |= PR_STEP;
1268        if ((flags & (PR_STOPPED|PR_ASLEEP)) && t->t_sysnum) {
1269                int i;

1271                sp->pr_syscall = get_syscall_args(lwp,
1272                    (long *)sp->pr_sysarg, &i);
1273                sp->pr_nsysarg = (ushort_t)i;
1274        }
1275        if ((flags & PR_STOPPED) || t == curthread)
1276                prgetprregs(lwp, sp->pr_reg);
1277        if ((t->t_state == TS_STOPPED && t->t_whystop == PR_SYSEXIT) ||
1278            (flags & PR_VFORKP)) {
1279                user_t *up;
1280                auxv_t *auxp;
1281                int i;

1283                sp->pr_errno = prgetrvals(lwp, &sp->pr_rval1, &sp->pr_rval2);
1284                if (sp->pr_errno == 0)
1285                        sp->pr_errpriv = PRIV_NONE;
1286                else
1287                        sp->pr_errpriv = lwp->lwp_badpriv;

1289                if (t->t_sysnum == SYS_execve) {
1290                        up = PTOU(p);
1291                        sp->pr_sysarg[0] = 0;
1292                        sp->pr_sysarg[1] = (uintptr_t)up->u_argv;
1293                        sp->pr_sysarg[2] = (uintptr_t)up->u_envp;
1294                        for (i = 0, auxp = up->u_auxv;
1295                            i < sizeof (up->u_auxv) / sizeof (up->u_auxv[0]);
1296                            i++, auxp++) {
1297                                if (auxp->a_type == AT_SUN_EXECNAME) {
1298                                        sp->pr_sysarg[0] =
1299                                            (uintptr_t)auxp->a_un.a_ptr;
1300                                        break;
1301                                }
1302                        }
1303                }
1304        }
1305        if (prhasfp())
1306                prgetprfpregs(lwp, &sp->pr_fpreg);
```



```
1307        mutex_enter(&p->p_lock);
1308 }

1310 /*
1311  * Get the sigaction structure for the specified signal.  The u-block
1312  * must already have been mapped in by the caller.
1313  */
1314 void
1315 prgetaction(proc_t *p, user_t *up, uint_t sig, struct sigaction *sp)
1316 {
1317        int nsig = PROC_IS_BRANDED(curproc)? BROP(curproc)->b_nsig : NSIG;

1319        bzero(sp, sizeof (*sp));

1321        if (sig != 0 && (unsigned)sig < nsig) {
1322                sp->sa_handler = up->u_signal[sig-1];
1323                prassignset(&sp->sa_mask, &up->u_sigmask[sig-1]);
1324                if (sigismember(&up->u_sigonstack, sig))
1325                        sp->sa_flags |= SA_ONSTACK;
1326                if (sigismember(&up->u_sigresethand, sig))
1327                        sp->sa_flags |= SA_RESETHAND;
1328                if (sigismember(&up->u_sigrestart, sig))
1329                        sp->sa_flags |= SA_RESTART;
1330                if (sigismember(&p->p_siginfo, sig))
1331                        sp->sa_flags |= SA_SIGINFO;
1332                if (sigismember(&up->u_signodefer, sig))
1333                        sp->sa_flags |= SA_NODEFER;
1334                if (sig == SIGCLD) {
1335                        if (p->p_flag & SNOWAIT)
1336                                sp->sa_flags |= SA_NOCLDWAIT;
1337                        if ((p->p_flag & SJCTL) == 0)
1338                                sp->sa_flags |= SA_NOCLDSTOP;
1339                }
1340        }
1341 }

1343 #ifdef _SYSCALL32_IMPL
1344 void
1345 prgetaction32(proc_t *p, user_t *up, uint_t sig, struct sigaction32 *sp)
1346 {
1347        int nsig = PROC_IS_BRANDED(curproc)? BROP(curproc)->b_nsig : NSIG;

1349        bzero(sp, sizeof (*sp));

1351        if (sig != 0 && (unsigned)sig < nsig) {
1352                sp->sa_handler = (caddr32_t)(uintptr_t)up->u_signal[sig-1];
1353                prassignset(&sp->sa_mask, &up->u_sigmask[sig-1]);
1354                if (sigismember(&up->u_sigonstack, sig))
1355                        sp->sa_flags |= SA_ONSTACK;
1356                if (sigismember(&up->u_sigresethand, sig))
1357                        sp->sa_flags |= SA_RESETHAND;
1358                if (sigismember(&up->u_sigrestart, sig))
1359                        sp->sa_flags |= SA_RESTART;
1360                if (sigismember(&p->p_siginfo, sig))
1361                        sp->sa_flags |= SA_SIGINFO;
1362                if (sigismember(&up->u_signodefer, sig))
1363                        sp->sa_flags |= SA_NODEFER;
1364                if (sig == SIGCLD) {
1365                        if (p->p_flag & SNOWAIT)
1366                                sp->sa_flags |= SA_NOCLDWAIT;
1367                        if ((p->p_flag & SJCTL) == 0)
1368                                sp->sa_flags |= SA_NOCLDSTOP;
1369                }
1370        }
1371 }
1372 #endif  /* _SYSCALL32_IMPL */
```

```
1374 /*
1375  * Count the number of segments in this process's address space.
1376  */
1377 int
1378 prnsegs(struct as *as, int reserved)
1379 {
1380         int n = 0;
1381         struct seg *seg;

1383         ASSERT(as != &kas && AS_WRITE_HELD(as, &as->a_lock));

1385         for (seg = AS_SEGFIRST(as); seg != NULL; seg = AS_SEGNEXT(as, seg)) {
1386                 caddr_t eaddr = seg->s_base + pr_getsegsize(seg, reserved);
1387                 caddr_t saddr, naddr;
1388                 void *tmp = NULL;

1390                 for (saddr = seg->s_base; saddr < eaddr; saddr = naddr) {
1391                         (void) pr_getprot(seg, reserved, &tmp,
1392                             &saddr, &naddr, eaddr);
1393                         if (saddr != naddr)
1394                                 n++;
1395                 }

1397                 ASSERT(tmp == NULL);
1398         }

1400         return (n);
1401 }

1403 /*
1404  * Convert uint32_t to decimal string w/o leading zeros.
1405  * Add trailing null characters if 'len' is greater than string length.
1406  * Return the string length.
1407  */
1408 int
1409 pr_u32tos(uint32_t n, char *s, int len)
1410 {
1411         char cbuf[11];          /* 32-bit unsigned integer fits in 10 digits */
1412         char *cp = cbuf;
1413         char *end = s + len;

1415         do {
1416                 *cp++ = (char)(n % 10 + '0');
1417                 n /= 10;
1418         } while (n);

1420         len = (int)(cp - cbuf);

1422         do {
1423                 *s++ = *--cp;
1424         } while (cp > cbuf);

1426         while (s < end)         /* optional pad */
1427                 *s++ = '\0';

1429         return (len);
1430 }

1432 /*
1433  * Convert uint64_t to decimal string w/o leading zeros.
1434  * Return the string length.
1435  */
1436 static int
1437 pr_u64tos(uint64_t n, char *s)
1438 {
```



```
1439         char cbuf[21];          /* 64-bit unsigned integer fits in 20 digits */
1440         char *cp = cbuf;
1441         int len;

1443         do {
1444                 *cp++ = (char)(n % 10 + '0');
1445                 n /= 10;
1446         } while (n);

1448         len = (int)(cp - cbuf);

1450         do {
1451                 *s++ = *--cp;
1452         } while (cp > cbuf);

1454         return (len);
1455 }

1457 void
1458 pr_object_name(char *name, vnode_t *vp, struct vattr *vattr)
1459 {
1460         char *s = name;
1461         struct vfs *vfsp;
1462         struct vfssw *vfsswp;

1464         if ((vfsp = vp->v_vfsp) != NULL &&
1465             ((vfsswp = vfssw + vfsp->vfs_fstype), vfsswp->vsw_name) &&
1466             *vfsswp->vsw_name) {
1467                 (void) strcpy(s, vfsswp->vsw_name);
1468                 s += strlen(s);
1469                 *s++ = '.';
1470         }
1471         s += pr_u32tos(getmajor(vattr->va_fsid), s, 0);
1472         *s++ = '.';
1473         s += pr_u32tos(getminor(vattr->va_fsid), s, 0);
1474         *s++ = '.';
1475         s += pr_u64tos(vattr->va_nodeid, s);
1476         *s++ = '\0';
1477 }

1479 struct seg *
1480 break_seg(proc_t *p)
1481 {
1482         caddr_t addr = p->p_brkbase;
1483         struct seg *seg;
1484         struct vnode *vp;

1486         if (p->p_brksize != 0)
1487                 addr += p->p_brksize - 1;
1488         seg = as_segat(p->p_as, addr);
1489         if (seg != NULL && seg->s_ops == &segvn_ops &&
1490             (SEGOP_GETVP(seg, seg->s_base, &vp) != 0 || vp == NULL))
1491                 return (seg);
1492         return (NULL);
1493 }

1495 /*
1496  * Implementation of service functions to handle procfs generic chained
1497  * copyout buffers.
1498  */
1499 typedef struct pr_iobuf_list {
1500         list_node_t     piol_link;      /* buffer linkage */
1501         size_t          piol_size;      /* total size (header + data) */
1502         size_t          piol_usedsize;  /* amount to copy out from this buf */
1503 } piol_t;
```

```
#define MAPSIZE (64 * 1024)
#define PIOL_DATABUF(iol)       ((void *)(&(iol)[1]))

void
pr_iol_initlist(list_t *iolhead, size_t itemsize, int n)
{
        piol_t  *iol;
        size_t  initial_size = MIN(1, n) * itemsize;

        list_create(iolhead, sizeof (piol_t), offsetof(piol_t, piol_link));

        ASSERT(list_head(iolhead) == NULL);
        ASSERT(itemsize < MAPSIZE - sizeof (*iol));
        ASSERT(initial_size > 0);

        /*
         * Someone creating chained copyout buffers may ask for less than
         * MAPSIZE if the amount of data to be buffered is known to be
         * smaller than that.
         * But in order to prevent involuntary self-denial of service,
         * the requested input size is clamped at MAPSIZE.
         */
        initial_size = MIN(MAPSIZE, initial_size + sizeof (*iol));
        iol = kmem_alloc(initial_size, KM_SLEEP);
        list_insert_head(iolhead, iol);
        iol->piol_usedsize = 0;
        iol->piol_size = initial_size;
}

void *
pr_iol_newbuf(list_t *iolhead, size_t itemsize)
{
        piol_t  *iol;
        char    *new;

        ASSERT(itemsize < MAPSIZE - sizeof (*iol));
        ASSERT(list_head(iolhead) != NULL);

        iol = (piol_t *)list_tail(iolhead);

        if (iol->piol_size <
            iol->piol_usedsize + sizeof (*iol) + itemsize) {
                /*
                 * Out of space in the current buffer. Allocate more.
                 */
                piol_t *newiol;

                newiol = kmem_alloc(MAPSIZE, KM_SLEEP);
                newiol->piol_size = MAPSIZE;
                newiol->piol_usedsize = 0;

                list_insert_after(iolhead, iol, newiol);
                iol = list_next(iolhead, iol);
                ASSERT(iol == newiol);
        }
        new = (char *)PIOL_DATABUF(iol) + iol->piol_usedsize;
        iol->piol_usedsize += itemsize;
        bzero(new, itemsize);
        return (new);
}

int
pr_iol_copyout_and_free(list_t *iolhead, caddr_t *tgt, int errin)
{
        int error = errin;
        piol_t  *iol;
```



```
        while ((iol = list_head(iolhead)) != NULL) {
                list_remove(iolhead, iol);
                if (!error) {
                        if (copyout(PIOL_DATABUF(iol), *tgt,
                            iol->piol_usedsize))
                                error = EFAULT;
                        *tgt += iol->piol_usedsize;
                }
                kmem_free(iol, iol->piol_size);
        }
        list_destroy(iolhead);

        return (error);
}

int
pr_iol_uiomove_and_free(list_t *iolhead, uio_t *uiop, int errin)
{
        offset_t        off = uiop->uio_offset;
        char            *base;
        size_t          size;
        piol_t          *iol;
        int             error = errin;

        while ((iol = list_head(iolhead)) != NULL) {
                list_remove(iolhead, iol);
                base = PIOL_DATABUF(iol);
                size = iol->piol_usedsize;
                if (off <= size && error == 0 && uiop->uio_resid > 0)
                        error = uiomove(base + off, size - off,
                            UIO_READ, uiop);
                off = MAX(0, off - (offset_t)size);
                kmem_free(iol, iol->piol_size);
        }
        list_destroy(iolhead);

        return (error);
}

/*
 * Return an array of structures with memory map information.
 * We allocate here; the caller must deallocate.
 */
int
prgetmap(proc_t *p, int reserved, list_t *iolhead)
{
        struct as *as = p->p_as;
        prmap_t *mp;
        struct seg *seg;
        struct seg *brkseg, *stkseg;
        struct vnode *vp;
        struct vattr vattr;
        uint_t prot;

        ASSERT(as != &kas && AS_WRITE_HELD(as, &as->a_lock));

        /*
         * Request an initial buffer size that doesn't waste memory
         * if the address space has only a small number of segments.
         */
        pr_iol_initlist(iolhead, sizeof (*mp), avl_numnodes(&as->a_segtree));

        if ((seg = AS_SEGFIRST(as)) == NULL)
                return (0);
```

```
1637         brkseg = break_seg(p);
1638         stkseg = as_segat(as, prgetstackbase(p));
1639 
1640         do {
1641                 caddr_t eaddr = seg->s_base + pr_getsegsize(seg, reserved);
1642                 caddr_t saddr, naddr;
1643                 void *tmp = NULL;
1644 
1645                 for (saddr = seg->s_base; saddr < eaddr; saddr = naddr) {
1646                         prot = pr_getprot(seg, reserved, &tmp,
1647                             &saddr, &naddr, eaddr);
1648                         if (saddr == naddr)
1649                                 continue;
1650 
1651                         mp = pr_iol_newbuf(iolhead, sizeof (*mp));
1652 
1653                         mp->pr_vaddr = (uintptr_t)saddr;
1654                         mp->pr_size = naddr - saddr;
1655                         mp->pr_offset = SEGOP_GETOFFSET(seg, saddr);
1656                         mp->pr_mflags = 0;
1657                         if (prot & PROT_READ)
1658                                 mp->pr_mflags |= MA_READ;
1659                         if (prot & PROT_WRITE)
1660                                 mp->pr_mflags |= MA_WRITE;
1661                         if (prot & PROT_EXEC)
1662                                 mp->pr_mflags |= MA_EXEC;
1663                         if (SEGOP_GETTYPE(seg, saddr) & MAP_SHARED)
1664                                 mp->pr_mflags |= MA_SHARED;
1665                         if (SEGOP_GETTYPE(seg, saddr) & MAP_NORESERVE)
1666                                 mp->pr_mflags |= MA_NORESERVE;
1667                         if (seg->s_ops == &segspt_shmops ||
1668                             (seg->s_ops == &segvn_ops &&
1669                             (SEGOP_GETVP(seg, saddr, &vp) != 0 || vp == NULL)))
1670                                 mp->pr_mflags |= MA_ANON;
1671                         if (seg == brkseg)
1672                                 mp->pr_mflags |= MA_BREAK;
1673                         else if (seg == stkseg) {
1674                                 mp->pr_mflags |= MA_STACK;
1675                                 if (reserved) {
1676                                         size_t maxstack =
1677                                             ((size_t)p->p_stk_ctl +
1678                                             PAGEOFFSET) & PAGEMASK;
1679                                         mp->pr_vaddr =
1680                                             (uintptr_t)prgetstackbase(p) +
1681                                             p->p_stksize - maxstack;
1682                                         mp->pr_size = (uintptr_t)naddr -
1683                                             mp->pr_vaddr;
1684                                 }
1685                         }
1686                         if (seg->s_ops == &segspt_shmops)
1687                                 mp->pr_mflags |= MA_ISM | MA_SHM;
1688                         mp->pr_pagesize = PAGESIZE;
1689 
1690                         /*
1691                          * Manufacture a filename for the "object" directory.
1692                          */
1693                         vattr.va_mask = AT_FSID|AT_NODEID;
1694                         if (seg->s_ops == &segvn_ops &&
1695                             SEGOP_GETVP(seg, saddr, &vp) == 0 &&
1696                             vp != NULL && vp->v_type == VREG &&
1697                             VOP_GETATTR(vp, &vattr, 0, CRED(), NULL) == 0) {
1698                                 if (vp == p->p_exec)
1699                                         (void) strcpy(mp->pr_mapname, "a.out");
1700                                 else
1701                                         pr_object_name(mp->pr_mapname,
1702                                             vp, &vattr);
```



```
1703                         }
1704 
1705                         /*
1706                          * Get the SysV shared memory id, if any.
1707                          */
1708                         if ((mp->pr_mflags & MA_SHARED) && p->p_segacct &&
1709                             (mp->pr_shmid = shmgetid(p, seg->s_base)) !=
1710                             SHMID_NONE) {
1711                                 if (mp->pr_shmid == SHMID_FREE)
1712                                         mp->pr_shmid = -1;
1713 
1714                                 mp->pr_mflags |= MA_SHM;
1715                         } else {
1716                                 mp->pr_shmid = -1;
1717                         }
1718                 }
1719                 ASSERT(tmp == NULL);
1720         } while ((seg = AS_SEGNEXT(as, seg)) != NULL);
1721 
1722         return (0);
1723 }
1724 
1725 #ifdef _SYSCALL32_IMPL
1726 int
1727 prgetmap32(proc_t *p, int reserved, list_t *iolhead)
1728 {
1729         struct as *as = p->p_as;
1730         prmap32_t *mp;
1731         struct seg *seg;
1732         struct seg *brkseg, *stkseg;
1733         struct vnode *vp;
1734         struct vattr vattr;
1735         uint_t prot;
1736 
1737         ASSERT(as != &kas && AS_WRITE_HELD(as, &as->a_lock));
1738 
1739         /*
1740          * Request an initial buffer size that doesn't waste memory
1741          * if the address space has only a small number of segments.
1742          */
1743         pr_iol_initlist(iolhead, sizeof (*mp), avl_numnodes(&as->a_segtree));
1744 
1745         if ((seg = AS_SEGFIRST(as)) == NULL)
1746                 return (0);
1747 
1748         brkseg = break_seg(p);
1749         stkseg = as_segat(as, prgetstackbase(p));
1750 
1751         do {
1752                 caddr_t eaddr = seg->s_base + pr_getsegsize(seg, reserved);
1753                 caddr_t saddr, naddr;
1754                 void *tmp = NULL;
1755 
1756                 for (saddr = seg->s_base; saddr < eaddr; saddr = naddr) {
1757                         prot = pr_getprot(seg, reserved, &tmp,
1758                             &saddr, &naddr, eaddr);
1759                         if (saddr == naddr)
1760                                 continue;
1761 
1762                         mp = pr_iol_newbuf(iolhead, sizeof (*mp));
1763 
1764                         mp->pr_vaddr = (caddr32_t)(uintptr_t)saddr;
1765                         mp->pr_size = (size32_t)(naddr - saddr);
1766                         mp->pr_offset = SEGOP_GETOFFSET(seg, saddr);
1767                         mp->pr_mflags = 0;
1768                         if (prot & PROT_READ)
```

```
1769                            mp->pr_mflags |= MA_READ;
1770                    if (prot & PROT_WRITE)
1771                            mp->pr_mflags |= MA_WRITE;
1772                    if (prot & PROT_EXEC)
1773                            mp->pr_mflags |= MA_EXEC;
1774                    if (SEGOP_GETTYPE(seg, saddr) & MAP_SHARED)
1775                            mp->pr_mflags |= MA_SHARED;
1776                    if (SEGOP_GETTYPE(seg, saddr) & MAP_NORESERVE)
1777                            mp->pr_mflags |= MA_NORESERVE;
1778                    if (seg->s_ops == &segspt_shmops ||
1779                        (seg->s_ops == &segvn_ops &&
1780                        (SEGOP_GETVP(seg, saddr, &vp) != 0 || vp == NULL)))
1781                            mp->pr_mflags |= MA_ANON;
1782                    if (seg == brkseg)
1783                            mp->pr_mflags |= MA_BREAK;
1784                    else if (seg == stkseg) {
1785                            mp->pr_mflags |= MA_STACK;
1786                            if (reserved) {
1787                                    size_t maxstack =
1788                                        ((size_t)p->p_stk_ctl +
1789                                        PAGEOFFSET) & PAGEMASK;
1790                                    uintptr_t vaddr =
1791                                        (uintptr_t)prgetstackbase(p) +
1792                                        p->p_stksize - maxstack;
1793                                    mp->pr_vaddr = (caddr32_t)vaddr;
1794                                    mp->pr_size = (size32_t)
1795                                        ((uintptr_t)naddr - vaddr);
1796                            }
1797                    }
1798                    if (seg->s_ops == &segspt_shmops)
1799                            mp->pr_mflags |= MA_ISM | MA_SHM;
1800                    mp->pr_pagesize = PAGESIZE;

1802                    /*
1803                     * Manufacture a filename for the "object" directory.
1804                     */
1805                    vattr.va_mask = AT_FSID|AT_NODEID;
1806                    if (seg->s_ops == &segvn_ops &&
1807                        SEGOP_GETVP(seg, saddr, &vp) == 0 &&
1808                        vp != NULL && vp->v_type == VREG &&
1809                        VOP_GETATTR(vp, &vattr, 0, CRED(), NULL) == 0) {
1810                            if (vp == p->p_exec)
1811                                    (void) strcpy(mp->pr_mapname, "a.out");
1812                            else
1813                                    pr_object_name(mp->pr_mapname,
1814                                        vp, &vattr);
1815                    }

1817                    /*
1818                     * Get the SysV shared memory id, if any.
1819                     */
1820                    if ((mp->pr_mflags & MA_SHARED) && p->p_segacct &&
1821                        (mp->pr_shmid = shmgetid(p, seg->s_base)) !=
1822                        SHMID_NONE) {
1823                            if (mp->pr_shmid == SHMID_FREE)
1824                                    mp->pr_shmid = -1;

1826                            mp->pr_mflags |= MA_SHM;
1827                    } else {
1828                            mp->pr_shmid = -1;
1829                    }
1830            }
1831            ASSERT(tmp == NULL);
1832        } while ((seg = AS_SEGNEXT(as, seg)) != NULL);

1834        return (0);
```



```
1835 }
1836 #endif  /* _SYSCALL32_IMPL */

1838 /*
1839  * Return the size of the /proc page data file.
1840  */
1841 size_t
1842 prpdsize(struct as *as)
1843 {
1844         struct seg *seg;
1845         size_t size;

1847         ASSERT(as != &kas && AS_WRITE_HELD(as, &as->a_lock));

1849         if ((seg = AS_SEGFIRST(as)) == NULL)
1850                 return (0);

1852         size = sizeof (prpageheader_t);
1853         do {
1854                 caddr_t eaddr = seg->s_base + pr_getsegsize(seg, 0);
1855                 caddr_t saddr, naddr;
1856                 void *tmp = NULL;
1857                 size_t npage;

1859                 for (saddr = seg->s_base; saddr < eaddr; saddr = naddr) {
1860                         (void) pr_getprot(seg, 0, &tmp, &saddr, &naddr, eaddr);
1861                         if ((npage = (naddr - saddr) / PAGESIZE) != 0)
1862                                 size += sizeof (prasmap_t) + round8(npage);
1863                 }
1864                 ASSERT(tmp == NULL);
1865         } while ((seg = AS_SEGNEXT(as, seg)) != NULL);

1867         return (size);
1868 }

1870 #ifdef _SYSCALL32_IMPL
1871 size_t
1872 prpdsize32(struct as *as)
1873 {
1874         struct seg *seg;
1875         size_t size;

1877         ASSERT(as != &kas && AS_WRITE_HELD(as, &as->a_lock));

1879         if ((seg = AS_SEGFIRST(as)) == NULL)
1880                 return (0);

1882         size = sizeof (prpageheader32_t);
1883         do {
1884                 caddr_t eaddr = seg->s_base + pr_getsegsize(seg, 0);
1885                 caddr_t saddr, naddr;
1886                 void *tmp = NULL;
1887                 size_t npage;

1889                 for (saddr = seg->s_base; saddr < eaddr; saddr = naddr) {
1890                         (void) pr_getprot(seg, 0, &tmp, &saddr, &naddr, eaddr);
1891                         if ((npage = (naddr - saddr) / PAGESIZE) != 0)
1892                                 size += sizeof (prasmap32_t) + round8(npage);
1893                 }
1894                 ASSERT(tmp == NULL);
1895         } while ((seg = AS_SEGNEXT(as, seg)) != NULL);

1897         return (size);
1898 }
1899 #endif  /* _SYSCALL32_IMPL */
```

```
/*
 * Read page data information.
 */
int
prpdread(proc_t *p, uint_t hatid, struct uio *uiop)
{
        struct as *as = p->p_as;
        caddr_t buf;
        size_t size;
        prpageheader_t *php;
        prasmap_t *pmp;
        struct seg *seg;
        int error;

again:
        AS_LOCK_ENTER(as, &as->a_lock, RW_WRITER);

        if ((seg = AS_SEGFIRST(as)) == NULL) {
                AS_LOCK_EXIT(as, &as->a_lock);
                return (0);
        }
        size = prpdsize(as);
        if (uiop->uio_resid < size) {
                AS_LOCK_EXIT(as, &as->a_lock);
                return (E2BIG);
        }

        buf = kmem_zalloc(size, KM_SLEEP);
        php = (prpageheader_t *)buf;
        pmp = (prasmap_t *)(buf + sizeof (prpageheader_t));

        hrt2ts(gethrtime(), &php->pr_tstamp);
        php->pr_nmap = 0;
        php->pr_npage = 0;
        do {
                caddr_t eaddr = seg->s_base + pr_getsegsize(seg, 0);
                caddr_t saddr, naddr;
                void *tmp = NULL;

                for (saddr = seg->s_base; saddr < eaddr; saddr = naddr) {
                        struct vnode *vp;
                        struct vattr vattr;
                        size_t len;
                        size_t npage;
                        uint_t prot;
                        uintptr_t next;

                        prot = pr_getprot(seg, 0, &tmp, &saddr, &naddr, eaddr);
                        if ((len = (size_t)(naddr - saddr)) == 0)
                                continue;
                        npage = len / PAGESIZE;
                        next = (uintptr_t)(pmp + 1) + round8(npage);
                        /*
                         * It's possible that the address space can change
                         * subtlely even though we're holding as->a_lock
                         * due to the nondeterminism of page_exists() in
                         * the presence of asychronously flushed pages or
                         * mapped files whose sizes are changing.
                         * page_exists() may be called indirectly from
                         * pr_getprot() by a SEGOP_INCORE() routine.
                         * If this happens we need to make sure we don't
                         * overrun the buffer whose size we computed based
                         * on the initial iteration through the segments.
                         * Once we've detected an overflow, we need to clean
                         * up the temporary memory allocated in pr_getprot()
                         * and retry. If there's a pending signal, we return
```



```
                         * EINTR so that this thread can be dislodged if
                         * a latent bug causes us to spin indefinitely.
                         */
                        if (next > (uintptr_t)buf + size) {
                                pr_getprot_done(&tmp);
                                AS_LOCK_EXIT(as, &as->a_lock);

                                kmem_free(buf, size);

                                if (ISSIG(curthread, JUSTLOOKING))
                                        return (EINTR);

                                goto again;
                        }

                        php->pr_nmap++;
                        php->pr_npage += npage;
                        pmp->pr_vaddr = (uintptr_t)saddr;
                        pmp->pr_npage = npage;
                        pmp->pr_offset = SEGOP_GETOFFSET(seg, saddr);
                        pmp->pr_mflags = 0;
                        if (prot & PROT_READ)
                                pmp->pr_mflags |= MA_READ;
                        if (prot & PROT_WRITE)
                                pmp->pr_mflags |= MA_WRITE;
                        if (prot & PROT_EXEC)
                                pmp->pr_mflags |= MA_EXEC;
                        if (SEGOP_GETTYPE(seg, saddr) & MAP_SHARED)
                                pmp->pr_mflags |= MA_SHARED;
                        if (SEGOP_GETTYPE(seg, saddr) & MAP_NORESERVE)
                                pmp->pr_mflags |= MA_NORESERVE;
                        if (seg->s_ops == &segspt_shmops ||
                            (seg->s_ops == &segvn_ops &&
                            (SEGOP_GETVP(seg, saddr, &vp) != 0 || vp == NULL)))
                                pmp->pr_mflags |= MA_ANON;
                        if (seg->s_ops == &segspt_shmops)
                                pmp->pr_mflags |= MA_ISM | MA_SHM;
                        pmp->pr_pagesize = PAGESIZE;
                        /*
                         * Manufacture a filename for the "object" directory.
                         */
                        vattr.va_mask = AT_FSID|AT_NODEID;
                        if (seg->s_ops == &segvn_ops &&
                            SEGOP_GETVP(seg, saddr, &vp) == 0 &&
                            vp != NULL && vp->v_type == VREG &&
                            VOP_GETATTR(vp, &vattr, 0, CRED(), NULL) == 0) {
                                if (vp == p->p_exec)
                                        (void) strcpy(pmp->pr_mapname, "a.out");
                                else
                                        pr_object_name(pmp->pr_mapname,
                                            vp, &vattr);
                        }

                        /*
                         * Get the SysV shared memory id, if any.
                         */
                        if ((pmp->pr_mflags & MA_SHARED) && p->p_segacct &&
                            (pmp->pr_shmid = shmgetid(p, seg->s_base)) !=
                            SHMID_NONE) {
                                if (pmp->pr_shmid == SHMID_FREE)
                                        pmp->pr_shmid = -1;

                                pmp->pr_mflags |= MA_SHM;
                        } else {
                                pmp->pr_shmid = -1;
                        }
```

```
2034                        hat_getstat(as, saddr, len, hatid,
2035                            (char *)(pmp + 1), HAT_SYNC_ZERORM);
2036                        pmp = (prasmap_t *)next;
2037                }
2038                ASSERT(tmp == NULL);
2039        } while ((seg = AS_SEGNEXT(as, seg)) != NULL);

2041        AS_LOCK_EXIT(as, &as->a_lock);

2043        ASSERT((uintptr_t)pmp <= (uintptr_t)buf + size);
2044        error = uiomove(buf, (caddr_t)pmp - buf, UIO_READ, uiop);
2045        kmem_free(buf, size);

2047        return (error);
2048 }

2050 #ifdef _SYSCALL32_IMPL
2051 int
2052 prpdread32(proc_t *p, uint_t hatid, struct uio *uiop)
2053 {
2054        struct as *as = p->p_as;
2055        caddr_t buf;
2056        size_t size;
2057        prpageheader32_t *php;
2058        prasmap32_t *pmp;
2059        struct seg *seg;
2060        int error;

2062 again:
2063        AS_LOCK_ENTER(as, &as->a_lock, RW_WRITER);

2065        if ((seg = AS_SEGFIRST(as)) == NULL) {
2066                AS_LOCK_EXIT(as, &as->a_lock);
2067                return (0);
2068        }
2069        size = prpdsize32(as);
2070        if (uiop->uio_resid < size) {
2071                AS_LOCK_EXIT(as, &as->a_lock);
2072                return (E2BIG);
2073        }

2075        buf = kmem_zalloc(size, KM_SLEEP);
2076        php = (prpageheader32_t *)buf;
2077        pmp = (prasmap32_t *)(buf + sizeof (prpageheader32_t));

2079        hrt2ts32(gethrtime(), &php->pr_tstamp);
2080        php->pr_nmap = 0;
2081        php->pr_npage = 0;
2082        do {
2083                caddr_t eaddr = seg->s_base + pr_getsegsize(seg, 0);
2084                caddr_t saddr, naddr;
2085                void *tmp = NULL;

2087                for (saddr = seg->s_base; saddr < eaddr; saddr = naddr) {
2088                        struct vnode *vp;
2089                        struct vattr vattr;
2090                        size_t len;
2091                        size_t npage;
2092                        uint_t prot;
2093                        uintptr_t next;

2095                        prot = pr_getprot(seg, 0, &tmp, &saddr, &naddr, eaddr);
2096                        if ((len = (size_t)(naddr - saddr)) == 0)
2097                                continue;
2098                        npage = len / PAGESIZE;
```



```
2099                        next = (uintptr_t)(pmp + 1) + round8(npage);
2100                        /*
2101                         * It's possible that the address space can change
2102                         * subtlely even though we're holding as->a_lock
2103                         * due to the nondeterminism of page_exists() in
2104                         * the presence of asychronously flushed pages or
2105                         * mapped files whose sizes are changing.
2106                         * page_exists() may be called indirectly from
2107                         * pr_getprot() by a SEGOP_INCORE() routine.
2108                         * If this happens we need to make sure we don't
2109                         * overrun the buffer whose size we computed based
2110                         * on the initial iteration through the segments.
2111                         * Once we've detected an overflow, we need to clean
2112                         * up the temporary memory allocated in pr_getprot()
2113                         * and retry. If there's a pending signal, we return
2114                         * EINTR so that this thread can be dislodged if
2115                         * a latent bug causes us to spin indefinitely.
2116                         */
2117                        if (next > (uintptr_t)buf + size) {
2118                                pr_getprot_done(&tmp);
2119                                AS_LOCK_EXIT(as, &as->a_lock);

2121                                kmem_free(buf, size);

2123                                if (ISSIG(curthread, JUSTLOOKING))
2124                                        return (EINTR);

2126                                goto again;
2127                        }

2129                        php->pr_nmap++;
2130                        php->pr_npage += npage;
2131                        pmp->pr_vaddr = (caddr32_t)(uintptr_t)saddr;
2132                        pmp->pr_npage = (size32_t)npage;
2133                        pmp->pr_offset = SEGOP_GETOFFSET(seg, saddr);
2134                        pmp->pr_mflags = 0;
2135                        if (prot & PROT_READ)
2136                                pmp->pr_mflags |= MA_READ;
2137                        if (prot & PROT_WRITE)
2138                                pmp->pr_mflags |= MA_WRITE;
2139                        if (prot & PROT_EXEC)
2140                                pmp->pr_mflags |= MA_EXEC;
2141                        if (SEGOP_GETTYPE(seg, saddr) & MAP_SHARED)
2142                                pmp->pr_mflags |= MA_SHARED;
2143                        if (SEGOP_GETTYPE(seg, saddr) & MAP_NORESERVE)
2144                                pmp->pr_mflags |= MA_NORESERVE;
2145                        if (seg->s_ops == &segspt_shmops ||
2146                            (seg->s_ops == &segvn_ops &&
2147                            (SEGOP_GETVP(seg, saddr, &vp) != 0 || vp == NULL)))
2148                                pmp->pr_mflags |= MA_ANON;
2149                        if (seg->s_ops == &segspt_shmops)
2150                                pmp->pr_mflags |= MA_ISM | MA_SHM;
2151                        pmp->pr_pagesize = PAGESIZE;
2152                        /*
2153                         * Manufacture a filename for the "object" directory.
2154                         */
2155                        vattr.va_mask = AT_FSID|AT_NODEID;
2156                        if (seg->s_ops == &segvn_ops &&
2157                            SEGOP_GETVP(seg, saddr, &vp) == 0 &&
2158                            vp != NULL && vp->v_type == VREG &&
2159                            VOP_GETATTR(vp, &vattr, 0, CRED(), NULL) == 0) {
2160                                if (vp == p->p_exec)
2161                                        (void) strcpy(pmp->pr_mapname, "a.out");
2162                                else
2163                                        pr_object_name(pmp->pr_mapname,
2164                                            vp, &vattr);
```

```
2165                        }
2167                        /*
2168                         * Get the SysV shared memory id, if any.
2169                         */
2170                        if ((pmp->pr_mflags & MA_SHARED) && p->p_segacct &&
2171                            (pmp->pr_shmid = shmgetid(p, seg->s_base)) !=
2172                            SHMID_NONE) {
2173                                if (pmp->pr_shmid == SHMID_FREE)
2174                                        pmp->pr_shmid = -1;
2176                                pmp->pr_mflags |= MA_SHM;
2177                        } else {
2178                                pmp->pr_shmid = -1;
2179                        }
2181                        hat_getstat(as, saddr, len, hatid,
2182                            (char *)(pmp + 1), HAT_SYNC_ZERORM);
2183                        pmp = (prasmap32_t *)next;
2184                }
2185                ASSERT(tmp == NULL);
2186        } while ((seg = AS_SEGNEXT(as, seg)) != NULL);
2188        AS_LOCK_EXIT(as, &as->a_lock);
2190        ASSERT((uintptr_t)pmp <= (uintptr_t)buf + size);
2191        error = uiomove(buf, (caddr_t)pmp - buf, UIO_READ, uiop);
2192        kmem_free(buf, size);
2194        return (error);
2195 }
2196 #endif  /* _SYSCALL32_IMPL */
2198 ushort_t
2199 prgetpctcpu(uint64_t pct)
2200 {
2201        /*
2202         * The value returned will be relevant in the zone of the examiner,
2203         * which may not be the same as the zone which performed the procfs
2204         * mount.
2205         */
2206        int nonline = zone_ncpus_online_get(curproc->p_zone);
2208        /*
2209         * Prorate over online cpus so we don't exceed 100%
2210         */
2211        if (nonline > 1)
2212                pct /= nonline;
2213        pct >>= 16;             /* convert to 16-bit scaled integer */
2214        if (pct > 0x8000)       /* might happen, due to rounding */
2215                pct = 0x8000;
2216        return ((ushort_t)pct);
2217 }
2219 /*
2220  * Return information used by ps(1).
2221  */
2222 void
2223 prgetpsinfo(proc_t *p, psinfo_t *psp)
2224 {
2225        kthread_t *t;
2226        struct cred *cred;
2227        hrtime_t hrutime, hrstime;
2229        ASSERT(MUTEX_HELD(&p->p_lock));
```



```
2231        if ((t = prchoose(p)) == NULL)  /* returns locked thread */
2232                bzero(psp, sizeof (*psp));
2233        else {
2234                thread_unlock(t);
2235                bzero(psp, sizeof (*psp) - sizeof (psp->pr_lwp));
2236        }
2238        /*
2239         * only export SSYS and SMSACCT; everything else is off-limits to
2240         * userland apps.
2241         */
2242        psp->pr_flag = p->p_flag & (SSYS | SMSACCT);
2243        psp->pr_nlwp = p->p_lwpcnt;
2244        psp->pr_nzomb = p->p_zombcnt;
2245        mutex_enter(&p->p_crlock);
2246        cred = p->p_cred;
2247        psp->pr_uid = crgetruid(cred);
2248        psp->pr_euid = crgetuid(cred);
2249        psp->pr_gid = crgetrgid(cred);
2250        psp->pr_egid = crgetgid(cred);
2251        mutex_exit(&p->p_crlock);
2252        psp->pr_pid = p->p_pid;
2253        if (curproc->p_zone->zone_id != GLOBAL_ZONEID &&
2254            (p->p_flag & SZONETOP)) {
2255                ASSERT(p->p_zone->zone_id != GLOBAL_ZONEID);
2256                /*
2257                 * Inside local zones, fake zsched's pid as parent pids for
2258                 * processes which reference processes outside of the zone.
2259                 */
2260                psp->pr_ppid = curproc->p_zone->zone_zsched->p_pid;
2261        } else {
2262                psp->pr_ppid = p->p_ppid;
2263        }
2264        psp->pr_pgid = p->p_pgrp;
2265        psp->pr_sid = p->p_sessp->s_sid;
2266        psp->pr_taskid = p->p_task->tk_tkid;
2267        psp->pr_projid = p->p_task->tk_proj->kpj_id;
2268        psp->pr_poolid = p->p_pool->pool_id;
2269        psp->pr_zoneid = p->p_zone->zone_id;
2270        if ((psp->pr_contract = PRCTID(p)) == 0)
2271                psp->pr_contract = -1;
2272        psp->pr_addr = (uintptr_t)prgetpsaddr(p);
2273        switch (p->p_model) {
2274        case DATAMODEL_ILP32:
2275                psp->pr_dmodel = PR_MODEL_ILP32;
2276                break;
2277        case DATAMODEL_LP64:
2278                psp->pr_dmodel = PR_MODEL_LP64;
2279                break;
2280        }
2281        hrutime = mstate_aggr_state(p, LMS_USER);
2282        hrstime = mstate_aggr_state(p, LMS_SYSTEM);
2283        hrt2ts((hrutime + hrstime), &psp->pr_time);
2284        TICK_TO_TIMESTRUC(p->p_cutime + p->p_cstime, &psp->pr_ctime);
2286        if (t == NULL) {
2287                int wcode = p->p_wcode;         /* must be atomic read */
2289                if (wcode)
2290                        psp->pr_wstat = wstat(wcode, p->p_wdata);
2291                psp->pr_ttydev = PRNODEV;
2292                psp->pr_lwp.pr_state = SZOMB;
2293                psp->pr_lwp.pr_sname = 'Z';
2294                psp->pr_lwp.pr_bindpro = PBIND_NONE;
2295                psp->pr_lwp.pr_bindpset = PS_NONE;
2296        } else {
```

```
2297                user_t *up = PTOU(p);
2298                struct as *as;
2299                dev_t d;
2300                extern dev_t rwsconsdev, rconsdev, uconsdev;

2302                d = cttydev(p);
2303                /*
2304                 * If the controlling terminal is the real
2305                 * or workstation console device, map to what the
2306                 * user thinks is the console device. Handle case when
2307                 * rwsconsdev or rconsdev is set to NODEV for Starfire.
2308                 */
2309                if ((d == rwsconsdev || d == rconsdev) && d != NODEV)
2310                        d = uconsdev;
2311                psp->pr_ttydev = (d == NODEV) ? PRNODEV : d;
2312                psp->pr_start = up->u_start;
2313                bcopy(up->u_comm, psp->pr_fname,
2314                    MIN(sizeof (up->u_comm), sizeof (psp->pr_fname)-1));
2315                bcopy(up->u_psargs, psp->pr_psargs,
2316                    MIN(PRARGSZ-1, PSARGSZ));
2317                psp->pr_argc = up->u_argc;
2318                psp->pr_argv = up->u_argv;
2319                psp->pr_envp = up->u_envp;

2321                /* get the chosen lwp's lwpsinfo */
2322                prgetlwpsinfo(t, &psp->pr_lwp);

2324                /* compute %cpu for the process */
2325                if (p->p_lwpcnt == 1)
2326                        psp->pr_pctcpu = psp->pr_lwp.pr_pctcpu;
2327                else {
2328                        uint64_t pct = 0;
2329                        hrtime_t cur_time = gethrtime_unscaled();

2331                        t = p->p_tlist;
2332                        do {
2333                                pct += cpu_update_pct(t, cur_time);
2334                        } while ((t = t->t_forw) != p->p_tlist);

2336                        psp->pr_pctcpu = prgetpctcpu(pct);
2337                }
2338                if ((p->p_flag & SSYS) || (as = p->p_as) == &kas) {
2339                        psp->pr_size = 0;
2340                        psp->pr_rssize = 0;
2341                } else {
2342                        mutex_exit(&p->p_lock);
2343                        AS_LOCK_ENTER(as, &as->a_lock, RW_READER);
2344                        psp->pr_size = btopr(as->a_resvsize) *
2345                            (PAGESIZE / 1024);
2346                        psp->pr_rssize = rm_asrss(as) * (PAGESIZE / 1024);
2347                        psp->pr_pctmem = rm_pctmemory(as);
2348                        AS_LOCK_EXIT(as, &as->a_lock);
2349                        mutex_enter(&p->p_lock);
2350                }
2351        }
2352 }

2354 #ifdef _SYSCALL32_IMPL
2355 void
2356 prgetpsinfo32(proc_t *p, psinfo32_t *psp)
2357 {
2358        kthread_t *t;
2359        struct cred *cred;
2360        hrtime_t hrutime, hrstime;

2362        ASSERT(MUTEX_HELD(&p->p_lock));
```



```
2364        if ((t = prchoose(p)) == NULL)  /* returns locked thread */
2365                bzero(psp, sizeof (*psp));
2366        else {
2367                thread_unlock(t);
2368                bzero(psp, sizeof (*psp) - sizeof (psp->pr_lwp));
2369        }

2371        /*
2372         * only export SSYS and SMSACCT; everything else is off-limits to
2373         * userland apps.
2374         */
2375        psp->pr_flag = p->p_flag & (SSYS | SMSACCT);
2376        psp->pr_nlwp = p->p_lwpcnt;
2377        psp->pr_nzomb = p->p_zombcnt;
2378        mutex_enter(&p->p_crlock);
2379        cred = p->p_cred;
2380        psp->pr_uid = crgetruid(cred);
2381        psp->pr_euid = crgetuid(cred);
2382        psp->pr_gid = crgetrgid(cred);
2383        psp->pr_egid = crgetgid(cred);
2384        mutex_exit(&p->p_crlock);
2385        psp->pr_pid = p->p_pid;
2386        if (curproc->p_zone->zone_id != GLOBAL_ZONEID &&
2387            (p->p_flag & SZONETOP)) {
2388                ASSERT(p->p_zone->zone_id != GLOBAL_ZONEID);
2389                /*
2390                 * Inside local zones, fake zsched's pid as parent pids for
2391                 * processes which reference processes outside of the zone.
2392                 */
2393                psp->pr_ppid = curproc->p_zone->zone_zsched->p_pid;
2394        } else {
2395                psp->pr_ppid = p->p_ppid;
2396        }
2397        psp->pr_pgid = p->p_pgrp;
2398        psp->pr_sid = p->p_sessp->s_sid;
2399        psp->pr_taskid = p->p_task->tk_tkid;
2400        psp->pr_projid = p->p_task->tk_proj->kpj_id;
2401        psp->pr_poolid = p->p_pool->pool_id;
2402        psp->pr_zoneid = p->p_zone->zone_id;
2403        if ((psp->pr_contract = PRCTID(p)) == 0)
2404                psp->pr_contract = -1;
2405        psp->pr_addr = 0;       /* cannot represent 64-bit addr in 32 bits */
2406        switch (p->p_model) {
2407        case DATAMODEL_ILP32:
2408                psp->pr_dmodel = PR_MODEL_ILP32;
2409                break;
2410        case DATAMODEL_LP64:
2411                psp->pr_dmodel = PR_MODEL_LP64;
2412                break;
2413        }
2414        hrutime = mstate_aggr_state(p, LMS_USER);
2415        hrstime = mstate_aggr_state(p, LMS_SYSTEM);
2416        hrt2ts32(hrutime + hrstime, &psp->pr_time);
2417        TICK_TO_TIMESTRUC32(p->p_cutime + p->p_cstime, &psp->pr_ctime);

2419        if (t == NULL) {
2420                extern int wstat(int, int);     /* needs a header file */
2421                int wcode = p->p_wcode;         /* must be atomic read */

2423                if (wcode)
2424                        psp->pr_wstat = wstat(wcode, p->p_wdata);
2425                psp->pr_ttydev = PRNODEV32;
2426                psp->pr_lwp.pr_state = SZOMB;
2427                psp->pr_lwp.pr_sname = 'Z';
2428        } else {
```

```
2429                user_t *up = PTOU(p);
2430                struct as *as;
2431                dev_t d;
2432                extern dev_t rwsconsdev, rconsdev, uconsdev;

2434                d = cttydev(p);
2435                /*
2436                 * If the controlling terminal is the real
2437                 * or workstation console device, map to what the
2438                 * user thinks is the console device. Handle case when
2439                 * rwsconsdev or rconsdev is set to NODEV for Starfire.
2440                 */
2441                if ((d == rwsconsdev || d == rconsdev) && d != NODEV)
2442                        d = uconsdev;
2443                (void) cmpldev(&psp->pr_ttydev, d);
2444                TIMESPEC_TO_TIMESPEC32(&psp->pr_start, &up->u_start);
2445                bcopy(up->u_comm, psp->pr_fname,
2446                    MIN(sizeof (up->u_comm), sizeof (psp->pr_fname)-1));
2447                bcopy(up->u_psargs, psp->pr_psargs,
2448                    MIN(PRARGSZ-1, PSARGSZ));
2449                psp->pr_argc = up->u_argc;
2450                psp->pr_argv = (caddr32_t)up->u_argv;
2451                psp->pr_envp = (caddr32_t)up->u_envp;

2453                /* get the chosen lwp's lwpsinfo */
2454                prgetlwpsinfo32(t, &psp->pr_lwp);

2456                /* compute %cpu for the process */
2457                if (p->p_lwpcnt == 1)
2458                        psp->pr_pctcpu = psp->pr_lwp.pr_pctcpu;
2459                else {
2460                        uint64_t pct = 0;
2461                        hrtime_t cur_time;

2463                        t = p->p_tlist;
2464                        cur_time = gethrtime_unscaled();
2465                        do {
2466                                pct += cpu_update_pct(t, cur_time);
2467                        } while ((t = t->t_forw) != p->p_tlist);

2469                        psp->pr_pctcpu = prgetpctcpu(pct);
2470                }
2471                if ((p->p_flag & SSYS) || (as = p->p_as) == &kas) {
2472                        psp->pr_size = 0;
2473                        psp->pr_rssize = 0;
2474                } else {
2475                        mutex_exit(&p->p_lock);
2476                        AS_LOCK_ENTER(as, &as->a_lock, RW_READER);
2477                        psp->pr_size = (size32_t)
2478                            (btopr(as->a_resvsize) * (PAGESIZE / 1024));
2479                        psp->pr_rssize = (size32_t)
2480                            (rm_asrss(as) * (PAGESIZE / 1024));
2481                        psp->pr_pctmem = rm_pctmemory(as);
2482                        AS_LOCK_EXIT(as, &as->a_lock);
2483                        mutex_enter(&p->p_lock);
2484                }
2485        }

2487        /*
2488         * If we are looking at an LP64 process, zero out
2489         * the fields that cannot be represented in ILP32.
2490         */
2491        if (p->p_model != DATAMODEL_ILP32) {
2492                psp->pr_size = 0;
2493                psp->pr_rssize = 0;
2494                psp->pr_argv = 0;
```



```
2495                psp->pr_envp = 0;
2496        }
2497 }

2499 #endif  /* _SYSCALL32_IMPL */

2501 void
2502 prgetlwpsinfo(kthread_t *t, lwpsinfo_t *psp)
2503 {
2504        klwp_t *lwp = ttolwp(t);
2505        sobj_ops_t *sobj;
2506        char c, state;
2507        uint64_t pct;
2508        int retval, niceval;
2509        hrtime_t hrutime, hrstime;

2511        ASSERT(MUTEX_HELD(&ttoproc(t)->p_lock));

2513        bzero(psp, sizeof (*psp));

2515        psp->pr_flag = 0;       /* lwpsinfo_t.pr_flag is deprecated */
2516        psp->pr_lwpid = t->t_tid;
2517        psp->pr_addr = (uintptr_t)t;
2518        psp->pr_wchan = (uintptr_t)t->t_wchan;

2520        /* map the thread state enum into a process state enum */
2521        state = VSTOPPED(t) ? TS_STOPPED : t->t_state;
2522        switch (state) {
2523        case TS_SLEEP:          state = SSLEEP;         c = 'S';        break;
2524        case TS_RUN:            state = SRUN;           c = 'R';        break;
2525        case TS_ONPROC:         state = SONPROC;        c = 'O';        break;
2526        case TS_ZOMB:           state = SZOMB;          c = 'Z';        break;
2527        case TS_STOPPED:        state = SSTOP;          c = 'T';        break;
2528        case TS_WAIT:           state = SWAIT;          c = 'W';        break;
2529        default:                state = 0;              c = '?';        break;
2530        }
2531        psp->pr_state = state;
2532        psp->pr_sname = c;
2533        if ((sobj = t->t_sobj_ops) != NULL)
2534                psp->pr_stype = SOBJ_TYPE(sobj);
2535        retval = CL_DONICE(t, NULL, 0, &niceval);
2536        if (retval == 0) {
2537                psp->pr_oldpri = v.v_maxsyspri - t->t_pri;
2538                psp->pr_nice = niceval + NZERO;
2539        }
2540        psp->pr_syscall = t->t_sysnum;
2541        psp->pr_pri = t->t_pri;
2542        psp->pr_start.tv_sec = t->t_start;
2543        psp->pr_start.tv_nsec = 0L;
2544        hrutime = lwp->lwp_mstate.ms_acct[LMS_USER];
2545        scalehrtime(&hrutime);
2546        hrstime = lwp->lwp_mstate.ms_acct[LMS_SYSTEM] +
2547            lwp->lwp_mstate.ms_acct[LMS_TRAP];
2548        scalehrtime(&hrstime);
2549        hrt2ts(hrutime + hrstime, &psp->pr_time);
2550        /* compute %cpu for the lwp */
2551        pct = cpu_update_pct(t, gethrtime_unscaled());
2552        psp->pr_pctcpu = prgetpctcpu(pct);
2553        psp->pr_cpu = (psp->pr_pctcpu*100 + 0x6000) >> 15;      /* [0..99] */
2554        if (psp->pr_cpu > 99)
2555                psp->pr_cpu = 99;

2557        (void) strncpy(psp->pr_clname, sclass[t->t_cid].cl_name,
2558            sizeof (psp->pr_clname) - 1);
2559        bzero(psp->pr_name, sizeof (psp->pr_name));     /* XXX ??? */
2560        psp->pr_onpro = t->t_cpu->cpu_id;
```

```
2561         psp->pr_bindpro = t->t_bind_cpu;
2562         psp->pr_bindpset = t->t_bind_pset;
2563         psp->pr_lgrp = t->t_lpl->lpl_lgrpid;
2564 }

2566 #ifdef _SYSCALL32_IMPL
2567 void
2568 prgetlwpsinfo32(kthread_t *t, lwpsinfo32_t *psp)
2569 {
2570         proc_t *p = ttoproc(t);
2571         klwp_t *lwp = ttolwp(t);
2572         sobj_ops_t *sobj;
2573         char c, state;
2574         uint64_t pct;
2575         int retval, niceval;
2576         hrtime_t hrutime, hrstime;

2578         ASSERT(MUTEX_HELD(&p->p_lock));

2580         bzero(psp, sizeof (*psp));

2582         psp->pr_flag = 0;       /* lwpsinfo_t.pr_flag is deprecated */
2583         psp->pr_lwpid = t->t_tid;
2584         psp->pr_addr = 0;       /* cannot represent 64-bit addr in 32 bits */
2585         psp->pr_wchan = 0;      /* cannot represent 64-bit addr in 32 bits */

2587         /* map the thread state enum into a process state enum */
2588         state = VSTOPPED(t) ? TS_STOPPED : t->t_state;
2589         switch (state) {
2590         case TS_SLEEP:          state = SSLEEP;         c = 'S';        break;
2591         case TS_RUN:            state = SRUN;           c = 'R';        break;
2592         case TS_ONPROC:         state = SONPROC;        c = 'O';        break;
2593         case TS_ZOMB:           state = SZOMB;          c = 'Z';        break;
2594         case TS_STOPPED:        state = SSTOP;          c = 'T';        break;
2595         case TS_WAIT:           state = SWAIT;          c = 'W';        break;
2596         default:                state = 0;              c = '?';        break;
2597         }
2598         psp->pr_state = state;
2599         psp->pr_sname = c;
2600         if ((sobj = t->t_sobj_ops) != NULL)
2601                 psp->pr_stype = SOBJ_TYPE(sobj);
2602         retval = CL_DONICE(t, NULL, 0, &niceval);
2603         if (retval == 0) {
2604                 psp->pr_oldpri = v.v_maxsyspri - t->t_pri;
2605                 psp->pr_nice = niceval + NZERO;
2606         } else {
2607                 psp->pr_oldpri = 0;
2608                 psp->pr_nice = 0;
2609         }
2610         psp->pr_syscall = t->t_sysnum;
2611         psp->pr_pri = t->t_pri;
2612         psp->pr_start.tv_sec = (time32_t)t->t_start;
2613         psp->pr_start.tv_nsec = 0L;
2614         hrutime = lwp->lwp_mstate.ms_acct[LMS_USER];
2615         scalehrtime(&hrutime);
2616         hrstime = lwp->lwp_mstate.ms_acct[LMS_SYSTEM] +
2617             lwp->lwp_mstate.ms_acct[LMS_TRAP];
2618         scalehrtime(&hrstime);
2619         hrt2ts32(hrutime + hrstime, &psp->pr_time);
2620         /* compute %cpu for the lwp */
2621         pct = cpu_update_pct(t, gethrtime_unscaled());
2622         psp->pr_pctcpu = prgetpctcpu(pct);
2623         psp->pr_cpu = (psp->pr_pctcpu*100 + 0x6000) >> 15;      /* [0..99] */
2624         if (psp->pr_cpu > 99)
2625                 psp->pr_cpu = 99;
```



```
2627         (void) strncpy(psp->pr_clname, sclass[t->t_cid].cl_name,
2628             sizeof (psp->pr_clname) - 1);
2629         bzero(psp->pr_name, sizeof (psp->pr_name));     /* XXX ??? */
2630         psp->pr_onpro = t->t_cpu->cpu_id;
2631         psp->pr_bindpro = t->t_bind_cpu;
2632         psp->pr_bindpset = t->t_bind_pset;
2633         psp->pr_lgrp = t->t_lpl->lpl_lgrpid;
2634 }
2635 #endif  /* _SYSCALL32_IMPL */

2637 #ifdef _SYSCALL32_IMPL

2639 #define PR_COPY_FIELD(s, d, field)       d->field = s->field

2641 #define PR_COPY_FIELD_ILP32(s, d, field)                                \
2642         if (s->pr_dmodel == PR_MODEL_ILP32) {                   \
2643                 d->field = s->field;                            \
2644         }

2646 #define PR_COPY_TIMESPEC(s, d, field)                           \
2647         TIMESPEC_TO_TIMESPEC32(&d->field, &s->field);

2649 #define PR_COPY_BUF(s, d, field)                                \
2650         bcopy(s->field, d->field, sizeof (d->field));

2652 #define PR_IGNORE_FIELD(s, d, field)

2654 void
2655 lwpsinfo_kto32(const struct lwpsinfo *src, struct lwpsinfo32 *dest)
2656 {
2657         bzero(dest, sizeof (*dest));

2659         PR_COPY_FIELD(src, dest, pr_flag);
2660         PR_COPY_FIELD(src, dest, pr_lwpid);
2661         PR_IGNORE_FIELD(src, dest, pr_addr);
2662         PR_IGNORE_FIELD(src, dest, pr_wchan);
2663         PR_COPY_FIELD(src, dest, pr_stype);
2664         PR_COPY_FIELD(src, dest, pr_state);
2665         PR_COPY_FIELD(src, dest, pr_sname);
2666         PR_COPY_FIELD(src, dest, pr_nice);
2667         PR_COPY_FIELD(src, dest, pr_syscall);
2668         PR_COPY_FIELD(src, dest, pr_oldpri);
2669         PR_COPY_FIELD(src, dest, pr_cpu);
2670         PR_COPY_FIELD(src, dest, pr_pri);
2671         PR_COPY_FIELD(src, dest, pr_pctcpu);
2672         PR_COPY_TIMESPEC(src, dest, pr_start);
2673         PR_COPY_BUF(src, dest, pr_clname);
2674         PR_COPY_BUF(src, dest, pr_name);
2675         PR_COPY_FIELD(src, dest, pr_onpro);
2676         PR_COPY_FIELD(src, dest, pr_bindpro);
2677         PR_COPY_FIELD(src, dest, pr_bindpset);
2678         PR_COPY_FIELD(src, dest, pr_lgrp);
2679 }

2681 void
2682 psinfo_kto32(const struct psinfo *src, struct psinfo32 *dest)
2683 {
2684         bzero(dest, sizeof (*dest));

2686         PR_COPY_FIELD(src, dest, pr_flag);
2687         PR_COPY_FIELD(src, dest, pr_nlwp);
2688         PR_COPY_FIELD(src, dest, pr_pid);
2689         PR_COPY_FIELD(src, dest, pr_ppid);
2690         PR_COPY_FIELD(src, dest, pr_pgid);
2691         PR_COPY_FIELD(src, dest, pr_sid);
2692         PR_COPY_FIELD(src, dest, pr_uid);
```

```
2693         PR_COPY_FIELD(src, dest, pr_euid);
2694         PR_COPY_FIELD(src, dest, pr_gid);
2695         PR_COPY_FIELD(src, dest, pr_egid);
2696         PR_IGNORE_FIELD(src, dest, pr_addr);
2697         PR_COPY_FIELD_ILP32(src, dest, pr_size);
2698         PR_COPY_FIELD_ILP32(src, dest, pr_rssize);
2699         PR_COPY_FIELD(src, dest, pr_ttydev);
2700         PR_COPY_FIELD(src, dest, pr_pctcpu);
2701         PR_COPY_FIELD(src, dest, pr_pctmem);
2702         PR_COPY_TIMESPEC(src, dest, pr_start);
2703         PR_COPY_TIMESPEC(src, dest, pr_time);
2704         PR_COPY_TIMESPEC(src, dest, pr_ctime);
2705         PR_COPY_BUF(src, dest, pr_fname);
2706         PR_COPY_BUF(src, dest, pr_psargs);
2707         PR_COPY_FIELD(src, dest, pr_wstat);
2708         PR_COPY_FIELD(src, dest, pr_argc);
2709         PR_COPY_FIELD_ILP32(src, dest, pr_argv);
2710         PR_COPY_FIELD_ILP32(src, dest, pr_envp);
2711         PR_COPY_FIELD(src, dest, pr_dmodel);
2712         PR_COPY_FIELD(src, dest, pr_taskid);
2713         PR_COPY_FIELD(src, dest, pr_projid);
2714         PR_COPY_FIELD(src, dest, pr_nzomb);
2715         PR_COPY_FIELD(src, dest, pr_poolid);
2716         PR_COPY_FIELD(src, dest, pr_contract);
2717         PR_COPY_FIELD(src, dest, pr_poolid);
2718         PR_COPY_FIELD(src, dest, pr_poolid);

2720         lwpsinfo_kto32(&src->pr_lwp, &dest->pr_lwp);
2721 }

2723 #undef  PR_COPY_FIELD
2724 #undef  PR_COPY_FIELD_ILP32
2725 #undef  PR_COPY_TIMESPEC
2726 #undef  PR_COPY_BUF
2727 #undef  PR_IGNORE_FIELD

2729 #endif  /* _SYSCALL32_IMPL */

2731 /*
2732  * This used to get called when microstate accounting was disabled but
2733  * microstate information was requested.  Since Microstate accounting is on
2734  * regardless of the proc flags, this simply makes it appear to procfs that
2735  * microstate accounting is on.  This is relatively meaningless since you
2736  * can't turn it off, but this is here for the sake of appearances.
2737  */

2739 /*ARGSUSED*/
2740 void
2741 estimate_msacct(kthread_t *t, hrtime_t curtime)
2742 {
2743         proc_t *p;

2745         if (t == NULL)
2746                 return;

2748         p = ttoproc(t);
2749         ASSERT(MUTEX_HELD(&p->p_lock));

2751         /*
2752          * A system process (p0) could be referenced if the thread is
2753          * in the process of exiting.  Don't turn on microstate accounting
2754          * in that case.
2755          */
2756         if (p->p_flag & SSYS)
2757                 return;
```



```
2759         /*
2760          * Loop through all the LWPs (kernel threads) in the process.
2761          */
2762         t = p->p_tlist;
2763         do {
2764                 t->t_proc_flag |= TP_MSACCT;
2765         } while ((t = t->t_forw) != p->p_tlist);

2767         p->p_flag |= SMSACCT;                   /* set process-wide MSACCT */
2768 }

2770 /*
2771  * It's not really possible to disable microstate accounting anymore.
2772  * However, this routine simply turns off the ms accounting flags in a process
2773  * This way procfs can still pretend to turn microstate accounting on and
2774  * off for a process, but it actually doesn't do anything.  This is
2775  * a neutered form of preemptive idiot-proofing.
2776  */
2777 void
2778 disable_msacct(proc_t *p)
2779 {
2780         kthread_t *t;

2782         ASSERT(MUTEX_HELD(&p->p_lock));

2784         p->p_flag &= ~SMSACCT;          /* clear process-wide MSACCT */
2785         /*
2786          * Loop through all the LWPs (kernel threads) in the process.
2787          */
2788         if ((t = p->p_tlist) != NULL) {
2789                 do {
2790                         /* clear per-thread flag */
2791                         t->t_proc_flag &= ~TP_MSACCT;
2792                 } while ((t = t->t_forw) != p->p_tlist);
2793         }
2794 }

2796 /*
2797  * Return resource usage information.
2798  */
2799 void
2800 prgetusage(kthread_t *t, prhusage_t *pup)
2801 {
2802         klwp_t *lwp = ttolwp(t);
2803         hrtime_t *mstimep;
2804         struct mstate *ms = &lwp->lwp_mstate;
2805         int state;
2806         int i;
2807         hrtime_t curtime;
2808         hrtime_t waitrq;
2809         hrtime_t tmp1;

2811         curtime = gethrtime_unscaled();

2813         pup->pr_lwpid   = t->t_tid;
2814         pup->pr_count   = 1;
2815         pup->pr_create  = ms->ms_start;
2816         pup->pr_term    = ms->ms_term;
2817         scalehrtime(&pup->pr_create);
2818         scalehrtime(&pup->pr_term);
2819         if (ms->ms_term == 0) {
2820                 pup->pr_rtime = curtime - ms->ms_start;
2821                 scalehrtime(&pup->pr_rtime);
2822         } else {
2823                 pup->pr_rtime = ms->ms_term - ms->ms_start;
2824                 scalehrtime(&pup->pr_rtime);
```

```
2825        }


2828        pup->pr_utime    = ms->ms_acct[LMS_USER];
2829        pup->pr_stime    = ms->ms_acct[LMS_SYSTEM];
2830        pup->pr_ttime    = ms->ms_acct[LMS_TRAP];
2831        pup->pr_tftime   = ms->ms_acct[LMS_TFAULT];
2832        pup->pr_dftime   = ms->ms_acct[LMS_DFAULT];
2833        pup->pr_kftime   = ms->ms_acct[LMS_KFAULT];
2834        pup->pr_ltime    = ms->ms_acct[LMS_USER_LOCK];
2835        pup->pr_slptime  = ms->ms_acct[LMS_SLEEP];
2836        pup->pr_wtime    = ms->ms_acct[LMS_WAIT_CPU];
2837        pup->pr_stoptime = ms->ms_acct[LMS_STOPPED];

2839        prscaleusage(pup);

2841        /*
2842         * Adjust for time waiting in the dispatcher queue.
2843         */
2844        waitrq = t->t_waitrq;   /* hopefully atomic */
2845        if (waitrq != 0) {
2846                if (waitrq > curtime) {
2847                        curtime = gethrtime_unscaled();
2848                }
2849                tmp1 = curtime - waitrq;
2850                scalehrtime(&tmp1);
2851                pup->pr_wtime += tmp1;
2852                curtime = waitrq;
2853        }

2855        /*
2856         * Adjust for time spent in current microstate.
2857         */
2858        if (ms->ms_state_start > curtime) {
2859                curtime = gethrtime_unscaled();
2860        }

2862        i = 0;
2863        do {
2864                switch (state = t->t_mstate) {
2865                case LMS_SLEEP:
2866                        /*
2867                         * Update the timer for the current sleep state.
2868                         */
2869                        switch (state = ms->ms_prev) {
2870                        case LMS_TFAULT:
2871                        case LMS_DFAULT:
2872                        case LMS_KFAULT:
2873                        case LMS_USER_LOCK:
2874                                break;
2875                        default:
2876                                state = LMS_SLEEP;
2877                                break;
2878                        }
2879                        break;
2880                case LMS_TFAULT:
2881                case LMS_DFAULT:
2882                case LMS_KFAULT:
2883                case LMS_USER_LOCK:
2884                        state = LMS_SYSTEM;
2885                        break;
2886                }
2887                switch (state) {
2888                case LMS_USER:          mstimep = &pup->pr_utime;       break;
2889                case LMS_SYSTEM:        mstimep = &pup->pr_stime;       break;
2890                case LMS_TRAP:          mstimep = &pup->pr_ttime;       break;
```



```
2891                case LMS_TFAULT:        mstimep = &pup->pr_tftime;      break;
2892                case LMS_DFAULT:        mstimep = &pup->pr_dftime;      break;
2893                case LMS_KFAULT:        mstimep = &pup->pr_kftime;      break;
2894                case LMS_USER_LOCK:     mstimep = &pup->pr_ltime;       break;
2895                case LMS_SLEEP:         mstimep = &pup->pr_slptime;     break;
2896                case LMS_WAIT_CPU:      mstimep = &pup->pr_wtime;       break;
2897                case LMS_STOPPED:       mstimep = &pup->pr_stoptime;    break;
2898                default:                panic("prgetusage: unknown microstate");
2899                }
2900                tmp1 = curtime - ms->ms_state_start;
2901                if (tmp1 < 0) {
2902                        curtime = gethrtime_unscaled();
2903                        i++;
2904                        continue;
2905                }
2906                scalehrtime(&tmp1);
2907        } while (tmp1 < 0 && i < MAX_ITERS_SPIN);

2909        *mstimep += tmp1;

2911        /* update pup timestamp */
2912        pup->pr_tstamp = curtime;
2913        scalehrtime(&pup->pr_tstamp);

2915        /*
2916         * Resource usage counters.
2917         */
2918        pup->pr_minf  = lwp->lwp_ru.minflt;
2919        pup->pr_majf  = lwp->lwp_ru.majflt;
2920        pup->pr_nswap = lwp->lwp_ru.nswap;
2921        pup->pr_inblk = lwp->lwp_ru.inblock;
2922        pup->pr_oublk = lwp->lwp_ru.oublock;
2923        pup->pr_msnd  = lwp->lwp_ru.msgsnd;
2924        pup->pr_mrcv  = lwp->lwp_ru.msgrcv;
2925        pup->pr_sigs  = lwp->lwp_ru.nsignals;
2926        pup->pr_vctx  = lwp->lwp_ru.nvcsw;
2927        pup->pr_ictx  = lwp->lwp_ru.nivcsw;
2928        pup->pr_sysc  = lwp->lwp_ru.sysc;
2929        pup->pr_ioch  = lwp->lwp_ru.ioch;
2930 }

2932 /*
2933  * Convert ms_acct stats from unscaled high-res time to nanoseconds
2934  */
2935 void
2936 prscaleusage(prhusage_t *usg)
2937 {
2938        scalehrtime(&usg->pr_utime);
2939        scalehrtime(&usg->pr_stime);
2940        scalehrtime(&usg->pr_ttime);
2941        scalehrtime(&usg->pr_tftime);
2942        scalehrtime(&usg->pr_dftime);
2943        scalehrtime(&usg->pr_kftime);
2944        scalehrtime(&usg->pr_ltime);
2945        scalehrtime(&usg->pr_slptime);
2946        scalehrtime(&usg->pr_wtime);
2947        scalehrtime(&usg->pr_stoptime);
2948 }


2951 /*
2952  * Sum resource usage information.
2953  */
2954 void
2955 praddusage(kthread_t *t, prhusage_t *pup)
2956 {
```

```
2957         klwp_t *lwp = ttolwp(t);
2958         hrtime_t *mstimep;
2959         struct mstate *ms = &lwp->lwp_mstate;
2960         int state;
2961         int i;
2962         hrtime_t curtime;
2963         hrtime_t waitrq;
2964         hrtime_t tmp;
2965         prhusage_t conv;

2967         curtime = gethrtime_unscaled();

2969         if (ms->ms_term == 0) {
2970                 tmp = curtime - ms->ms_start;
2971                 scalehrtime(&tmp);
2972                 pup->pr_rtime += tmp;
2973         } else {
2974                 tmp = ms->ms_term - ms->ms_start;
2975                 scalehrtime(&tmp);
2976                 pup->pr_rtime += tmp;
2977         }

2979         conv.pr_utime = ms->ms_acct[LMS_USER];
2980         conv.pr_stime = ms->ms_acct[LMS_SYSTEM];
2981         conv.pr_ttime = ms->ms_acct[LMS_TRAP];
2982         conv.pr_tftime = ms->ms_acct[LMS_TFAULT];
2983         conv.pr_dftime = ms->ms_acct[LMS_DFAULT];
2984         conv.pr_kftime = ms->ms_acct[LMS_KFAULT];
2985         conv.pr_ltime = ms->ms_acct[LMS_USER_LOCK];
2986         conv.pr_slptime = ms->ms_acct[LMS_SLEEP];
2987         conv.pr_wtime = ms->ms_acct[LMS_WAIT_CPU];
2988         conv.pr_stoptime = ms->ms_acct[LMS_STOPPED];

2990         prscaleusage(&conv);

2992         pup->pr_utime   += conv.pr_utime;
2993         pup->pr_stime   += conv.pr_stime;
2994         pup->pr_ttime   += conv.pr_ttime;
2995         pup->pr_tftime  += conv.pr_tftime;
2996         pup->pr_dftime  += conv.pr_dftime;
2997         pup->pr_kftime  += conv.pr_kftime;
2998         pup->pr_ltime   += conv.pr_ltime;
2999         pup->pr_slptime += conv.pr_slptime;
3000         pup->pr_wtime   += conv.pr_wtime;
3001         pup->pr_stoptime += conv.pr_stoptime;

3003         /*
3004          * Adjust for time waiting in the dispatcher queue.
3005          */
3006         waitrq = t->t_waitrq;   /* hopefully atomic */
3007         if (waitrq != 0) {
3008                 if (waitrq > curtime) {
3009                         curtime = gethrtime_unscaled();
3010                 }
3011                 tmp = curtime - waitrq;
3012                 scalehrtime(&tmp);
3013                 pup->pr_wtime += tmp;
3014                 curtime = waitrq;
3015         }

3017         /*
3018          * Adjust for time spent in current microstate.
3019          */
3020         if (ms->ms_state_start > curtime) {
3021                 curtime = gethrtime_unscaled();
3022         }
```



```
3024         i = 0;
3025         do {
3026                 switch (state = t->t_mstate) {
3027                 case LMS_SLEEP:
3028                         /*
3029                          * Update the timer for the current sleep state.
3030                          */
3031                         switch (state = ms->ms_prev) {
3032                         case LMS_TFAULT:
3033                         case LMS_DFAULT:
3034                         case LMS_KFAULT:
3035                         case LMS_USER_LOCK:
3036                                 break;
3037                         default:
3038                                 state = LMS_SLEEP;
3039                                 break;
3040                         }
3041                         break;
3042                 case LMS_TFAULT:
3043                 case LMS_DFAULT:
3044                 case LMS_KFAULT:
3045                 case LMS_USER_LOCK:
3046                         state = LMS_SYSTEM;
3047                         break;
3048                 }
3049                 switch (state) {
3050                 case LMS_USER:          mstimep = &pup->pr_utime;       break;
3051                 case LMS_SYSTEM:        mstimep = &pup->pr_stime;       break;
3052                 case LMS_TRAP:          mstimep = &pup->pr_ttime;       break;
3053                 case LMS_TFAULT:        mstimep = &pup->pr_tftime;      break;
3054                 case LMS_DFAULT:        mstimep = &pup->pr_dftime;      break;
3055                 case LMS_KFAULT:        mstimep = &pup->pr_kftime;      break;
3056                 case LMS_USER_LOCK:     mstimep = &pup->pr_ltime;       break;
3057                 case LMS_SLEEP:         mstimep = &pup->pr_slptime;     break;
3058                 case LMS_WAIT_CPU:      mstimep = &pup->pr_wtime;       break;
3059                 case LMS_STOPPED:       mstimep = &pup->pr_stoptime;    break;
3060                 default:                panic("praddusage: unknown microstate");
3061                 }
3062                 tmp = curtime - ms->ms_state_start;
3063                 if (tmp < 0) {
3064                         curtime = gethrtime_unscaled();
3065                         i++;
3066                         continue;
3067                 }
3068                 scalehrtime(&tmp);
3069         } while (tmp < 0 && i < MAX_ITERS_SPIN);

3071         *mstimep += tmp;

3073         /* update pup timestamp */
3074         pup->pr_tstamp = curtime;
3075         scalehrtime(&pup->pr_tstamp);

3077         /*
3078          * Resource usage counters.
3079          */
3080         pup->pr_minf  += lwp->lwp_ru.minflt;
3081         pup->pr_majf  += lwp->lwp_ru.majflt;
3082         pup->pr_nswap += lwp->lwp_ru.nswap;
3083         pup->pr_inblk += lwp->lwp_ru.inblock;
3084         pup->pr_oublk += lwp->lwp_ru.oublock;
3085         pup->pr_msnd  += lwp->lwp_ru.msgsnd;
3086         pup->pr_mrcv  += lwp->lwp_ru.msgrcv;
3087         pup->pr_sigs  += lwp->lwp_ru.nsignals;
3088         pup->pr_vctx  += lwp->lwp_ru.nvcsw;
```

```
3089         pup->pr_ictx  += lwp->lwp_ru.nivcsw;
3090         pup->pr_sysc  += lwp->lwp_ru.sysc;
3091         pup->pr_ioch  += lwp->lwp_ru.ioch;
3092 }

3094 /*
3095  * Convert a prhusage_t to a prusage_t.
3096  * This means convert each hrtime_t to a timestruc_t
3097  * and copy the count fields uint64_t => ulong_t.
3098  */
3099 void
3100 prcvtusage(prhusage_t *pup, prusage_t *upup)
3101 {
3102         uint64_t *ullp;
3103         ulong_t *ulp;
3104         int i;

3106         upup->pr_lwpid = pup->pr_lwpid;
3107         upup->pr_count = pup->pr_count;

3109         hrt2ts(pup->pr_tstamp,  &upup->pr_tstamp);
3110         hrt2ts(pup->pr_create,  &upup->pr_create);
3111         hrt2ts(pup->pr_term,    &upup->pr_term);
3112         hrt2ts(pup->pr_rtime,   &upup->pr_rtime);
3113         hrt2ts(pup->pr_utime,   &upup->pr_utime);
3114         hrt2ts(pup->pr_stime,   &upup->pr_stime);
3115         hrt2ts(pup->pr_ttime,   &upup->pr_ttime);
3116         hrt2ts(pup->pr_tftime,  &upup->pr_tftime);
3117         hrt2ts(pup->pr_dftime,  &upup->pr_dftime);
3118         hrt2ts(pup->pr_kftime,  &upup->pr_kftime);
3119         hrt2ts(pup->pr_ltime,   &upup->pr_ltime);
3120         hrt2ts(pup->pr_slptime, &upup->pr_slptime);
3121         hrt2ts(pup->pr_wtime,   &upup->pr_wtime);
3122         hrt2ts(pup->pr_stoptime, &upup->pr_stoptime);
3123         bzero(upup->filltime, sizeof (upup->filltime));

3125         ullp = &pup->pr_minf;
3126         ulp = &upup->pr_minf;
3127         for (i = 0; i < 22; i++)
3128                 *ulp++ = (ulong_t)*ullp++;
3129 }

3131 #ifdef _SYSCALL32_IMPL
3132 void
3133 prcvtusage32(prhusage_t *pup, prusage32_t *upup)
3134 {
3135         uint64_t *ullp;
3136         uint32_t *ulp;
3137         int i;

3139         upup->pr_lwpid = pup->pr_lwpid;
3140         upup->pr_count = pup->pr_count;

3142         hrt2ts32(pup->pr_tstamp,        &upup->pr_tstamp);
3143         hrt2ts32(pup->pr_create,        &upup->pr_create);
3144         hrt2ts32(pup->pr_term,          &upup->pr_term);
3145         hrt2ts32(pup->pr_rtime,         &upup->pr_rtime);
3146         hrt2ts32(pup->pr_utime,         &upup->pr_utime);
3147         hrt2ts32(pup->pr_stime,         &upup->pr_stime);
3148         hrt2ts32(pup->pr_ttime,         &upup->pr_ttime);
3149         hrt2ts32(pup->pr_tftime,        &upup->pr_tftime);
3150         hrt2ts32(pup->pr_dftime,        &upup->pr_dftime);
3151         hrt2ts32(pup->pr_kftime,        &upup->pr_kftime);
3152         hrt2ts32(pup->pr_ltime,         &upup->pr_ltime);
3153         hrt2ts32(pup->pr_slptime,       &upup->pr_slptime);
3154         hrt2ts32(pup->pr_wtime,         &upup->pr_wtime);
```



```
3155         hrt2ts32(pup->pr_stoptime,      &upup->pr_stoptime);
3156         bzero(upup->filltime, sizeof (upup->filltime));

3158         ullp = &pup->pr_minf;
3159         ulp = &upup->pr_minf;
3160         for (i = 0; i < 22; i++)
3161                 *ulp++ = (uint32_t)*ullp++;
3162 }
3163 #endif  /* _SYSCALL32_IMPL */

3165 /*
3166  * Determine whether a set is empty.
3167  */
3168 int
3169 setisempty(uint32_t *sp, uint_t n)
3170 {
3171         while (n--)
3172                 if (*sp++)
3173                         return (0);
3174         return (1);
3175 }

3177 /*
3178  * Utility routine for establishing a watched area in the process.
3179  * Keep the list of watched areas sorted by virtual address.
3180  */
3181 int
3182 set_watched_area(proc_t *p, struct watched_area *pwa)
3183 {
3184         caddr_t vaddr = pwa->wa_vaddr;
3185         caddr_t eaddr = pwa->wa_eaddr;
3186         ulong_t flags = pwa->wa_flags;
3187         struct watched_area *target;
3188         avl_index_t where;
3189         int error = 0;

3191         /* we must not be holding p->p_lock, but the process must be locked */
3192         ASSERT(MUTEX_NOT_HELD(&p->p_lock));
3193         ASSERT(p->p_proc_flag & P_PR_LOCK);

3195         /*
3196          * If this is our first watchpoint, enable watchpoints for the process.
3197          */
3198         if (!pr_watch_active(p)) {
3199                 kthread_t *t;

3201                 mutex_enter(&p->p_lock);
3202                 if ((t = p->p_tlist) != NULL) {
3203                         do {
3204                                 watch_enable(t);
3205                         } while ((t = t->t_forw) != p->p_tlist);
3206                 }
3207                 mutex_exit(&p->p_lock);
3208         }

3210         target = pr_find_watched_area(p, pwa, &where);
3211         if (target != NULL) {
3212                 /*
3213                  * We discovered an existing, overlapping watched area.
3214                  * Allow it only if it is an exact match.
3215                  */
3216                 if (target->wa_vaddr != vaddr ||
3217                     target->wa_eaddr != eaddr)
3218                         error = EINVAL;
3219                 else if (target->wa_flags != flags) {
3220                         error = set_watched_page(p, vaddr, eaddr,
```

```
3221                            flags, target->wa_flags);
3222                        target->wa_flags = flags;
3223                }
3224                kmem_free(pwa, sizeof (struct watched_area));
3225        } else {
3226                avl_insert(&p->p_warea, pwa, where);
3227                error = set_watched_page(p, vaddr, eaddr, flags, 0);
3228        }

3230        return (error);
3231 }

3233 /*
3234  * Utility routine for clearing a watched area in the process.
3235  * Must be an exact match of the virtual address.
3236  * size and flags don't matter.
3237  */
3238 int
3239 clear_watched_area(proc_t *p, struct watched_area *pwa)
3240 {
3241        struct watched_area *found;

3243        /* we must not be holding p->p_lock, but the process must be locked */
3244        ASSERT(MUTEX_NOT_HELD(&p->p_lock));
3245        ASSERT(p->p_proc_flag & P_PR_LOCK);


3248        if (!pr_watch_active(p)) {
3249                kmem_free(pwa, sizeof (struct watched_area));
3250                return (0);
3251        }

3253        /*
3254         * Look for a matching address in the watched areas.  If a match is
3255         * found, clear the old watched area and adjust the watched page(s).  It
3256         * is not an error if there is no match.
3257         */
3258        if ((found = pr_find_watched_area(p, pwa, NULL)) != NULL &&
3259            found->wa_vaddr == pwa->wa_vaddr) {
3260                clear_watched_page(p, found->wa_vaddr, found->wa_eaddr,
3261                    found->wa_flags);
3262                avl_remove(&p->p_warea, found);
3263                kmem_free(found, sizeof (struct watched_area));
3264        }

3266        kmem_free(pwa, sizeof (struct watched_area));

3268        /*
3269         * If we removed the last watched area from the process, disable
3270         * watchpoints.
3271         */
3272        if (!pr_watch_active(p)) {
3273                kthread_t *t;

3275                mutex_enter(&p->p_lock);
3276                if ((t = p->p_tlist) != NULL) {
3277                        do {
3278                                watch_disable(t);
3279                        } while ((t = t->t_forw) != p->p_tlist);
3280                }
3281                mutex_exit(&p->p_lock);
3282        }

3284        return (0);
3285 }
```



```
3287 /*
3288  * Frees all the watched_area structures
3289  */
3290 void
3291 pr_free_watchpoints(proc_t *p)
3292 {
3293        struct watched_area *delp;
3294        void *cookie;

3296        cookie = NULL;
3297        while ((delp = avl_destroy_nodes(&p->p_warea, &cookie)) != NULL)
3298                kmem_free(delp, sizeof (struct watched_area));

3300        avl_destroy(&p->p_warea);
3301 }

3303 /*
3304  * This one is called by the traced process to unwatch all the
3305  * pages while deallocating the list of watched_page structs.
3306  */
3307 void
3308 pr_free_watched_pages(proc_t *p)
3309 {
3310        struct as *as = p->p_as;
3311        struct watched_page *pwp;
3312        uint_t prot;
3313        int    retrycnt, err;
3314        void *cookie;

3316        if (as == NULL || avl_numnodes(&as->a_wpage) == 0)
3317                return;

3319        ASSERT(MUTEX_NOT_HELD(&curproc->p_lock));
3320        AS_LOCK_ENTER(as, &as->a_lock, RW_WRITER);

3322        pwp = avl_first(&as->a_wpage);

3324        cookie = NULL;
3325        while ((pwp = avl_destroy_nodes(&as->a_wpage, &cookie)) != NULL) {
3326                retrycnt = 0;
3327                if ((prot = pwp->wp_oprot) != 0) {
3328                        caddr_t addr = pwp->wp_vaddr;
3329                        struct seg *seg;
3330                retry:

3332                        if ((pwp->wp_prot != prot ||
3333                            (pwp->wp_flags & WP_NOWATCH)) &&
3334                            (seg = as_segat(as, addr)) != NULL) {
3335                                err = SEGOP_SETPROT(seg, addr, PAGESIZE, prot);
3336                                if (err == IE_RETRY) {
3337                                        ASSERT(retrycnt == 0);
3338                                        retrycnt++;
3339                                        goto retry;
3340                                }
3341                        }
3342                }
3343                kmem_free(pwp, sizeof (struct watched_page));
3344        }

3346        avl_destroy(&as->a_wpage);
3347        p->p_wprot = NULL;

3349        AS_LOCK_EXIT(as, &as->a_lock);
3350 }

3352 /*
```

```
 * Insert a watched area into the list of watched pages.
 * If oflags is zero then we are adding a new watched area.
 * Otherwise we are changing the flags of an existing watched area.
 */
static int
set_watched_page(proc_t *p, caddr_t vaddr, caddr_t eaddr,
        ulong_t flags, ulong_t oflags)
{
        struct as *as = p->p_as;
        avl_tree_t *pwp_tree;
        struct watched_page *pwp, *newpwp;
        struct watched_page tpw;
        avl_index_t where;
        struct seg *seg;
        uint_t prot;
        caddr_t addr;

        /*
         * We need to pre-allocate a list of structures before we grab the
         * address space lock to avoid calling kmem_alloc(KM_SLEEP) with locks
         * held.
         */
        newpwp = NULL;
        for (addr = (caddr_t)((uintptr_t)vaddr & (uintptr_t)PAGEMASK);
            addr < eaddr; addr += PAGESIZE) {
                pwp = kmem_zalloc(sizeof (struct watched_page), KM_SLEEP);
                pwp->wp_list = newpwp;
                newpwp = pwp;
        }

        AS_LOCK_ENTER(as, &as->a_lock, RW_WRITER);

        /*
         * Search for an existing watched page to contain the watched area.
         * If none is found, grab a new one from the available list
         * and insert it in the active list, keeping the list sorted
         * by user-level virtual address.
         */
        if (p->p_flag & SVFWAIT)
                pwp_tree = &p->p_wpage;
        else
                pwp_tree = &as->a_wpage;

again:
        if (avl_numnodes(pwp_tree) > prnwatch) {
                AS_LOCK_EXIT(as, &as->a_lock);
                while (newpwp != NULL) {
                        pwp = newpwp->wp_list;
                        kmem_free(newpwp, sizeof (struct watched_page));
                        newpwp = pwp;
                }
                return (E2BIG);
        }

        tpw.wp_vaddr = (caddr_t)((uintptr_t)vaddr & (uintptr_t)PAGEMASK);
        if ((pwp = avl_find(pwp_tree, &tpw, &where)) == NULL) {
                pwp = newpwp;
                newpwp = newpwp->wp_list;
                pwp->wp_list = NULL;
                pwp->wp_vaddr = (caddr_t)((uintptr_t)vaddr &
                    (uintptr_t)PAGEMASK);
                avl_insert(pwp_tree, pwp, where);
        }

        ASSERT(vaddr >= pwp->wp_vaddr && vaddr < pwp->wp_vaddr + PAGESIZE);
```



```
        if (oflags & WA_READ)
                pwp->wp_read--;
        if (oflags & WA_WRITE)
                pwp->wp_write--;
        if (oflags & WA_EXEC)
                pwp->wp_exec--;

        ASSERT(pwp->wp_read >= 0);
        ASSERT(pwp->wp_write >= 0);
        ASSERT(pwp->wp_exec >= 0);

        if (flags & WA_READ)
                pwp->wp_read++;
        if (flags & WA_WRITE)
                pwp->wp_write++;
        if (flags & WA_EXEC)
                pwp->wp_exec++;

        if (!(p->p_flag & SVFWAIT)) {
                vaddr = pwp->wp_vaddr;
                if (pwp->wp_oprot == 0 &&
                    (seg = as_segat(as, vaddr)) != NULL) {
                        SEGOP_GETPROT(seg, vaddr, 0, &prot);
                        pwp->wp_oprot = (uchar_t)prot;
                        pwp->wp_prot = (uchar_t)prot;
                }
                if (pwp->wp_oprot != 0) {
                        prot = pwp->wp_oprot;
                        if (pwp->wp_read)
                                prot &= ~(PROT_READ|PROT_WRITE|PROT_EXEC);
                        if (pwp->wp_write)
                                prot &= ~PROT_WRITE;
                        if (pwp->wp_exec)
                                prot &= ~(PROT_READ|PROT_WRITE|PROT_EXEC);
                        if (!(pwp->wp_flags & WP_NOWATCH) &&
                            pwp->wp_prot != prot &&
                            (pwp->wp_flags & WP_SETPROT) == 0) {
                                pwp->wp_flags |= WP_SETPROT;
                                pwp->wp_list = p->p_wprot;
                                p->p_wprot = pwp;
                        }
                        pwp->wp_prot = (uchar_t)prot;
                }
        }

        /*
         * If the watched area extends into the next page then do
         * it over again with the virtual address of the next page.
         */
        if ((vaddr = pwp->wp_vaddr + PAGESIZE) < eaddr)
                goto again;

        AS_LOCK_EXIT(as, &as->a_lock);

        /*
         * Free any pages we may have over-allocated
         */
        while (newpwp != NULL) {
                pwp = newpwp->wp_list;
                kmem_free(newpwp, sizeof (struct watched_page));
                newpwp = pwp;
        }

        return (0);
}
```

```
3485 /*
3486  * Remove a watched area from the list of watched pages.
3487  * A watched area may extend over more than one page.
3488  */
3489 static void
3490 clear_watched_page(proc_t *p, caddr_t vaddr, caddr_t eaddr, ulong_t flags)
3491 {
3492         struct as *as = p->p_as;
3493         struct watched_page *pwp;
3494         struct watched_page tpw;
3495         avl_tree_t *tree;
3496         avl_index_t where;

3498         AS_LOCK_ENTER(as, &as->a_lock, RW_WRITER);

3500         if (p->p_flag & SVFWAIT)
3501                 tree = &p->p_wpage;
3502         else
3503                 tree = &as->a_wpage;

3505         tpw.wp_vaddr = vaddr =
3506             (caddr_t)((uintptr_t)vaddr & (uintptr_t)PAGEMASK);
3507         pwp = avl_find(tree, &tpw, &where);
3508         if (pwp == NULL)
3509                 pwp = avl_nearest(tree, where, AVL_AFTER);

3511         while (pwp != NULL && pwp->wp_vaddr < eaddr) {
3512                 ASSERT(vaddr <=  pwp->wp_vaddr);

3514                 if (flags & WA_READ)
3515                         pwp->wp_read--;
3516                 if (flags & WA_WRITE)
3517                         pwp->wp_write--;
3518                 if (flags & WA_EXEC)
3519                         pwp->wp_exec--;

3521                 if (pwp->wp_read + pwp->wp_write + pwp->wp_exec != 0) {
3522                         /*
3523                          * Reset the hat layer's protections on this page.
3524                          */
3525                         if (pwp->wp_oprot != 0) {
3526                                 uint_t prot = pwp->wp_oprot;

3528                                 if (pwp->wp_read)
3529                                         prot &=
3530                                             ~(PROT_READ|PROT_WRITE|PROT_EXEC);
3531                                 if (pwp->wp_write)
3532                                         prot &= ~PROT_WRITE;
3533                                 if (pwp->wp_exec)
3534                                         prot &=
3535                                             ~(PROT_READ|PROT_WRITE|PROT_EXEC);
3536                                 if (!(pwp->wp_flags & WP_NOWATCH) &&
3537                                     pwp->wp_prot != prot &&
3538                                     (pwp->wp_flags & WP_SETPROT) == 0) {
3539                                         pwp->wp_flags |= WP_SETPROT;
3540                                         pwp->wp_list = p->p_wprot;
3541                                         p->p_wprot = pwp;
3542                                 }
3543                                 pwp->wp_prot = (uchar_t)prot;
3544                         }
3545                 } else {
3546                         /*
3547                          * No watched areas remain in this page.
3548                          * Reset everything to normal.
3549                          */
3550                         if (pwp->wp_oprot != 0) {
```



```
3551                                 pwp->wp_prot = pwp->wp_oprot;
3552                                 if ((pwp->wp_flags & WP_SETPROT) == 0) {
3553                                         pwp->wp_flags |= WP_SETPROT;
3554                                         pwp->wp_list = p->p_wprot;
3555                                         p->p_wprot = pwp;
3556                                 }
3557                         }
3558                 }

3560                 pwp = AVL_NEXT(tree, pwp);
3561         }

3563         AS_LOCK_EXIT(as, &as->a_lock);
3564 }

3566 /*
3567  * Return the original protections for the specified page.
3568  */
3569 static void
3570 getwatchprot(struct as *as, caddr_t addr, uint_t *prot)
3571 {
3572         struct watched_page *pwp;
3573         struct watched_page tpw;

3575         ASSERT(AS_LOCK_HELD(as, &as->a_lock));

3577         tpw.wp_vaddr = (caddr_t)((uintptr_t)addr & (uintptr_t)PAGEMASK);
3578         if ((pwp = avl_find(&as->a_wpage, &tpw, NULL)) != NULL)
3579                 *prot = pwp->wp_oprot;
3580 }

3582 static prpagev_t *
3583 pr_pagev_create(struct seg *seg, int check_noreserve)
3584 {
3585         prpagev_t *pagev = kmem_alloc(sizeof (prpagev_t), KM_SLEEP);
3586         size_t total_pages = seg_pages(seg);

3588         /*
3589          * Limit the size of our vectors to pagev_lim pages at a time.  We need
3590          * 4 or 5 bytes of storage per page, so this means we limit ourself
3591          * to about a megabyte of kernel heap by default.
3592          */
3593         pagev->pg_npages = MIN(total_pages, pagev_lim);
3594         pagev->pg_pnbase = 0;

3596         pagev->pg_protv =
3597             kmem_alloc(pagev->pg_npages * sizeof (uint_t), KM_SLEEP);

3599         if (check_noreserve)
3600                 pagev->pg_incore =
3601                     kmem_alloc(pagev->pg_npages * sizeof (char), KM_SLEEP);
3602         else
3603                 pagev->pg_incore = NULL;

3605         return (pagev);
3606 }

3608 static void
3609 pr_pagev_destroy(prpagev_t *pagev)
3610 {
3611         if (pagev->pg_incore != NULL)
3612                 kmem_free(pagev->pg_incore, pagev->pg_npages * sizeof (char));

3614         kmem_free(pagev->pg_protv, pagev->pg_npages * sizeof (uint_t));
3615         kmem_free(pagev, sizeof (prpagev_t));
3616 }
```

```
3618 static caddr_t
3619 pr_pagev_fill(prpagev_t *pagev, struct seg *seg, caddr_t addr, caddr_t eaddr)
3620 {
3621         ulong_t lastpg = seg_page(seg, eaddr - 1);
3622         ulong_t pn, pnlim;
3623         caddr_t saddr;
3624         size_t len;

3626         ASSERT(addr >= seg->s_base && addr <= eaddr);

3628         if (addr == eaddr)
3629                 return (eaddr);

3631 refill:
3632         ASSERT(addr < eaddr);
3633         pagev->pg_pnbase = seg_page(seg, addr);
3634         pnlim = pagev->pg_pnbase + pagev->pg_npages;
3635         saddr = addr;

3637         if (lastpg < pnlim)
3638                 len = (size_t)(eaddr - addr);
3639         else
3640                 len = pagev->pg_npages * PAGESIZE;

3642         if (pagev->pg_incore != NULL) {
3643                 /*
3644                  * INCORE cleverly has different semantics than GETPROT:
3645                  * it returns info on pages up to but NOT including addr + len.
3646                  */
3647                 SEGOP_INCORE(seg, addr, len, pagev->pg_incore);
3648                 pn = pagev->pg_pnbase;

3650                 do {
3651                         /*
3652                          * Guilty knowledge here:  We know that segvn_incore
3653                          * returns more than just the low-order bit that
3654                          * indicates the page is actually in memory.  If any
3655                          * bits are set, then the page has backing store.
3656                          */
3657                         if (pagev->pg_incore[pn++ - pagev->pg_pnbase])
3658                                 goto out;

3660                 } while ((addr += PAGESIZE) < eaddr && pn < pnlim);

3662                 /*
3663                  * If we examined all the pages in the vector but we're not
3664                  * at the end of the segment, take another lap.
3665                  */
3666                 if (addr < eaddr)
3667                         goto refill;
3668         }

3670         /*
3671          * Need to take len - 1 because addr + len is the address of the
3672          * first byte of the page just past the end of what we want.
3673          */
3674 out:
3675         SEGOP_GETPROT(seg, saddr, len - 1, pagev->pg_protv);
3676         return (addr);
3677 }

3679 static caddr_t
3680 pr_pagev_nextprot(prpagev_t *pagev, struct seg *seg,
3681     caddr_t *saddrp, caddr_t eaddr, uint_t *protp)
3682 {
```



```
3683         /*
3684          * Our starting address is either the specified address, or the base
3685          * address from the start of the pagev.  If the latter is greater,
3686          * this means a previous call to pr_pagev_fill has already scanned
3687          * further than the end of the previous mapping.
3688          */
3689         caddr_t base = seg->s_base + pagev->pg_pnbase * PAGESIZE;
3690         caddr_t addr = MAX(*saddrp, base);
3691         ulong_t pn = seg_page(seg, addr);
3692         uint_t prot, nprot;

3694         /*
3695          * If we're dealing with noreserve pages, then advance addr to
3696          * the address of the next page which has backing store.
3697          */
3698         if (pagev->pg_incore != NULL) {
3699                 while (pagev->pg_incore[pn - pagev->pg_pnbase] == 0) {
3700                         if ((addr += PAGESIZE) == eaddr) {
3701                                 *saddrp = addr;
3702                                 prot = 0;
3703                                 goto out;
3704                         }
3705                         if (++pn == pagev->pg_pnbase + pagev->pg_npages) {
3706                                 addr = pr_pagev_fill(pagev, seg, addr, eaddr);
3707                                 if (addr == eaddr) {
3708                                         *saddrp = addr;
3709                                         prot = 0;
3710                                         goto out;
3711                                 }
3712                                 pn = seg_page(seg, addr);
3713                         }
3714                 }
3715         }

3717         /*
3718          * Get the protections on the page corresponding to addr.
3719          */
3720         pn = seg_page(seg, addr);
3721         ASSERT(pn >= pagev->pg_pnbase);
3722         ASSERT(pn < (pagev->pg_pnbase + pagev->pg_npages));

3724         prot = pagev->pg_protv[pn - pagev->pg_pnbase];
3725         getwatchprot(seg->s_as, addr, &prot);
3726         *saddrp = addr;

3728         /*
3729          * Now loop until we find a backed page with different protections
3730          * or we reach the end of this segment.
3731          */
3732         while ((addr += PAGESIZE) < eaddr) {
3733                 /*
3734                  * If pn has advanced to the page number following what we
3735                  * have information on, refill the page vector and reset
3736                  * addr and pn.  If pr_pagev_fill does not return the
3737                  * address of the next page, we have a discontiguity and
3738                  * thus have reached the end of the current mapping.
3739                  */
3740                 if (++pn == pagev->pg_pnbase + pagev->pg_npages) {
3741                         caddr_t naddr = pr_pagev_fill(pagev, seg, addr, eaddr);
3742                         if (naddr != addr)
3743                                 goto out;
3744                         pn = seg_page(seg, addr);
3745                 }

3747                 /*
3748                  * The previous page's protections are in prot, and it has
```

```
3749                 * backing.  If this page is MAP_NORESERVE and has no backing,
3750                 * then end this mapping and return the previous protections.
3751                 */
3752                if (pagev->pg_incore != NULL &&
3753                    pagev->pg_incore[pn - pagev->pg_pnbase] == 0)
3754                        break;

3756                /*
3757                 * Otherwise end the mapping if this page's protections (nprot)
3758                 * are different than those in the previous page (prot).
3759                 */
3760                nprot = pagev->pg_protv[pn - pagev->pg_pnbase];
3761                getwatchprot(seg->s_as, addr, &nprot);

3763                if (nprot != prot)
3764                        break;
3765        }

3767 out:
3768        *protp = prot;
3769        return (addr);
3770 }

3772 size_t
3773 pr_getsegsize(struct seg *seg, int reserved)
3774 {
3775        size_t size = seg->s_size;

3777        /*
3778         * If we're interested in the reserved space, return the size of the
3779         * segment itself.  Everything else in this function is a special case
3780         * to determine the actual underlying size of various segment types.
3781         */
3782        if (reserved)
3783                return (size);

3785        /*
3786         * If this is a segvn mapping of a regular file, return the smaller
3787         * of the segment size and the remaining size of the file beyond
3788         * the file offset corresponding to seg->s_base.
3789         */
3790        if (seg->s_ops == &segvn_ops) {
3791                vattr_t vattr;
3792                vnode_t *vp;

3794                vattr.va_mask = AT_SIZE;

3796                if (SEGOP_GETVP(seg, seg->s_base, &vp) == 0 &&
3797                    vp != NULL && vp->v_type == VREG &&
3798                    VOP_GETATTR(vp, &vattr, 0, CRED(), NULL) == 0) {

3800                        u_offset_t fsize = vattr.va_size;
3801                        u_offset_t offset = SEGOP_GETOFFSET(seg, seg->s_base);

3803                        if (fsize < offset)
3804                                fsize = 0;
3805                        else
3806                                fsize -= offset;

3808                        fsize = roundup(fsize, (u_offset_t)PAGESIZE);

3810                        if (fsize < (u_offset_t)size)
3811                                size = (size_t)fsize;
3812                }

3814                return (size);
```



```
3815        }

3817        /*
3818         * If this is an ISM shared segment, don't include pages that are
3819         * beyond the real size of the spt segment that backs it.
3820         */
3821        if (seg->s_ops == &segspt_shmops)
3822                return (MIN(spt_realsize(seg), size));

3824        /*
3825         * If this is segment is a mapping from /dev/null, then this is a
3826         * reservation of virtual address space and has no actual size.
3827         * Such segments are backed by segdev and have type set to neither
3828         * MAP_SHARED nor MAP_PRIVATE.
3829         */
3830        if (seg->s_ops == &segdev_ops &&
3831            ((SEGOP_GETTYPE(seg, seg->s_base) &
3832            (MAP_SHARED | MAP_PRIVATE)) == 0))
3833                return (0);

3835        /*
3836         * If this segment doesn't match one of the special types we handle,
3837         * just return the size of the segment itself.
3838         */
3839        return (size);
3840 }

3842 uint_t
3843 pr_getprot(struct seg *seg, int reserved, void **tmp,
3844        caddr_t *saddrp, caddr_t *naddrp, caddr_t eaddr)
3845 {
3846        struct as *as = seg->s_as;

3848        caddr_t saddr = *saddrp;
3849        caddr_t naddr;

3851        int check_noreserve;
3852        uint_t prot;

3854        union {
3855                struct segvn_data *svd;
3856                struct segdev_data *sdp;
3857                void *data;
3858        } s;

3860        s.data = seg->s_data;

3862        ASSERT(AS_WRITE_HELD(as, &as->a_lock));
3863        ASSERT(saddr >= seg->s_base && saddr < eaddr);
3864        ASSERT(eaddr <= seg->s_base + seg->s_size);

3866        /*
3867         * Don't include MAP_NORESERVE pages in the address range
3868         * unless their mappings have actually materialized.
3869         * We cheat by knowing that segvn is the only segment
3870         * driver that supports MAP_NORESERVE.
3871         */
3872        check_noreserve =
3873            (!reserved && seg->s_ops == &segvn_ops && s.svd != NULL &&
3874            (s.svd->vp == NULL || s.svd->vp->v_type != VREG) &&
3875            (s.svd->flags & MAP_NORESERVE));

3877        /*
3878         * Examine every page only as a last resort.  We use guilty knowledge
3879         * of segvn and segdev to avoid this: if there are no per-page
3880         * protections present in the segment and we don't care about
```

```
3881          * MAP_NORESERVE, then s_data->prot is the prot for the whole segment.
3882          */
3883         if (!check_noreserve && saddr == seg->s_base &&
3884             seg->s_ops == &segvn_ops && s.svd != NULL && s.svd->pageprot == 0) {
3885                 prot = s.svd->prot;
3886                 getwatchprot(as, saddr, &prot);
3887                 naddr = eaddr;
3888
3889         } else if (saddr == seg->s_base && seg->s_ops == &segdev_ops &&
3890             s.sdp != NULL && s.sdp->pageprot == 0) {
3891                 prot = s.sdp->prot;
3892                 getwatchprot(as, saddr, &prot);
3893                 naddr = eaddr;
3894
3895         } else {
3896                 prpagev_t *pagev;
3897
3898                 /*
3899                  * If addr is sitting at the start of the segment, then
3900                  * create a page vector to store protection and incore
3901                  * information for pages in the segment, and fill it.
3902                  * Otherwise, we expect *tmp to address the prpagev_t
3903                  * allocated by a previous call to this function.
3904                  */
3905                 if (saddr == seg->s_base) {
3906                         pagev = pr_pagev_create(seg, check_noreserve);
3907                         saddr = pr_pagev_fill(pagev, seg, saddr, eaddr);
3908
3909                         ASSERT(*tmp == NULL);
3910                         *tmp = pagev;
3911
3912                         ASSERT(saddr <= eaddr);
3913                         *saddrp = saddr;
3914
3915                         if (saddr == eaddr) {
3916                                 naddr = saddr;
3917                                 prot = 0;
3918                                 goto out;
3919                         }
3920
3921                 } else {
3922                         ASSERT(*tmp != NULL);
3923                         pagev = (prpagev_t *)*tmp;
3924                 }
3925
3926                 naddr = pr_pagev_nextprot(pagev, seg, saddrp, eaddr, &prot);
3927                 ASSERT(naddr <= eaddr);
3928         }
3929
3930 out:
3931         if (naddr == eaddr)
3932                 pr_getprot_done(tmp);
3933         *naddrp = naddr;
3934         return (prot);
3935 }
3936
3937 void
3938 pr_getprot_done(void **tmp)
3939 {
3940         if (*tmp != NULL) {
3941                 pr_pagev_destroy((prpagev_t *)*tmp);
3942                 *tmp = NULL;
3943         }
3944 }
3945
3946 /*
```



```
3947  * Return true iff the vnode is a /proc file from the object directory.
3948  */
3949 int
3950 pr_isobject(vnode_t *vp)
3951 {
3952         return (vn_matchops(vp, prvnodeops) && VTOP(vp)->pr_type == PR_OBJECT);
3953 }
3954
3955 /*
3956  * Return true iff the vnode is a /proc file opened by the process itself.
3957  */
3958 int
3959 pr_isself(vnode_t *vp)
3960 {
3961         /*
3962          * XXX: To retain binary compatibility with the old
3963          * ioctl()-based version of /proc, we exempt self-opens
3964          * of /proc/<pid> from being marked close-on-exec.
3965          */
3966         return (vn_matchops(vp, prvnodeops) &&
3967             (VTOP(vp)->pr_flags & PR_ISSELF) &&
3968             VTOP(vp)->pr_type != PR_PIDDIR);
3969 }
3970
3971 static ssize_t
3972 pr_getpagesize(struct seg *seg, caddr_t saddr, caddr_t *naddrp, caddr_t eaddr)
3973 {
3974         ssize_t pagesize, hatsize;
3975
3976         ASSERT(AS_WRITE_HELD(seg->s_as, &seg->s_as->a_lock));
3977         ASSERT(IS_P2ALIGNED(saddr, PAGESIZE));
3978         ASSERT(IS_P2ALIGNED(eaddr, PAGESIZE));
3979         ASSERT(saddr < eaddr);
3980
3981         pagesize = hatsize = hat_getpagesize(seg->s_as->a_hat, saddr);
3982         ASSERT(pagesize == -1 || IS_P2ALIGNED(pagesize, pagesize));
3983         ASSERT(pagesize != 0);
3984
3985         if (pagesize == -1)
3986                 pagesize = PAGESIZE;
3987
3988         saddr += P2NPHASE((uintptr_t)saddr, pagesize);
3989
3990         while (saddr < eaddr) {
3991                 if (hatsize != hat_getpagesize(seg->s_as->a_hat, saddr))
3992                         break;
3993                 ASSERT(IS_P2ALIGNED(saddr, pagesize));
3994                 saddr += pagesize;
3995         }
3996
3997         *naddrp = ((saddr < eaddr) ? saddr : eaddr);
3998         return (hatsize);
3999 }
4000
4001 /*
4002  * Return an array of structures with extended memory map information.
4003  * We allocate here; the caller must deallocate.
4004  */
4005 int
4006 prgetxmap(proc_t *p, list_t *iolhead)
4007 {
4008         struct as *as = p->p_as;
4009         prxmap_t *mp;
4010         struct seg *seg;
4011         struct seg *brkseg, *stkseg;
4012         struct vnode *vp;
```

```
4013        struct vattr vattr;
4014        uint_t prot;

4016        ASSERT(as != &kas && AS_WRITE_HELD(as, &as->a_lock));

4018        /*
4019         * Request an initial buffer size that doesn't waste memory
4020         * if the address space has only a small number of segments.
4021         */
4022        pr_iol_initlist(iolhead, sizeof (*mp), avl_numnodes(&as->a_segtree));

4024        if ((seg = AS_SEGFIRST(as)) == NULL)
4025                return (0);

4027        brkseg = break_seg(p);
4028        stkseg = as_segat(as, prgetstackbase(p));

4030        do {
4031                caddr_t eaddr = seg->s_base + pr_getsegsize(seg, 0);
4032                caddr_t saddr, naddr, baddr;
4033                void *tmp = NULL;
4034                ssize_t psz;
4035                char *parr;
4036                uint64_t npages;
4037                uint64_t pagenum;

4039                /*
4040                 * Segment loop part one: iterate from the base of the segment
4041                 * to its end, pausing at each address boundary (baddr) between
4042                 * ranges that have different virtual memory protections.
4043                 */
4044                for (saddr = seg->s_base; saddr < eaddr; saddr = baddr) {
4045                        prot = pr_getprot(seg, 0, &tmp, &saddr, &baddr, eaddr);
4046                        ASSERT(baddr >= saddr && baddr <= eaddr);

4048                        /*
4049                         * Segment loop part two: iterate from the current
4050                         * position to the end of the protection boundary,
4051                         * pausing at each address boundary (naddr) between
4052                         * ranges that have different underlying page sizes.
4053                         */
4054                        for (; saddr < baddr; saddr = naddr) {
4055                                psz = pr_getpagesize(seg, saddr, &naddr, baddr);
4056                                ASSERT(naddr >= saddr && naddr <= baddr);

4058                                mp = pr_iol_newbuf(iolhead, sizeof (*mp));

4060                                mp->pr_vaddr = (uintptr_t)saddr;
4061                                mp->pr_size = naddr - saddr;
4062                                mp->pr_offset = SEGOP_GETOFFSET(seg, saddr);
4063                                mp->pr_mflags = 0;
4064                                if (prot & PROT_READ)
4065                                        mp->pr_mflags |= MA_READ;
4066                                if (prot & PROT_WRITE)
4067                                        mp->pr_mflags |= MA_WRITE;
4068                                if (prot & PROT_EXEC)
4069                                        mp->pr_mflags |= MA_EXEC;
4070                                if (SEGOP_GETTYPE(seg, saddr) & MAP_SHARED)
4071                                        mp->pr_mflags |= MA_SHARED;
4072                                if (SEGOP_GETTYPE(seg, saddr) & MAP_NORESERVE)
4073                                        mp->pr_mflags |= MA_NORESERVE;
4074                                if (seg->s_ops == &segspt_shmops ||
4075                                    (seg->s_ops == &segvn_ops &&
4076                                    (SEGOP_GETVP(seg, saddr, &vp) != 0 ||
4077                                    vp == NULL)))
4078                                        mp->pr_mflags |= MA_ANON;
```



```
4079                                if (seg == brkseg)
4080                                        mp->pr_mflags |= MA_BREAK;
4081                                else if (seg == stkseg)
4082                                        mp->pr_mflags |= MA_STACK;
4083                                if (seg->s_ops == &segspt_shmops)
4084                                        mp->pr_mflags |= MA_ISM | MA_SHM;

4086                                mp->pr_pagesize = PAGESIZE;
4087                                if (psz == -1) {
4088                                        mp->pr_hatpagesize = 0;
4089                                } else {
4090                                        mp->pr_hatpagesize = psz;
4091                                }

4093                                /*
4094                                 * Manufacture a filename for the "object" dir.
4095                                 */
4096                                mp->pr_dev = PRNODEV;
4097                                vattr.va_mask = AT_FSID|AT_NODEID;
4098                                if (seg->s_ops == &segvn_ops &&
4099                                    SEGOP_GETVP(seg, saddr, &vp) == 0 &&
4100                                    vp != NULL && vp->v_type == VREG &&
4101                                    VOP_GETATTR(vp, &vattr, 0, CRED(),
4102                                    NULL) == 0) {
4103                                        mp->pr_dev = vattr.va_fsid;
4104                                        mp->pr_ino = vattr.va_nodeid;
4105                                        if (vp == p->p_exec)
4106                                                (void) strcpy(mp->pr_mapname,
4107                                                    "a.out");
4108                                        else
4109                                                pr_object_name(mp->pr_mapname,
4110                                                    vp, &vattr);
4111                                }

4113                                /*
4114                                 * Get the SysV shared memory id, if any.
4115                                 */
4116                                if ((mp->pr_mflags & MA_SHARED) &&
4117                                    p->p_segacct && (mp->pr_shmid = shmgetid(p,
4118                                    seg->s_base)) != SHMID_NONE) {
4119                                        if (mp->pr_shmid == SHMID_FREE)
4120                                                mp->pr_shmid = -1;

4122                                        mp->pr_mflags |= MA_SHM;
4123                                } else {
4124                                        mp->pr_shmid = -1;
4125                                }

4127                                npages = ((uintptr_t)(naddr - saddr)) >>
4128                                    PAGESHIFT;
4129                                parr = kmem_zalloc(npages, KM_SLEEP);

4131                                SEGOP_INCORE(seg, saddr, naddr - saddr, parr);

4133                                for (pagenum = 0; pagenum < npages; pagenum++) {
4134                                        if (parr[pagenum] & SEG_PAGE_INCORE)
4135                                                mp->pr_rss++;
4136                                        if (parr[pagenum] & SEG_PAGE_ANON)
4137                                                mp->pr_anon++;
4138                                        if (parr[pagenum] & SEG_PAGE_LOCKED)
4139                                                mp->pr_locked++;
4140                                }
4141                                kmem_free(parr, npages);
4142                        }
4143                }
4144                ASSERT(tmp == NULL);
```

```
4145         } while ((seg = AS_SEGNEXT(as, seg)) != NULL);

4147         return (0);
4148 }

4150 /*
4151  * Return the process's credentials.  We don't need a 32-bit equivalent of
4152  * this function because prcred_t and prcred32_t are actually the same.
4153  */
4154 void
4155 prgetcred(proc_t *p, prcred_t *pcrp)
4156 {
4157         mutex_enter(&p->p_crlock);
4158         cred2prcred(p->p_cred, pcrp);
4159         mutex_exit(&p->p_crlock);
4160 }

4162 /*
4163  * Compute actual size of the prpriv_t structure.
4164  */

4166 size_t
4167 prgetprivsize(void)
4168 {
4169         return (priv_prgetprivsize(NULL));
4170 }

4172 /*
4173  * Return the process's privileges.  We don't need a 32-bit equivalent of
4174  * this function because prpriv_t and prpriv32_t are actually the same.
4175  */
4176 void
4177 prgetpriv(proc_t *p, prpriv_t *pprp)
4178 {
4179         mutex_enter(&p->p_crlock);
4180         cred2prpriv(p->p_cred, pprp);
4181         mutex_exit(&p->p_crlock);
4182 }

4184 #ifdef _SYSCALL32_IMPL
4185 /*
4186  * Return an array of structures with HAT memory map information.
4187  * We allocate here; the caller must deallocate.
4188  */
4189 int
4190 prgetxmap32(proc_t *p, list_t *iolhead)
4191 {
4192         struct as *as = p->p_as;
4193         prxmap32_t *mp;
4194         struct seg *seg;
4195         struct seg *brkseg, *stkseg;
4196         struct vnode *vp;
4197         struct vattr vattr;
4198         uint_t prot;

4200         ASSERT(as != &kas && AS_WRITE_HELD(as, &as->a_lock));

4202         /*
4203          * Request an initial buffer size that doesn't waste memory
4204          * if the address space has only a small number of segments.
4205          */
4206         pr_iol_initlist(iolhead, sizeof (*mp), avl_numnodes(&as->a_segtree));

4208         if ((seg = AS_SEGFIRST(as)) == NULL)
4209                 return (0);
```



```
4211         brkseg = break_seg(p);
4212         stkseg = as_segat(as, prgetstackbase(p));

4214         do {
4215                 caddr_t eaddr = seg->s_base + pr_getsegsize(seg, 0);
4216                 caddr_t saddr, naddr, baddr;
4217                 void *tmp = NULL;
4218                 ssize_t psz;
4219                 char *parr;
4220                 uint64_t npages;
4221                 uint64_t pagenum;

4223                 /*
4224                  * Segment loop part one: iterate from the base of the segment
4225                  * to its end, pausing at each address boundary (baddr) between
4226                  * ranges that have different virtual memory protections.
4227                  */
4228                 for (saddr = seg->s_base; saddr < eaddr; saddr = baddr) {
4229                         prot = pr_getprot(seg, 0, &tmp, &saddr, &baddr, eaddr);
4230                         ASSERT(baddr >= saddr && baddr <= eaddr);

4232                         /*
4233                          * Segment loop part two: iterate from the current
4234                          * position to the end of the protection boundary,
4235                          * pausing at each address boundary (naddr) between
4236                          * ranges that have different underlying page sizes.
4237                          */
4238                         for (; saddr < baddr; saddr = naddr) {
4239                                 psz = pr_getpagesize(seg, saddr, &naddr, baddr);
4240                                 ASSERT(naddr >= saddr && naddr <= baddr);

4242                                 mp = pr_iol_newbuf(iolhead, sizeof (*mp));

4244                                 mp->pr_vaddr = (caddr32_t)(uintptr_t)saddr;
4245                                 mp->pr_size = (size32_t)(naddr - saddr);
4246                                 mp->pr_offset = SEGOP_GETOFFSET(seg, saddr);
4247                                 mp->pr_mflags = 0;
4248                                 if (prot & PROT_READ)
4249                                         mp->pr_mflags |= MA_READ;
4250                                 if (prot & PROT_WRITE)
4251                                         mp->pr_mflags |= MA_WRITE;
4252                                 if (prot & PROT_EXEC)
4253                                         mp->pr_mflags |= MA_EXEC;
4254                                 if (SEGOP_GETTYPE(seg, saddr) & MAP_SHARED)
4255                                         mp->pr_mflags |= MA_SHARED;
4256                                 if (SEGOP_GETTYPE(seg, saddr) & MAP_NORESERVE)
4257                                         mp->pr_mflags |= MA_NORESERVE;
4258                                 if (seg->s_ops == &segspt_shmops ||
4259                                     (seg->s_ops == &segvn_ops &&
4260                                     (SEGOP_GETVP(seg, saddr, &vp) != 0 ||
4261                                     vp == NULL)))
4262                                         mp->pr_mflags |= MA_ANON;
4263                                 if (seg == brkseg)
4264                                         mp->pr_mflags |= MA_BREAK;
4265                                 else if (seg == stkseg)
4266                                         mp->pr_mflags |= MA_STACK;
4267                                 if (seg->s_ops == &segspt_shmops)
4268                                         mp->pr_mflags |= MA_ISM | MA_SHM;

4270                                 mp->pr_pagesize = PAGESIZE;
4271                                 if (psz == -1) {
4272                                         mp->pr_hatpagesize = 0;
4273                                 } else {
4274                                         mp->pr_hatpagesize = psz;
4275                                 }
```

```
4277                                /*
4278                                 * Manufacture a filename for the "object" dir.
4279                                 */
4280                                mp->pr_dev = PRNODEV32;
4281                                vattr.va_mask = AT_FSID|AT_NODEID;
4282                                if (seg->s_ops == &segvn_ops &&
4283                                    SEGOP_GETVP(seg, saddr, &vp) == 0 &&
4284                                    vp != NULL && vp->v_type == VREG &&
4285                                    VOP_GETATTR(vp, &vattr, 0, CRED(),
4286                                    NULL) == 0) {
4287                                        (void) cmpldev(&mp->pr_dev,
4288                                            vattr.va_fsid);
4289                                        mp->pr_ino = vattr.va_nodeid;
4290                                        if (vp == p->p_exec)
4291                                                (void) strcpy(mp->pr_mapname,
4292                                                    "a.out");
4293                                        else
4294                                                pr_object_name(mp->pr_mapname,
4295                                                    vp, &vattr);
4296                                }

4298                                /*
4299                                 * Get the SysV shared memory id, if any.
4300                                 */
4301                                if ((mp->pr_mflags & MA_SHARED) &&
4302                                    p->p_segacct && (mp->pr_shmid = shmgetid(p,
4303                                    seg->s_base)) != SHMID_NONE) {
4304                                        if (mp->pr_shmid == SHMID_FREE)
4305                                                mp->pr_shmid = -1;

4307                                        mp->pr_mflags |= MA_SHM;
4308                                } else {
4309                                        mp->pr_shmid = -1;
4310                                }

4312                                npages = ((uintptr_t)(naddr - saddr)) >>
4313                                    PAGESHIFT;
4314                                parr = kmem_zalloc(npages, KM_SLEEP);

4316                                SEGOP_INCORE(seg, saddr, naddr - saddr, parr);

4318                                for (pagenum = 0; pagenum < npages; pagenum++) {
4319                                        if (parr[pagenum] & SEG_PAGE_INCORE)
4320                                                mp->pr_rss++;
4321                                        if (parr[pagenum] & SEG_PAGE_ANON)
4322                                                mp->pr_anon++;
4323                                        if (parr[pagenum] & SEG_PAGE_LOCKED)
4324                                                mp->pr_locked++;
4325                                }
4326                                kmem_free(parr, npages);
4327                        }
4328                }
4329                ASSERT(tmp == NULL);
4330        } while ((seg = AS_SEGNEXT(as, seg)) != NULL);

4332        return (0);
4333 }
4334 #endif  /* _SYSCALL32_IMPL */
```

```
**********************************************************
   32725 Wed May 27 19:49:27 2015
new/usr/src/uts/common/os/cred.c
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
______unchanged_portion_omitted_



 162 /*
 163  * Initialize credentials data structures.
 164  */

 166 void
 167 cred_init(void)
 168 {
 169         priv_init();

 171         crsize = sizeof (cred_t);

 173         if (get_c2audit_load() > 0) {
 174 #ifdef _LP64
 175                 /* assure audit context is 64-bit aligned */
 176                 audoff = (crsize +
 177                     sizeof (int64_t) - 1) & ~(sizeof (int64_t) - 1);
 178 #else   /* _LP64 */
 179                 audoff = crsize;
 180 #endif  /* _LP64 */
 181                 crsize = audoff + sizeof (auditinfo_addr_t);
 182                 crsize = (crsize + sizeof (int) - 1) & ~(sizeof (int) - 1);
 183         }

 185         cred_cache = kmem_cache_create("cred_cache", crsize, 0,
 186             NULL, NULL, NULL, NULL, NULL, 0);

 188         /*
 189          * dummycr is used to copy initial state for creds.
 190          */
 191         dummycr = cralloc();
 192         bzero(dummycr, crsize);
 193         dummycr->cr_ref = 1;
 194         dummycr->cr_uid = (uid_t)-1;
 195         dummycr->cr_gid = (gid_t)-1;
 196         dummycr->cr_ruid = (uid_t)-1;
 197         dummycr->cr_rgid = (gid_t)-1;
 198         dummycr->cr_suid = (uid_t)-1;
 199         dummycr->cr_sgid = (gid_t)-1;


 202         /*
 203          * kcred is used by anything that needs all privileges; it's
 204          * also the template used for crget as it has all the compatible
 205          * sets filled in.
 206          */
 207         kcred = cralloc();

 209         bzero(kcred, crsize);
 210         kcred->cr_ref = 1;

 212         /* kcred is never freed, so we don't need zone_cred_hold here */
 213         kcred->cr_zone = &zone0;

 215         priv_fillset(&CR_LPRIV(kcred));
```



```
 216         CR_IPRIV(kcred) = *priv_basic;

 218         priv_addset(&CR_IPRIV(kcred), PRIV_PROC_SECFLAGS);

 220 #endif /* ! codereview */
 221         /* Not a basic privilege, if chown is not restricted add it to I0 */
 222         if (!rstchown)
 223                 priv_addset(&CR_IPRIV(kcred), PRIV_FILE_CHOWN_SELF);

 225         /* Basic privilege, if link is restricted remove it from I0 */
 226         if (rstlink)
 227                 priv_delset(&CR_IPRIV(kcred), PRIV_FILE_LINK_ANY);

 229         CR_EPRIV(kcred) = CR_PPRIV(kcred) = CR_IPRIV(kcred);

 231         CR_FLAGS(kcred) = NET_MAC_AWARE;

 233         /*
 234          * Set up credentials of p0.
 235          */
 236         ttoproc(curthread)->p_cred = kcred;
 237         curthread->t_cred = kcred;

 239         ucredsize = UCRED_SIZE;

 241         mutex_init(&ephemeral_zone_mutex, NULL, MUTEX_DEFAULT, NULL);
 242         zone_key_create(&ephemeral_zone_key, NULL, NULL, destroy_ephemeral_zsd);
 243 }

 245 /*
 246  * Allocate (nearly) uninitialized cred_t.
 247  */
 248 static cred_t *
 249 cralloc_flags(int flgs)
 250 {
 251         cred_t *cr = kmem_cache_alloc(cred_cache, flgs);

 253         if (cr == NULL)
 254                 return (NULL);

 256         cr->cr_ref = 1;         /* So we can crfree() */
 257         cr->cr_zone = NULL;
 258         cr->cr_label = NULL;
 259         cr->cr_ksid = NULL;
 260         cr->cr_klpd = NULL;
 261         cr->cr_grps = NULL;
 262         return (cr);
 263 }

 265 cred_t *
 266 cralloc(void)
 267 {
 268         return (cralloc_flags(KM_SLEEP));
 269 }

 271 /*
 272  * As cralloc but prepared for ksid change (if appropriate).
 273  */
 274 cred_t *
 275 cralloc_ksid(void)
 276 {
 277         cred_t *cr = cralloc();
 278         if (hasephids)
 279                 cr->cr_ksid = kcrsid_alloc();
 280         return (cr);
 281 }
```

```
283 /*
284  * Allocate a initialized cred structure and crhold() it.
285  * Initialized means: all ids 0, group count 0, L=Full, E=P=I=I0
286  */
287 cred_t *
288 crget(void)
289 {
290         cred_t *cr = kmem_cache_alloc(cred_cache, KM_SLEEP);

292         bcopy(kcred, cr, crsize);
293         cr->cr_ref = 1;
294         zone_cred_hold(cr->cr_zone);
295         if (cr->cr_label)
296                 label_hold(cr->cr_label);
297         ASSERT(cr->cr_klpd == NULL);
298         ASSERT(cr->cr_grps == NULL);
299         return (cr);
300 }

302 /*
303  * Broadcast the cred to all the threads in the process.
304  * The current thread's credentials can be set right away, but other
305  * threads must wait until the start of the next system call or trap.
306  * This avoids changing the cred in the middle of a system call.
307  *
308  * The cred has already been held for the process and the thread (2 holds),
309  * and p->p_cred set.
310  *
311  * p->p_crlock shouldn't be held here, since p_lock must be acquired.
312  */
313 void
314 crset(proc_t *p, cred_t *cr)
315 {
316         kthread_id_t    t;
317         kthread_id_t    first;
318         cred_t *oldcr;

320         ASSERT(p == curproc);   /* assumes p_lwpcnt can't change */

322         /*
323          * DTrace accesses t_cred in probe context.  t_cred must always be
324          * either NULL, or point to a valid, allocated cred structure.
325          */
326         t = curthread;
327         oldcr = t->t_cred;
328         t->t_cred = cr;         /* the cred is held by caller for this thread */
329         crfree(oldcr);          /* free the old cred for the thread */

331         /*
332          * Broadcast to other threads, if any.
333          */
334         if (p->p_lwpcnt > 1) {
335                 mutex_enter(&p->p_lock);        /* to keep thread list safe */
336                 first = curthread;
337                 for (t = first->t_forw; t != first; t = t->t_forw)
338                         t->t_pre_sys = 1; /* so syscall will get new cred */
339                 mutex_exit(&p->p_lock);
340         }
341 }

343 /*
344  * Put a hold on a cred structure.
345  */
346 void
347 crhold(cred_t *cr)
```



```
348 {
349         ASSERT(cr->cr_ref != 0xdeadbeef && cr->cr_ref != 0);
350         atomic_inc_32(&cr->cr_ref);
351 }

353 /*
354  * Release previous hold on a cred structure.  Free it if refcnt == 0.
355  * If cred uses label different from zone label, free it.
356  */
357 void
358 crfree(cred_t *cr)
359 {
360         ASSERT(cr->cr_ref != 0xdeadbeef && cr->cr_ref != 0);
361         if (atomic_dec_32_nv(&cr->cr_ref) == 0) {
362                 ASSERT(cr != kcred);
363                 if (cr->cr_label)
364                         label_rele(cr->cr_label);
365                 if (cr->cr_klpd)
366                         crklpd_rele(cr->cr_klpd);
367                 if (cr->cr_zone)
368                         zone_cred_rele(cr->cr_zone);
369                 if (cr->cr_ksid)
370                         kcrsid_rele(cr->cr_ksid);
371                 if (cr->cr_grps)
372                         crgrprele(cr->cr_grps);

374                 kmem_cache_free(cred_cache, cr);
375         }
376 }

378 /*
379  * Copy a cred structure to a new one and free the old one.
380  *      The new cred will have two references.  One for the calling process,
381  *      and one for the thread.
382  */
383 cred_t *
384 crcopy(cred_t *cr)
385 {
386         cred_t *newcr;

388         newcr = cralloc();
389         bcopy(cr, newcr, crsize);
390         if (newcr->cr_zone)
391                 zone_cred_hold(newcr->cr_zone);
392         if (newcr->cr_label)
393                 label_hold(newcr->cr_label);
394         if (newcr->cr_ksid)
395                 kcrsid_hold(newcr->cr_ksid);
396         if (newcr->cr_klpd)
397                 crklpd_hold(newcr->cr_klpd);
398         if (newcr->cr_grps)
399                 crgrphold(newcr->cr_grps);
400         crfree(cr);
401         newcr->cr_ref = 2;              /* caller gets two references */
402         return (newcr);
403 }

405 /*
406  * Copy a cred structure to a new one and free the old one.
407  *      The new cred will have two references.  One for the calling process,
408  *      and one for the thread.
409  * This variation on crcopy uses a pre-allocated structure for the
410  * "new" cred.
411  */
412 void
413 crcopy_to(cred_t *oldcr, cred_t *newcr)
```

```
414 {
415         credsid_t *nkcr = newcr->cr_ksid;

417         bcopy(oldcr, newcr, crsize);
418         if (newcr->cr_zone)
419                 zone_cred_hold(newcr->cr_zone);
420         if (newcr->cr_label)
421                 label_hold(newcr->cr_label);
422         if (newcr->cr_klpd)
423                 crklpd_hold(newcr->cr_klpd);
424         if (newcr->cr_grps)
425                 crgrphold(newcr->cr_grps);
426         if (nkcr) {
427                 newcr->cr_ksid = nkcr;
428                 kcrsidcopy_to(oldcr->cr_ksid, newcr->cr_ksid);
429         } else if (newcr->cr_ksid)
430                 kcrsid_hold(newcr->cr_ksid);
431         crfree(oldcr);
432         newcr->cr_ref = 2;              /* caller gets two references */
433 }

435 /*
436  * Dup a cred struct to a new held one.
437  *      The old cred is not freed.
438  */
439 static cred_t *
440 crdup_flags(const cred_t *cr, int flgs)
441 {
442         cred_t *newcr;

444         newcr = cralloc_flags(flgs);

446         if (newcr == NULL)
447                 return (NULL);

449         bcopy(cr, newcr, crsize);
450         if (newcr->cr_zone)
451                 zone_cred_hold(newcr->cr_zone);
452         if (newcr->cr_label)
453                 label_hold(newcr->cr_label);
454         if (newcr->cr_klpd)
455                 crklpd_hold(newcr->cr_klpd);
456         if (newcr->cr_ksid)
457                 kcrsid_hold(newcr->cr_ksid);
458         if (newcr->cr_grps)
459                 crgrphold(newcr->cr_grps);
460         newcr->cr_ref = 1;
461         return (newcr);
462 }

464 cred_t *
465 crdup(cred_t *cr)
466 {
467         return (crdup_flags(cr, KM_SLEEP));
468 }

470 /*
471  * Dup a cred struct to a new held one.
472  *      The old cred is not freed.
473  * This variation on crdup uses a pre-allocated structure for the
474  * "new" cred.
475  */
476 void
477 crdup_to(cred_t *oldcr, cred_t *newcr)
478 {
479         credsid_t *nkcr = newcr->cr_ksid;
```



```
481         bcopy(oldcr, newcr, crsize);
482         if (newcr->cr_zone)
483                 zone_cred_hold(newcr->cr_zone);
484         if (newcr->cr_label)
485                 label_hold(newcr->cr_label);
486         if (newcr->cr_klpd)
487                 crklpd_hold(newcr->cr_klpd);
488         if (newcr->cr_grps)
489                 crgrphold(newcr->cr_grps);
490         if (nkcr) {
491                 newcr->cr_ksid = nkcr;
492                 kcrsidcopy_to(oldcr->cr_ksid, newcr->cr_ksid);
493         } else if (newcr->cr_ksid)
494                 kcrsid_hold(newcr->cr_ksid);
495         newcr->cr_ref = 1;
496 }

498 /*
499  * Return the (held) credentials for the current running process.
500  */
501 cred_t *
502 crgetcred(void)
503 {
504         cred_t *cr;
505         proc_t *p;

507         p = ttoproc(curthread);
508         mutex_enter(&p->p_crlock);
509         crhold(cr = p->p_cred);
510         mutex_exit(&p->p_crlock);
511         return (cr);
512 }

514 /*
515  * Backward compatibility check for suser().
516  * Accounting flag is now set in the policy functions; auditing is
517  * done through use of privilege in the audit trail.
518  */
519 int
520 suser(cred_t *cr)
521 {
522         return (PRIV_POLICY(cr, PRIV_SYS_SUSER_COMPAT, B_FALSE, EPERM, NULL)
523             == 0);
524 }

526 /*
527  * Determine whether the supplied group id is a member of the group
528  * described by the supplied credentials.
529  */
530 int
531 groupmember(gid_t gid, const cred_t *cr)
532 {
533         if (gid == cr->cr_gid)
534                 return (1);
535         return (supgroupmember(gid, cr));
536 }

538 /*
539  * As groupmember but only check against the supplemental groups.
540  */
541 int
542 supgroupmember(gid_t gid, const cred_t *cr)
543 {
544         int hi, lo;
545         credgrp_t *grps = cr->cr_grps;
```

```
546         const gid_t *gp, *endgp;

548         if (grps == NULL)
549                 return (0);

551         /* For a small number of groups, use sequentials search. */
552         if (grps->crg_ngroups <= BIN_GROUP_SEARCH_CUTOFF) {
553                 endgp = &grps->crg_groups[grps->crg_ngroups];
554                 for (gp = grps->crg_groups; gp < endgp; gp++)
555                         if (*gp == gid)
556                                 return (1);
557                 return (0);
558         }

560         /* We use binary search when we have many groups. */
561         lo = 0;
562         hi = grps->crg_ngroups - 1;
563         gp = grps->crg_groups;

565         do {
566                 int m = (lo + hi) / 2;

568                 if (gid > gp[m])
569                         lo = m + 1;
570                 else if (gid < gp[m])
571                         hi = m - 1;
572                 else
573                         return (1);
574         } while (lo <= hi);

576         return (0);
577 }

579 /*
580  * This function is called to check whether the credentials set
581  * "scrp" has permission to act on credentials set "tcrp".  It enforces the
582  * permission requirements needed to send a signal to a process.
583  * The same requirements are imposed by other system calls, however.
584  *
585  * The rules are:
586  * (1) if the credentials are the same, the check succeeds
587  * (2) if the zone ids don't match, and scrp is not in the global zone or
588  *     does not have the PRIV_PROC_ZONE privilege, the check fails
589  * (3) if the real or effective user id of scrp matches the real or saved
590  *     user id of tcrp or scrp has the PRIV_PROC_OWNER privilege, the check
591  *     succeeds
592  * (4) otherwise, the check fails
593  */
594 int
595 hasprocperm(const cred_t *tcrp, const cred_t *scrp)
596 {
597         if (scrp == tcrp)
598                 return (1);
599         if (scrp->cr_zone != tcrp->cr_zone &&
600             (scrp->cr_zone != global_zone ||
601             secpolicy_proc_zone(scrp) != 0))
602                 return (0);
603         if (scrp->cr_uid == tcrp->cr_ruid ||
604             scrp->cr_ruid == tcrp->cr_ruid ||
605             scrp->cr_uid  == tcrp->cr_suid ||
606             scrp->cr_ruid == tcrp->cr_suid ||
607             !PRIV_POLICY(scrp, PRIV_PROC_OWNER, B_FALSE, EPERM, "hasprocperm"))
608                 return (1);
609         return (0);
610 }
```



```
612 /*
613  * This interface replaces hasprocperm; it works like hasprocperm but
614  * additionally returns success if the proc_t's match
615  * It is the preferred interface for most uses.
616  * And it will acquire p_crlock itself, so it assert's that it shouldn't
617  * be held.
618  */
619 int
620 prochasprocperm(proc_t *tp, proc_t *sp, const cred_t *scrp)
621 {
622         int rets;
623         cred_t *tcrp;

625         ASSERT(MUTEX_NOT_HELD(&tp->p_crlock));

627         if (tp == sp)
628                 return (1);

630         if (tp->p_sessp != sp->p_sessp && secpolicy_basic_proc(scrp) != 0)
631                 return (0);

633         mutex_enter(&tp->p_crlock);
634         crhold(tcrp = tp->p_cred);
635         mutex_exit(&tp->p_crlock);
636         rets = hasprocperm(tcrp, scrp);
637         crfree(tcrp);

639         return (rets);
640 }

642 /*
643  * This routine is used to compare two credentials to determine if
644  * they refer to the same "user".  If the pointers are equal, then
645  * they must refer to the same user.  Otherwise, the contents of
646  * the credentials are compared to see whether they are equivalent.
647  *
648  * This routine returns 0 if the credentials refer to the same user,
649  * 1 if they do not.
650  */
651 int
652 crcmp(const cred_t *cr1, const cred_t *cr2)
653 {
654         credgrp_t *grp1, *grp2;

656         if (cr1 == cr2)
657                 return (0);

659         if (cr1->cr_uid == cr2->cr_uid &&
660             cr1->cr_gid == cr2->cr_gid &&
661             cr1->cr_ruid == cr2->cr_ruid &&
662             cr1->cr_rgid == cr2->cr_rgid &&
663             cr1->cr_zone == cr2->cr_zone &&
664             ((grp1 = cr1->cr_grps) == (grp2 = cr2->cr_grps) ||
665             (grp1 != NULL && grp2 != NULL &&
666             grp1->crg_ngroups == grp2->crg_ngroups &&
667             bcmp(grp1->crg_groups, grp2->crg_groups,
668             grp1->crg_ngroups * sizeof (gid_t)) == 0))) {
669                 return (!priv_isequalset(&CR_OEPRIV(cr1), &CR_OEPRIV(cr2)));
670         }
671         return (1);
672 }

674 /*
675  * Read access functions to cred_t.
676  */
677 uid_t
```

```
678 crgetuid(const cred_t *cr)
679 {
680         return (cr->cr_uid);
681 }

683 uid_t
684 crgetruid(const cred_t *cr)
685 {
686         return (cr->cr_ruid);
687 }

689 uid_t
690 crgetsuid(const cred_t *cr)
691 {
692         return (cr->cr_suid);
693 }

695 gid_t
696 crgetgid(const cred_t *cr)
697 {
698         return (cr->cr_gid);
699 }

701 gid_t
702 crgetrgid(const cred_t *cr)
703 {
704         return (cr->cr_rgid);
705 }

707 gid_t
708 crgetsgid(const cred_t *cr)
709 {
710         return (cr->cr_sgid);
711 }

713 const auditinfo_addr_t *
714 crgetauinfo(const cred_t *cr)
715 {
716         return ((const auditinfo_addr_t *)CR_AUINFO(cr));
717 }

719 auditinfo_addr_t *
720 crgetauinfo_modifiable(cred_t *cr)
721 {
722         return (CR_AUINFO(cr));
723 }

725 zoneid_t
726 crgetzoneid(const cred_t *cr)
727 {
728         return (cr->cr_zone == NULL ?
729             (cr->cr_uid == -1 ? (zoneid_t)-1 : GLOBAL_ZONEID) :
730             cr->cr_zone->zone_id);
731 }

733 projid_t
734 crgetprojid(const cred_t *cr)
735 {
736         return (cr->cr_projid);
737 }

739 zone_t *
740 crgetzone(const cred_t *cr)
741 {
742         return (cr->cr_zone);
743 }
```



```
745 struct ts_label_s *
746 crgetlabel(const cred_t *cr)
747 {
748         return (cr->cr_label ?
749             cr->cr_label :
750             (cr->cr_zone ? cr->cr_zone->zone_slabel : NULL));
751 }

753 boolean_t
754 crisremote(const cred_t *cr)
755 {
756         return (REMOTE_PEER_CRED(cr));
757 }

759 #define BADUID(x, zn)   ((x) != -1 && !VALID_UID((x), (zn)))
760 #define BADGID(x, zn)   ((x) != -1 && !VALID_GID((x), (zn)))

762 int
763 crsetresuid(cred_t *cr, uid_t r, uid_t e, uid_t s)
764 {
765         zone_t  *zone = crgetzone(cr);

767         ASSERT(cr->cr_ref <= 2);

769         if (BADUID(r, zone) || BADUID(e, zone) || BADUID(s, zone))
770                 return (-1);

772         if (r != -1)
773                 cr->cr_ruid = r;
774         if (e != -1)
775                 cr->cr_uid = e;
776         if (s != -1)
777                 cr->cr_suid = s;

779         return (0);
780 }

782 int
783 crsetresgid(cred_t *cr, gid_t r, gid_t e, gid_t s)
784 {
785         zone_t  *zone = crgetzone(cr);

787         ASSERT(cr->cr_ref <= 2);

789         if (BADGID(r, zone) || BADGID(e, zone) || BADGID(s, zone))
790                 return (-1);

792         if (r != -1)
793                 cr->cr_rgid = r;
794         if (e != -1)
795                 cr->cr_gid = e;
796         if (s != -1)
797                 cr->cr_sgid = s;

799         return (0);
800 }

802 int
803 crsetugid(cred_t *cr, uid_t uid, gid_t gid)
804 {
805         zone_t  *zone = crgetzone(cr);

807         ASSERT(cr->cr_ref <= 2);

809         if (!VALID_UID(uid, zone) || !VALID_GID(gid, zone))
```

```
810                 return (-1);

812         cr->cr_uid = cr->cr_ruid = cr->cr_suid = uid;
813         cr->cr_gid = cr->cr_rgid = cr->cr_sgid = gid;

815         return (0);
816 }

818 static int
819 gidcmp(const void *v1, const void *v2)
820 {
821         gid_t g1 = *(gid_t *)v1;
822         gid_t g2 = *(gid_t *)v2;

824         if (g1 < g2)
825                 return (-1);
826         else if (g1 > g2)
827                 return (1);
828         else
829                 return (0);
830 }

832 int
833 crsetgroups(cred_t *cr, int n, gid_t *grp)
834 {
835         ASSERT(cr->cr_ref <= 2);

837         if (n > ngroups_max || n < 0)
838                 return (-1);

840         if (cr->cr_grps != NULL)
841                 crgrprele(cr->cr_grps);

843         if (n > 0) {
844                 cr->cr_grps = kmem_alloc(CREDGRPSZ(n), KM_SLEEP);
845                 bcopy(grp, cr->cr_grps->crg_groups, n * sizeof (gid_t));
846                 cr->cr_grps->crg_ref = 1;
847                 cr->cr_grps->crg_ngroups = n;
848                 qsort(cr->cr_grps->crg_groups, n, sizeof (gid_t), gidcmp);
849         } else {
850                 cr->cr_grps = NULL;
851         }

853         return (0);
854 }

856 void
857 crsetprojid(cred_t *cr, projid_t projid)
858 {
859         ASSERT(projid >= 0 && projid <= MAXPROJID);
860         cr->cr_projid = projid;
861 }

863 /*
864  * This routine returns the pointer to the first element of the crg_groups
865  * array.  It can move around in an implementation defined way.
866  * Note that when we have no grouplist, we return one element but the
867  * caller should never reference it.
868  */
869 const gid_t *
870 crgetgroups(const cred_t *cr)
871 {
872         return (cr->cr_grps == NULL ? &cr->cr_gid : cr->cr_grps->crg_groups);
873 }

875 int
```



```
876 crgetngroups(const cred_t *cr)
877 {
878         return (cr->cr_grps == NULL ? 0 : cr->cr_grps->crg_ngroups);
879 }

881 void
882 cred2prcred(const cred_t *cr, prcred_t *pcrp)
883 {
884         pcrp->pr_euid = cr->cr_uid;
885         pcrp->pr_ruid = cr->cr_ruid;
886         pcrp->pr_suid = cr->cr_suid;
887         pcrp->pr_egid = cr->cr_gid;
888         pcrp->pr_rgid = cr->cr_rgid;
889         pcrp->pr_sgid = cr->cr_sgid;
890         pcrp->pr_groups[0] = 0; /* in case ngroups == 0 */
891         pcrp->pr_ngroups = cr->cr_grps == NULL ? 0 : cr->cr_grps->crg_ngroups;

893         if (pcrp->pr_ngroups != 0)
894                 bcopy(cr->cr_grps->crg_groups, pcrp->pr_groups,
895                     sizeof (gid_t) * pcrp->pr_ngroups);
896 }

898 static int
899 cred2ucaud(const cred_t *cr, auditinfo64_addr_t *ainfo, const cred_t *rcr)
900 {
901         auditinfo_addr_t        *ai;
902         au_tid_addr_t   tid;

904         if (secpolicy_audit_getattr(rcr, B_TRUE) != 0)
905                 return (-1);

907         ai = CR_AUINFO(cr);     /* caller makes sure this is non-NULL */
908         tid = ai->ai_termid;

910         ainfo->ai_auid = ai->ai_auid;
911         ainfo->ai_mask = ai->ai_mask;
912         ainfo->ai_asid = ai->ai_asid;

914         ainfo->ai_termid.at_type = tid.at_type;
915         bcopy(&tid.at_addr, &ainfo->ai_termid.at_addr, 4 * sizeof (uint_t));

917         ainfo->ai_termid.at_port.at_major = (uint32_t)getmajor(tid.at_port);
918         ainfo->ai_termid.at_port.at_minor = (uint32_t)getminor(tid.at_port);

920         return (0);
921 }

923 void
924 cred2uclabel(const cred_t *cr, bslabel_t *labelp)
925 {
926         ts_label_t      *tslp;

928         if ((tslp = crgetlabel(cr)) != NULL)
929                 bcopy(&tslp->tsl_label, labelp, sizeof (bslabel_t));
930 }

932 /*
933  * Convert a credential into a "ucred".  Allow the caller to specify
934  * and aligned buffer, e.g., in an mblk, so we don't have to allocate
935  * memory and copy it twice.
936  *
937  * This function may call cred2ucaud(), which calls CRED(). Since this
938  * can be called from an interrupt thread, receiver's cred (rcr) is needed
939  * to determine whether audit info should be included.
940  */
941 struct ucred_s *
```

```
 942 cred2ucred(const cred_t *cr, pid_t pid, void *buf, const cred_t *rcr)
 943 {
 944         struct ucred_s *uc;
 945         uint32_t realsz = ucredminsize(cr);
 946         ts_label_t *tslp = is_system_labeled() ? crgetlabel(cr) : NULL;

 948         /* The structure isn't always completely filled in, so zero it */
 949         if (buf == NULL) {
 950                 uc = kmem_zalloc(realsz, KM_SLEEP);
 951         } else {
 952                 bzero(buf, realsz);
 953                 uc = buf;
 954         }
 955         uc->uc_size = realsz;
 956         uc->uc_pid = pid;
 957         uc->uc_projid = cr->cr_projid;
 958         uc->uc_zoneid = crgetzoneid(cr);

 960         if (REMOTE_PEER_CRED(cr)) {
 961                 /*
 962                  * Other than label, the rest of cred info about a
 963                  * remote peer isn't available. Copy the label directly
 964                  * after the header where we generally copy the prcred.
 965                  * That's why we use sizeof (struct ucred_s).  The other
 966                  * offset fields are initialized to 0.
 967                  */
 968                 uc->uc_labeloff = tslp == NULL ? 0 : sizeof (struct ucred_s);
 969         } else {
 970                 uc->uc_credoff = UCRED_CRED_OFF;
 971                 uc->uc_privoff = UCRED_PRIV_OFF;
 972                 uc->uc_audoff = UCRED_AUD_OFF;
 973                 uc->uc_labeloff = tslp == NULL ? 0 : UCRED_LABEL_OFF;

 975                 cred2prcred(cr, UCCRED(uc));
 976                 cred2prpriv(cr, UCPRIV(uc));

 978                 if (audoff == 0 || cred2ucaud(cr, UCAUD(uc), rcr) != 0)
 979                         uc->uc_audoff = 0;
 980         }
 981         if (tslp != NULL)
 982                 bcopy(&tslp->tsl_label, UCLABEL(uc), sizeof (bslabel_t));

 984         return (uc);
 985 }

 987 /*
 988  * Don't allocate the non-needed group entries.  Note: this function
 989  * must match the code in cred2ucred; they must agree about the
 990  * minimal size of the ucred.
 991  */
 992 uint32_t
 993 ucredminsize(const cred_t *cr)
 994 {
 995         int ndiff;

 997         if (cr == NULL)
 998                 return (ucredsize);

1000         if (REMOTE_PEER_CRED(cr)) {
1001                 if (is_system_labeled())
1002                         return (sizeof (struct ucred_s) + sizeof (bslabel_t));
1003                 else
1004                         return (sizeof (struct ucred_s));
1005         }

1007         if (cr->cr_grps == NULL)
```



```
1008                 ndiff = ngroups_max - 1;        /* Needs one for prcred_t */
1009         else
1010                 ndiff = ngroups_max - cr->cr_grps->crg_ngroups;

1012         return (ucredsize - ndiff * sizeof (gid_t));
1013 }

1015 /*
1016  * Get the "ucred" of a process.
1017  */
1018 struct ucred_s *
1019 pgetucred(proc_t *p)
1020 {
1021         cred_t *cr;
1022         struct ucred_s *uc;

1024         mutex_enter(&p->p_crlock);
1025         cr = p->p_cred;
1026         crhold(cr);
1027         mutex_exit(&p->p_crlock);

1029         uc = cred2ucred(cr, p->p_pid, NULL, CRED());
1030         crfree(cr);

1032         return (uc);
1033 }

1035 /*
1036  * If the reply status is NFSERR_EACCES, it may be because we are
1037  * root (no root net access).  Check the real uid, if it isn't root
1038  * make that the uid instead and retry the call.
1039  * Private interface for NFS.
1040  */
1041 cred_t *
1042 crnetadjust(cred_t *cr)
1043 {
1044         if (cr->cr_uid == 0 && cr->cr_ruid != 0) {
1045                 cr = crdup(cr);
1046                 cr->cr_uid = cr->cr_ruid;
1047                 return (cr);
1048         }
1049         return (NULL);
1050 }

1052 /*
1053  * The reference count is of interest when you want to check
1054  * whether it is ok to modify the credential in place.
1055  */
1056 uint_t
1057 crgetref(const cred_t *cr)
1058 {
1059         return (cr->cr_ref);
1060 }

1062 static int
1063 get_c2audit_load(void)
1064 {
1065         static int      gotit = 0;
1066         static int      c2audit_load;

1068         if (gotit)
1069                 return (c2audit_load);
1070         c2audit_load = 1;               /* set default value once */
1071         if (mod_sysctl(SYS_CHECK_EXCLUDE, "c2audit") != 0)
1072                 c2audit_load = 0;
1073         gotit++;
```

```
1075         return (c2audit_load);
1076 }

1078 int
1079 get_audit_ucrsize(void)
1080 {
1081         return (get_c2audit_load() ? sizeof (auditinfo64_addr_t) : 0);
1082 }

1084 /*
1085  * Set zone pointer in credential to indicated value.  First adds a
1086  * hold for the new zone, then drops the hold on previous zone (if any).
1087  * This is done in this order in case the old and new zones are the
1088  * same.
1089  */
1090 void
1091 crsetzone(cred_t *cr, zone_t *zptr)
1092 {
1093         zone_t *oldzptr = cr->cr_zone;

1095         ASSERT(cr != kcred);
1096         ASSERT(cr->cr_ref <= 2);
1097         cr->cr_zone = zptr;
1098         zone_cred_hold(zptr);
1099         if (oldzptr)
1100                 zone_cred_rele(oldzptr);
1101 }

1103 /*
1104  * Create a new cred based on the supplied label
1105  */
1106 cred_t *
1107 newcred_from_bslabel(bslabel_t *blabel, uint32_t doi, int flags)
1108 {
1109         ts_label_t *lbl = labelalloc(blabel, doi, flags);
1110         cred_t *cr = NULL;

1112         if (lbl != NULL) {
1113                 if ((cr = crdup_flags(dummycr, flags)) != NULL) {
1114                         cr->cr_label = lbl;
1115                 } else {
1116                         label_rele(lbl);
1117                 }
1118         }

1120         return (cr);
1121 }

1123 /*
1124  * Derive a new cred from the existing cred, but with a different label.
1125  * To be used when a cred is being shared, but the label needs to be changed
1126  * by a caller without affecting other users
1127  */
1128 cred_t *
1129 copycred_from_tslabel(const cred_t *cr, ts_label_t *label, int flags)
1130 {
1131         cred_t *newcr = NULL;

1133         if ((newcr = crdup_flags(cr, flags)) != NULL) {
1134                 if (newcr->cr_label != NULL)
1135                         label_rele(newcr->cr_label);
1136                 label_hold(label);
1137                 newcr->cr_label = label;
1138         }
```



```
1140         return (newcr);
1141 }

1143 /*
1144  * Derive a new cred from the existing cred, but with a different label.
1145  */
1146 cred_t *
1147 copycred_from_bslabel(const cred_t *cr, bslabel_t *blabel,
1148     uint32_t doi, int flags)
1149 {
1150         ts_label_t *lbl = labelalloc(blabel, doi, flags);
1151         cred_t  *newcr = NULL;

1153         if (lbl != NULL) {
1154                 newcr = copycred_from_tslabel(cr, lbl, flags);
1155                 label_rele(lbl);
1156         }

1158         return (newcr);
1159 }

1161 /*
1162  * This function returns a pointer to the kcred-equivalent in the current zone.
1163  */
1164 cred_t *
1165 zone_kcred(void)
1166 {
1167         zone_t *zone;

1169         if ((zone = CRED()->cr_zone) != NULL)
1170                 return (zone->zone_kcred);
1171         else
1172                 return (kcred);
1173 }

1175 boolean_t
1176 valid_ephemeral_uid(zone_t *zone, uid_t id)
1177 {
1178         ephemeral_zsd_t *eph_zsd;
1179         if (id <= IDMAP_WK__MAX_UID)
1180                 return (B_TRUE);

1182         eph_zsd = get_ephemeral_zsd(zone);
1183         ASSERT(eph_zsd != NULL);
1184         membar_consumer();
1185         return (id > eph_zsd->min_uid && id <= eph_zsd->last_uid);
1186 }

1188 boolean_t
1189 valid_ephemeral_gid(zone_t *zone, gid_t id)
1190 {
1191         ephemeral_zsd_t *eph_zsd;
1192         if (id <= IDMAP_WK__MAX_GID)
1193                 return (B_TRUE);

1195         eph_zsd = get_ephemeral_zsd(zone);
1196         ASSERT(eph_zsd != NULL);
1197         membar_consumer();
1198         return (id > eph_zsd->min_gid && id <= eph_zsd->last_gid);
1199 }

1201 int
1202 eph_uid_alloc(zone_t *zone, int flags, uid_t *start, int count)
1203 {
1204         ephemeral_zsd_t *eph_zsd = get_ephemeral_zsd(zone);
```

```
1206         ASSERT(eph_zsd != NULL);

1208         mutex_enter(&eph_zsd->eph_lock);

1210         /* Test for unsigned integer wrap around */
1211         if (eph_zsd->last_uid + count < eph_zsd->last_uid) {
1212                 mutex_exit(&eph_zsd->eph_lock);
1213                 return (-1);
1214         }

1216         /* first call or idmap crashed and state corrupted */
1217         if (flags != 0)
1218                 eph_zsd->min_uid = eph_zsd->last_uid;

1220         hasephids = B_TRUE;
1221         *start = eph_zsd->last_uid + 1;
1222         atomic_add_32(&eph_zsd->last_uid, count);
1223         mutex_exit(&eph_zsd->eph_lock);
1224         return (0);
1225 }

1227 int
1228 eph_gid_alloc(zone_t *zone, int flags, gid_t *start, int count)
1229 {
1230         ephemeral_zsd_t *eph_zsd = get_ephemeral_zsd(zone);

1232         ASSERT(eph_zsd != NULL);

1234         mutex_enter(&eph_zsd->eph_lock);

1236         /* Test for unsigned integer wrap around */
1237         if (eph_zsd->last_gid + count < eph_zsd->last_gid) {
1238                 mutex_exit(&eph_zsd->eph_lock);
1239                 return (-1);
1240         }

1242         /* first call or idmap crashed and state corrupted */
1243         if (flags != 0)
1244                 eph_zsd->min_gid = eph_zsd->last_gid;

1246         hasephids = B_TRUE;
1247         *start = eph_zsd->last_gid + 1;
1248         atomic_add_32(&eph_zsd->last_gid, count);
1249         mutex_exit(&eph_zsd->eph_lock);
1250         return (0);
1251 }

1253 /*
1254  * IMPORTANT.The two functions get_ephemeral_data() and set_ephemeral_data()
1255  * are project private functions that are for use of the test system only and
1256  * are not to be used for other purposes.
1257  */

1259 void
1260 get_ephemeral_data(zone_t *zone, uid_t *min_uid, uid_t *last_uid,
1261         gid_t *min_gid, gid_t *last_gid)
1262 {
1263         ephemeral_zsd_t *eph_zsd = get_ephemeral_zsd(zone);

1265         ASSERT(eph_zsd != NULL);

1267         mutex_enter(&eph_zsd->eph_lock);

1269         *min_uid = eph_zsd->min_uid;
1270         *last_uid = eph_zsd->last_uid;
1271         *min_gid = eph_zsd->min_gid;
```



```
1272         *last_gid = eph_zsd->last_gid;

1274         mutex_exit(&eph_zsd->eph_lock);
1275 }


1278 void
1279 set_ephemeral_data(zone_t *zone, uid_t min_uid, uid_t last_uid,
1280         gid_t min_gid, gid_t last_gid)
1281 {
1282         ephemeral_zsd_t *eph_zsd = get_ephemeral_zsd(zone);

1284         ASSERT(eph_zsd != NULL);

1286         mutex_enter(&eph_zsd->eph_lock);

1288         if (min_uid != 0)
1289                 eph_zsd->min_uid = min_uid;
1290         if (last_uid != 0)
1291                 eph_zsd->last_uid = last_uid;
1292         if (min_gid != 0)
1293                 eph_zsd->min_gid = min_gid;
1294         if (last_gid != 0)
1295                 eph_zsd->last_gid = last_gid;

1297         mutex_exit(&eph_zsd->eph_lock);
1298 }

1300 /*
1301  * If the credential user SID or group SID is mapped to an ephemeral
1302  * ID, map the credential to nobody.
1303  */
1304 cred_t *
1305 crgetmapped(const cred_t *cr)
1306 {
1307         ephemeral_zsd_t *eph_zsd;
1308         /*
1309          * Someone incorrectly passed a NULL cred to a vnode operation
1310          * either on purpose or by calling CRED() in interrupt context.
1311          */
1312         if (cr == NULL)
1313                 return (NULL);

1315         if (cr->cr_ksid != NULL) {
1316                 if (cr->cr_ksid->kr_sidx[KSID_USER].ks_id > MAXUID) {
1317                         eph_zsd = get_ephemeral_zsd(crgetzone(cr));
1318                         return (eph_zsd->eph_nobody);
1319                 }

1321                 if (cr->cr_ksid->kr_sidx[KSID_GROUP].ks_id > MAXUID) {
1322                         eph_zsd = get_ephemeral_zsd(crgetzone(cr));
1323                         return (eph_zsd->eph_nobody);
1324                 }
1325         }

1327         return ((cred_t *)cr);
1328 }

1330 /* index should be in range for a ksidindex_t */
1331 void
1332 crsetsid(cred_t *cr, ksid_t *ksp, int index)
1333 {
1334         ASSERT(cr->cr_ref <= 2);
1335         ASSERT(index >= 0 && index < KSID_COUNT);
1336         if (cr->cr_ksid == NULL && ksp == NULL)
1337                 return;
```

```
1338         cr->cr_ksid = kcrsid_setsid(cr->cr_ksid, ksp, index);
1339 }

1341 void
1342 crsetsidlist(cred_t *cr, ksidlist_t *ksl)
1343 {
1344         ASSERT(cr->cr_ref <= 2);
1345         if (cr->cr_ksid == NULL && ksl == NULL)
1346                 return;
1347         cr->cr_ksid = kcrsid_setsidlist(cr->cr_ksid, ksl);
1348 }

1350 ksid_t *
1351 crgetsid(const cred_t *cr, int i)
1352 {
1353         ASSERT(i >= 0 && i < KSID_COUNT);
1354         if (cr->cr_ksid != NULL && cr->cr_ksid->kr_sidx[i].ks_domain)
1355                 return ((ksid_t *)&cr->cr_ksid->kr_sidx[i]);
1356         return (NULL);
1357 }

1359 ksidlist_t *
1360 crgetsidlist(const cred_t *cr)
1361 {
1362         if (cr->cr_ksid != NULL)
1363                 return (cr->cr_ksid->kr_sidlist);
1364         return (NULL);
1365 }

1367 /*
1368  * Interface to set the effective and permitted privileges for
1369  * a credential; this interface does no security checks and is
1370  * intended for kernel (file)servers creating credentials with
1371  * specific privileges.
1372  */
1373 int
1374 crsetpriv(cred_t *cr, ...)
1375 {
1376         va_list ap;
1377         const char *privnm;

1379         ASSERT(cr->cr_ref <= 2);

1381         priv_set_PA(cr);

1383         va_start(ap, cr);

1385         while ((privnm = va_arg(ap, const char *)) != NULL) {
1386                 int priv = priv_getbyname(privnm, 0);
1387                 if (priv < 0)
1388                         return (-1);

1390                 priv_addset(&CR_PPRIV(cr), priv);
1391                 priv_addset(&CR_EPRIV(cr), priv);
1392         }
1393         priv_adjust_PA(cr);
1394         va_end(ap);
1395         return (0);
1396 }

1398 /*
1399  * Interface to effectively set the PRIV_ALL for
1400  * a credential; this interface does no security checks and is
1401  * intended for kernel (file)servers to extend the user credentials
1402  * to be ALL, like either kcred or zcred.
1403  */
```



```
1404 void
1405 crset_zone_privall(cred_t *cr)
1406 {
1407         zone_t  *zone = crgetzone(cr);

1409         priv_fillset(&CR_LPRIV(cr));
1410         CR_EPRIV(cr) = CR_PPRIV(cr) = CR_IPRIV(cr) = CR_LPRIV(cr);
1411         priv_intersect(zone->zone_privset, &CR_LPRIV(cr));
1412         priv_intersect(zone->zone_privset, &CR_EPRIV(cr));
1413         priv_intersect(zone->zone_privset, &CR_IPRIV(cr));
1414         priv_intersect(zone->zone_privset, &CR_PPRIV(cr));
1415 }

1417 struct credklpd *
1418 crgetcrklpd(const cred_t *cr)
1419 {
1420         return (cr->cr_klpd);
1421 }

1423 void
1424 crsetcrklpd(cred_t *cr, struct credklpd *crklpd)
1425 {
1426         ASSERT(cr->cr_ref <= 2);

1428         if (cr->cr_klpd != NULL)
1429                 crklpd_rele(cr->cr_klpd);
1430         cr->cr_klpd = crklpd;
1431 }

1433 credgrp_t *
1434 crgrpcopyin(int n, gid_t *gidset)
1435 {
1436         credgrp_t *mem;
1437         size_t sz = CREDGRPSZ(n);

1439         ASSERT(n > 0);

1441         mem = kmem_alloc(sz, KM_SLEEP);

1443         if (copyin(gidset, mem->crg_groups, sizeof (gid_t) * n)) {
1444                 kmem_free(mem, sz);
1445                 return (NULL);
1446         }
1447         mem->crg_ref = 1;
1448         mem->crg_ngroups = n;
1449         qsort(mem->crg_groups, n, sizeof (gid_t), gidcmp);
1450         return (mem);
1451 }

1453 const gid_t *
1454 crgetggroups(const credgrp_t *grps)
1455 {
1456         return (grps->crg_groups);
1457 }

1459 void
1460 crsetcredgrp(cred_t *cr, credgrp_t *grps)
1461 {
1462         ASSERT(cr->cr_ref <= 2);

1464         if (cr->cr_grps != NULL)
1465                 crgrprele(cr->cr_grps);

1467         cr->cr_grps = grps;
1468 }
```

```
1470 void
1471 crgrprele(credgrp_t *grps)
1472 {
1473         if (atomic_dec_32_nv(&grps->crg_ref) == 0)
1474                 kmem_free(grps, CREDGRPSZ(grps->crg_ngroups));
1475 }

1477 static void
1478 crgrphold(credgrp_t *grps)
1479 {
1480         atomic_inc_32(&grps->crg_ref);
1481 }
```

```
**********************************************************
   53826 Wed May 27 19:49:28 2015
new/usr/src/uts/common/os/exec.c
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
```

```
   1 /*
   2  * CDDL HEADER START
   3  *
   4  * The contents of this file are subject to the terms of the
   5  * Common Development and Distribution License (the "License").
   6  * You may not use this file except in compliance with the License.
   7  *
   8  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
   9  * or http://www.opensolaris.org/os/licensing.
  10  * See the License for the specific language governing permissions
  11  * and limitations under the License.
  12  *
  13  * When distributing Covered Code, include this CDDL HEADER in each
  14  * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  15  * If applicable, add the following below this CDDL HEADER, with the
  16  * fields enclosed by brackets "[]" replaced with your own identifying
  17  * information: Portions Copyright [yyyy] [name of copyright owner]
  18  *
  19  * CDDL HEADER END
  20  */

  22 /*
  23  * Copyright (c) 1988, 2010, Oracle and/or its affiliates. All rights reserved.
  24  */

  26 /*      Copyright (c) 1988 AT&T */
  27 /*        All Rights Reserved   */
  28 /*
  29  * Copyright 2014, Joyent, Inc.  All rights reserved.
  30  */

  32 #include <sys/types.h>
  33 #include <sys/param.h>
  34 #include <sys/sysmacros.h>
  35 #include <sys/systm.h>
  36 #include <sys/signal.h>
  37 #include <sys/cred_impl.h>
  38 #include <sys/policy.h>
  39 #include <sys/user.h>
  40 #include <sys/errno.h>
  41 #include <sys/file.h>
  42 #include <sys/vfs.h>
  43 #include <sys/vnode.h>
  44 #include <sys/mman.h>
  45 #include <sys/acct.h>
  46 #include <sys/cpuvar.h>
  47 #include <sys/proc.h>
  48 #include <sys/cmn_err.h>
  49 #include <sys/debug.h>
  50 #include <sys/pathname.h>
  51 #include <sys/vm.h>
  52 #include <sys/lgrp.h>
  53 #include <sys/vtrace.h>
  54 #include <sys/exec.h>
  55 #include <sys/exechdr.h>
  56 #include <sys/kmem.h>
  57 #include <sys/prsystm.h>
  58 #include <sys/modctl.h>
```



```
  59 #include <sys/vmparam.h>
  60 #include <sys/door.h>
  61 #include <sys/schedctl.h>
  62 #include <sys/utrap.h>
  63 #include <sys/systeminfo.h>
  64 #include <sys/stack.h>
  65 #include <sys/rctl.h>
  66 #include <sys/dtrace.h>
  67 #include <sys/lwpchan_impl.h>
  68 #include <sys/pool.h>
  69 #include <sys/sdt.h>
  70 #include <sys/brand.h>
  71 #include <sys/klpd.h>
  72 #include <sys/random.h>
  73 #endif /* ! codereview */

  75 #include <c2/audit.h>

  77 #include <vm/hat.h>
  78 #include <vm/anon.h>
  79 #include <vm/as.h>
  80 #include <vm/seg.h>
  81 #include <vm/seg_vn.h>

  83 #define PRIV_RESET              0x01    /* needs to reset privs */
  84 #define PRIV_SETID              0x02    /* needs to change uids */
  85 #define PRIV_SETUGID            0x04    /* is setuid/setgid/forced privs */
  86 #define PRIV_INCREASE           0x08    /* child runs with more privs */
  87 #define MAC_FLAGS               0x10    /* need to adjust MAC flags */
  88 #define PRIV_FORCED             0x20    /* has forced privileges */

  90 static int execsetid(struct vnode *, struct vattr *, uid_t *, uid_t *,
  91     priv_set_t *, cred_t *, const char *);
  92 static int hold_execsw(struct execsw *);

  94 uint_t auxv_hwcap = 0;  /* auxv AT_SUN_HWCAP value; determined on the fly */
  95 uint_t auxv_hwcap_2 = 0;        /* AT_SUN_HWCAP2 */
  96 #if defined(_SYSCALL32_IMPL)
  97 uint_t auxv_hwcap32 = 0;        /* 32-bit version of auxv_hwcap */
  98 uint_t auxv_hwcap32_2 = 0;      /* 32-bit version of auxv_hwcap2 */
  99 #endif

 101 #define PSUIDFLAGS              (SNOCD|SUGID)

 103 /*
 104  * These are consumed within the specific exec modules, but are defined here bec
 105  *
 106  * 1) The exec modules are unloadable, which would make this near useless.
 107  *
 108  * 2) We want them to be common across all of them, should more than ELF come
 109  *    to support them.
 110  *
 111  * All must be powers of 2.
 112  */
 113 volatile size_t aslr_max_brk_skew = 16 * 1024 * 1024; /* 16MB */
 114 #pragma weak exec_stackgap = aslr_max_stack_skew      /* Old, compatible name */
 115 volatile size_t aslr_max_stack_skew = 64 * 1024;      /* 64KB */

 117 /*
 118 #endif /* ! codereview */
 119  * exece() - system call wrapper around exec_common()
 120  */
 121 int
 122 exece(const char *fname, const char **argp, const char **envp)
 123 {
 124         int error;
```

```
126         error = exec_common(fname, argp, envp, EBA_NONE);
127         return (error ? (set_errno(error)) : 0);
128 }

130 int
131 exec_common(const char *fname, const char **argp, const char **envp,
132     int brand_action)
133 {
134         vnode_t *vp = NULL, *dir = NULL, *tmpvp = NULL;
135         proc_t *p = ttoproc(curthread);
136         klwp_t *lwp = ttolwp(curthread);
137         struct user *up = PTOU(p);
138         long execsz;            /* temporary count of exec size */
139         int i;
140         int error;
141         char exec_file[MAXCOMLEN+1];
142         struct pathname pn;
143         struct pathname resolvepn;
144         struct uarg args;
145         struct execa ua;
146         k_sigset_t savedmask;
147         lwpdir_t *lwpdir = NULL;
148         tidhash_t *tidhash;
149         lwpdir_t *old_lwpdir = NULL;
150         uint_t old_lwpdir_sz;
151         tidhash_t *old_tidhash;
152         uint_t old_tidhash_sz;
153         ret_tidhash_t *ret_tidhash;
154         lwpent_t *lep;
155         boolean_t brandme = B_FALSE;

157         /*
158          * exec() is not supported for the /proc agent lwp.
159          */
160         if (curthread == p->p_agenttp)
161                 return (ENOTSUP);

163         if (brand_action != EBA_NONE) {
164                 /*
165                  * Brand actions are not supported for processes that are not
166                  * running in a branded zone.
167                  */
168                 if (!ZONE_IS_BRANDED(p->p_zone))
169                         return (ENOTSUP);

171                 if (brand_action == EBA_NATIVE) {
172                         /* Only branded processes can be unbranded */
173                         if (!PROC_IS_BRANDED(p))
174                                 return (ENOTSUP);
175                 } else {
176                         /* Only unbranded processes can be branded */
177                         if (PROC_IS_BRANDED(p))
178                                 return (ENOTSUP);
179                         brandme = B_TRUE;
180                 }
181         } else {
182                 /*
183                  * If this is a native zone, or if the process is already
184                  * branded, then we don't need to do anything.  If this is
185                  * a native process in a branded zone, we need to brand the
186                  * process as it exec()s the new binary.
187                  */
188                 if (ZONE_IS_BRANDED(p->p_zone) && !PROC_IS_BRANDED(p))
189                         brandme = B_TRUE;
190         }
```



```
192         /*
193          * Inform /proc that an exec() has started.
194          * Hold signals that are ignored by default so that we will
195          * not be interrupted by a signal that will be ignored after
196          * successful completion of gexec().
197          */
198         mutex_enter(&p->p_lock);
199         prexecstart();
200         schedctl_finish_sigblock(curthread);
201         savedmask = curthread->t_hold;
202         sigorset(&curthread->t_hold, &ignoredefault);
203         mutex_exit(&p->p_lock);

205         /*
206          * Look up path name and remember last component for later.
207          * To help coreadm expand its %d token, we attempt to save
208          * the directory containing the executable in p_execdir. The
209          * first call to lookuppn() may fail and return EINVAL because
210          * dirvpp is non-NULL. In that case, we make a second call to
211          * lookuppn() with dirvpp set to NULL; p_execdir will be NULL,
212          * but coreadm is allowed to expand %d to the empty string and
213          * there are other cases in which that failure may occur.
214          */
215         if ((error = pn_get((char *)fname, UIO_USERSPACE, &pn)) != 0)
216                 goto out;
217         pn_alloc(&resolvepn);
218         if ((error = lookuppn(&pn, &resolvepn, FOLLOW, &dir, &vp)) != 0) {
219                 pn_free(&resolvepn);
220                 pn_free(&pn);
221                 if (error != EINVAL)
222                         goto out;

224                 dir = NULL;
225                 if ((error = pn_get((char *)fname, UIO_USERSPACE, &pn)) != 0)
226                         goto out;
227                 pn_alloc(&resolvepn);
228                 if ((error = lookuppn(&pn, &resolvepn, FOLLOW, NULLVPP,
229                     &vp)) != 0) {
230                         pn_free(&resolvepn);
231                         pn_free(&pn);
232                         goto out;
233                 }
234         }
235         if (vp == NULL) {
236                 if (dir != NULL)
237                         VN_RELE(dir);
238                 error = ENOENT;
239                 pn_free(&resolvepn);
240                 pn_free(&pn);
241                 goto out;
242         }

244         if ((error = secpolicy_basic_exec(CRED(), vp)) != 0) {
245                 if (dir != NULL)
246                         VN_RELE(dir);
247                 pn_free(&resolvepn);
248                 pn_free(&pn);
249                 VN_RELE(vp);
250                 goto out;
251         }

253         /*
254          * We do not allow executing files in attribute directories.
255          * We test this by determining whether the resolved path
256          * contains a "/" when we're in an attribute directory;
```

```
257          * only if the pathname does not contain a "/" the resolved path
258          * points to a file in the current working (attribute) directory.
259          */
260         if ((p->p_user.u_cdir->v_flag & V_XATTRDIR) != 0 &&
261             strchr(resolvepn.pn_path, '/') == NULL) {
262                 if (dir != NULL)
263                         VN_RELE(dir);
264                 error = EACCES;
265                 pn_free(&resolvepn);
266                 pn_free(&pn);
267                 VN_RELE(vp);
268                 goto out;
269         }
271         bzero(exec_file, MAXCOMLEN+1);
272         (void) strncpy(exec_file, pn.pn_path, MAXCOMLEN);
273         bzero(&args, sizeof (args));
274         args.pathname = resolvepn.pn_path;
275         /* don't free resolvepn until we are done with args */
276         pn_free(&pn);
278         /*
279          * If we're running in a profile shell, then call pfexecd.
280          */
281         if ((CR_FLAGS(p->p_cred) & PRIV_PFEXEC) != 0) {
282                 error = pfexec_call(p->p_cred, &resolvepn, &args.pfcred,
283                     &args.scrubenv);
285                 /* Returning errno in case we're not allowed to execute. */
286                 if (error > 0) {
287                         if (dir != NULL)
288                                 VN_RELE(dir);
289                         pn_free(&resolvepn);
290                         VN_RELE(vp);
291                         goto out;
292                 }
294                 /* Don't change the credentials when using old ptrace. */
295                 if (args.pfcred != NULL &&
296                     (p->p_proc_flag & P_PR_PTRACE) != 0) {
297                         crfree(args.pfcred);
298                         args.pfcred = NULL;
299                         args.scrubenv = B_FALSE;
300                 }
301         }
303         /*
304          * Specific exec handlers, or policies determined via
305          * /etc/system may override the historical default.
306          */
307         args.stk_prot = PROT_ZFOD;
308         args.dat_prot = PROT_ZFOD;
310         CPU_STATS_ADD_K(sys, sysexec, 1);
311         DTRACE_PROC1(exec, char *, args.pathname);
313         ua.fname = fname;
314         ua.argp = argp;
315         ua.envp = envp;
317         /* If necessary, brand this process before we start the exec. */
318         if (brandme)
319                 brand_setbrand(p);
321         if ((error = gexec(&vp, &ua, &args, NULL, 0, &execsz,
322             exec_file, p->p_cred, brand_action)) != 0) {
```



```
323                 if (brandme)
324                         brand_clearbrand(p, B_FALSE);
325                 VN_RELE(vp);
326                 if (dir != NULL)
327                         VN_RELE(dir);
328                 pn_free(&resolvepn);
329                 goto fail;
330         }
332         /*
333          * Free floating point registers (sun4u only)
334          */
335         ASSERT(lwp != NULL);
336         lwp_freeregs(lwp, 1);
338         /*
339          * Free thread and process context ops.
340          */
341         if (curthread->t_ctx)
342                 freectx(curthread, 1);
343         if (p->p_pctx)
344                 freepctx(p, 1);
346         /*
347          * Remember file name for accounting; clear any cached DTrace predicate.
348          */
349         up->u_acflag &= ~AFORK;
350         bcopy(exec_file, up->u_comm, MAXCOMLEN+1);
351         curthread->t_predcache = NULL;
353         /*
354          * Clear contract template state
355          */
356         lwp_ctmpl_clear(lwp);
358         /*
359          * Save the directory in which we found the executable for expanding
360          * the %d token used in core file patterns.
361          */
362         mutex_enter(&p->p_lock);
363         tmpvp = p->p_execdir;
364         p->p_execdir = dir;
365         if (p->p_execdir != NULL)
366                 VN_HOLD(p->p_execdir);
367         mutex_exit(&p->p_lock);
369         if (tmpvp != NULL)
370                 VN_RELE(tmpvp);
372         /*
373          * Reset stack state to the user stack, clear set of signals
374          * caught on the signal stack, and reset list of signals that
375          * restart system calls; the new program's environment should
376          * not be affected by detritus from the old program.  Any
377          * pending held signals remain held, so don't clear t_hold.
378          */
379         mutex_enter(&p->p_lock);
380         lwp->lwp_oldcontext = 0;
381         lwp->lwp_ustack = 0;
382         lwp->lwp_old_stk_ctl = 0;
383         sigemptyset(&up->u_signodefer);
384         sigemptyset(&up->u_sigonstack);
385         sigemptyset(&up->u_sigresethand);
386         lwp->lwp_sigaltstack.ss_sp = 0;
387         lwp->lwp_sigaltstack.ss_size = 0;
388         lwp->lwp_sigaltstack.ss_flags = SS_DISABLE;
```

```
390         /*
391          * Make saved resource limit == current resource limit.
392          */
393         for (i = 0; i < RLIM_NLIMITS; i++) {
394                 /*CONSTCOND*/
395                 if (RLIM_SAVED(i)) {
396                         (void) rctl_rlimit_get(rctlproc_legacy[i], p,
397                             &up->u_saved_rlimit[i]);
398                 }
399         }

401         /*
402          * If the action was to catch the signal, then the action
403          * must be reset to SIG_DFL.
404          */
405         sigdefault(p);
406         p->p_flag &= ~(SNOWAIT|SJCTL);
407         p->p_flag |= (SEXECED|SMSACCT|SMSFORK);
408         up->u_signal[SIGCLD - 1] = SIG_DFL;

410         /*
411          * Delete the dot4 sigqueues/signotifies.
412          */
413         sigqfree(p);

415         mutex_exit(&p->p_lock);

417         mutex_enter(&p->p_pflock);
418         p->p_prof.pr_base = NULL;
419         p->p_prof.pr_size = 0;
420         p->p_prof.pr_off = 0;
421         p->p_prof.pr_scale = 0;
422         p->p_prof.pr_samples = 0;
423         mutex_exit(&p->p_pflock);

425         ASSERT(curthread->t_schedctl == NULL);

427 #if defined(__sparc)
428         if (p->p_utraps != NULL)
429                 utrap_free(p);
430 #endif  /* __sparc */

432         /*
433          * Close all close-on-exec files.
434          */
435         close_exec(P_FINFO(p));
436         TRACE_2(TR_FAC_PROC, TR_PROC_EXEC, "proc_exec:p %p up %p", p, up);

438         /* Unbrand ourself if necessary. */
439         if (PROC_IS_BRANDED(p) && (brand_action == EBA_NATIVE))
440                 brand_clearbrand(p, B_FALSE);

442         setregs(&args);

444         /* Mark this as an executable vnode */
445         mutex_enter(&vp->v_lock);
446         vp->v_flag |= VVMEXEC;
447         mutex_exit(&vp->v_lock);

449         VN_RELE(vp);
450         if (dir != NULL)
451                 VN_RELE(dir);
452         pn_free(&resolvepn);

454         /*
```



```
455          * Allocate a new lwp directory and lwpid hash table if necessary.
456          */
457         if (curthread->t_tid != 1 || p->p_lwpdir_sz != 2) {
458                 lwpdir = kmem_zalloc(2 * sizeof (lwpdir_t), KM_SLEEP);
459                 lwpdir->ld_next = lwpdir + 1;
460                 tidhash = kmem_zalloc(2 * sizeof (tidhash_t), KM_SLEEP);
461                 if (p->p_lwpdir != NULL)
462                         lep = p->p_lwpdir[curthread->t_dslot].ld_entry;
463                 else
464                         lep = kmem_zalloc(sizeof (*lep), KM_SLEEP);
465         }

467         if (PROC_IS_BRANDED(p))
468                 BROP(p)->b_exec();

470         mutex_enter(&p->p_lock);
471         prbarrier(p);

473         /*
474          * Reset lwp id to the default value of 1.
475          * This is a single-threaded process now
476          * and lwp #1 is lwp_wait()able by default.
477          * The t_unpark flag should not be inherited.
478          */
479         ASSERT(p->p_lwpcnt == 1 && p->p_zombcnt == 0);
480         curthread->t_tid = 1;
481         kpreempt_disable();
482         ASSERT(curthread->t_lpl != NULL);
483         p->p_t1_lgrpid = curthread->t_lpl->lpl_lgrpid;
484         kpreempt_enable();
485         if (p->p_tr_lgrpid != LGRP_NONE && p->p_tr_lgrpid != p->p_t1_lgrpid) {
486                 lgrp_update_trthr_migrations(1);
487         }
488         curthread->t_unpark = 0;
489         curthread->t_proc_flag |= TP_TWAIT;
490         curthread->t_proc_flag &= ~TP_DAEMON;   /* daemons shouldn't exec */
491         p->p_lwpdaemon = 0;                     /* but oh well ... */
492         p->p_lwpid = 1;

494         /*
495          * Install the newly-allocated lwp directory and lwpid hash table
496          * and insert the current thread into the new hash table.
497          */
498         if (lwpdir != NULL) {
499                 old_lwpdir = p->p_lwpdir;
500                 old_lwpdir_sz = p->p_lwpdir_sz;
501                 old_tidhash = p->p_tidhash;
502                 old_tidhash_sz = p->p_tidhash_sz;
503                 p->p_lwpdir = p->p_lwpfree = lwpdir;
504                 p->p_lwpdir_sz = 2;
505                 lep->le_thread = curthread;
506                 lep->le_lwpid = curthread->t_tid;
507                 lep->le_start = curthread->t_start;
508                 lwp_hash_in(p, lep, tidhash, 2, 0);
509                 p->p_tidhash = tidhash;
510                 p->p_tidhash_sz = 2;
511         }
512         ret_tidhash = p->p_ret_tidhash;
513         p->p_ret_tidhash = NULL;

515         /*
516          * Restore the saved signal mask and
517          * inform /proc that the exec() has finished.
518          */
519         curthread->t_hold = savedmask;
520         prexecend();
```

```
521         mutex_exit(&p->p_lock);
522         if (old_lwpdir) {
523                 kmem_free(old_lwpdir, old_lwpdir_sz * sizeof (lwpdir_t));
524                 kmem_free(old_tidhash, old_tidhash_sz * sizeof (tidhash_t));
525         }
526         while (ret_tidhash != NULL) {
527                 ret_tidhash_t *next = ret_tidhash->rth_next;
528                 kmem_free(ret_tidhash->rth_tidhash,
529                     ret_tidhash->rth_tidhash_sz * sizeof (tidhash_t));
530                 kmem_free(ret_tidhash, sizeof (*ret_tidhash));
531                 ret_tidhash = next;
532         }

534         ASSERT(error == 0);
535         DTRACE_PROC(exec__success);
536         return (0);

538 fail:
539         DTRACE_PROC1(exec__failure, int, error);
540 out:            /* error return */
541         mutex_enter(&p->p_lock);
542         curthread->t_hold = savedmask;
543         prexecend();
544         mutex_exit(&p->p_lock);
545         ASSERT(error != 0);
546         return (error);
547 }


550 /*
551  * Perform generic exec duties and switchout to object-file specific
552  * handler.
553  */
554 int
555 gexec(
556         struct vnode **vpp,
557         struct execa *uap,
558         struct uarg *args,
559         struct intpdata *idatap,
560         int level,
561         long *execsz,
562         caddr_t exec_file,
563         struct cred *cred,
564         int brand_action)
565 {
566         struct vnode *vp, *execvp = NULL;
567         proc_t *pp = ttoproc(curthread);
568         struct execsw *eswp;
569         int error = 0;
570         int suidflags = 0;
571         ssize_t resid;
572         uid_t uid, gid;
573         struct vattr vattr;
574         char magbuf[MAGIC_BYTES];
575         int setid;
576         cred_t *oldcred, *newcred = NULL;
577         int privflags = 0;
578         int setidfl;
579         priv_set_t fset;

581         /*
582          * If the SNOCD or SUGID flag is set, turn it off and remember the
583          * previous setting so we can restore it if we encounter an error.
584          */
585         if (level == 0 && (pp->p_flag & PSUIDFLAGS)) {
586                 mutex_enter(&pp->p_lock);
```



```
587                 suidflags = pp->p_flag & PSUIDFLAGS;
588                 pp->p_flag &= ~PSUIDFLAGS;
589                 mutex_exit(&pp->p_lock);
590         }

592         if ((error = execpermissions(*vpp, &vattr, args)) != 0)
593                 goto bad_noclose;

595         /* need to open vnode for stateful file systems */
596         if ((error = VOP_OPEN(vpp, FREAD, CRED(), NULL)) != 0)
597                 goto bad_noclose;
598         vp = *vpp;

600         /*
601          * Note: to support binary compatibility with SunOS a.out
602          * executables, we read in the first four bytes, as the
603          * magic number is in bytes 2-3.
604          */
605         if (error = vn_rdwr(UIO_READ, vp, magbuf, sizeof (magbuf),
606             (offset_t)0, UIO_SYSSPACE, 0, (rlim64_t)0, CRED(), &resid))
607                 goto bad;
608         if (resid != 0)
609                 goto bad;

611         if ((eswp = findexec_by_hdr(magbuf)) == NULL)
612                 goto bad;

614         if (level == 0 &&
615             (privflags = execsetid(vp, &vattr, &uid, &gid, &fset,
616             args->pfcred == NULL ? cred : args->pfcred, args->pathname)) != 0) {

618                 /* Pfcred is a credential with a ref count of 1 */

620                 if (args->pfcred != NULL) {
621                         privflags |= PRIV_INCREASE|PRIV_RESET;
622                         newcred = cred = args->pfcred;
623                 } else {
624                         newcred = cred = crdup(cred);
625                 }

627                 /* If we can, drop the PA bit */
628                 if ((privflags & PRIV_RESET) != 0)
629                         priv_adjust_PA(cred);

631                 if (privflags & PRIV_SETID) {
632                         cred->cr_uid = uid;
633                         cred->cr_gid = gid;
634                         cred->cr_suid = uid;
635                         cred->cr_sgid = gid;
636                 }

638                 if (privflags & MAC_FLAGS) {
639                         if (!(CR_FLAGS(cred) & NET_MAC_AWARE_INHERIT))
640                                 CR_FLAGS(cred) &= ~NET_MAC_AWARE;
641                         CR_FLAGS(cred) &= ~NET_MAC_AWARE_INHERIT;
642                 }

644                 /*
645                  * Implement the privilege updates:
646                  *
647                  * Restrict with L:
648                  *
649                  *      I' = I & L
650                  *
651                  *      E' = P' = (I' + F) & A
652                  *
```

```
653                  * But if running under ptrace, we cap I and F with P.
654                  */
655                 if ((privflags & (PRIV_RESET|PRIV_FORCED)) != 0) {
656                         if ((privflags & PRIV_INCREASE) != 0 &&
657                             (pp->p_proc_flag & P_PR_PTRACE) != 0) {
658                                 priv_intersect(&CR_OPPRIV(cred),
659                                     &CR_IPRIV(cred));
660                                 priv_intersect(&CR_OPPRIV(cred), &fset);
661                         }
662                         priv_intersect(&CR_LPRIV(cred), &CR_IPRIV(cred));
663                         CR_EPRIV(cred) = CR_PPRIV(cred) = CR_IPRIV(cred);
664                         if (privflags & PRIV_FORCED) {
665                                 priv_set_PA(cred);
666                                 priv_union(&fset, &CR_EPRIV(cred));
667                                 priv_union(&fset, &CR_PPRIV(cred));
668                         }
669                         priv_adjust_PA(cred);
670                 }
671         } else if (level == 0 && args->pfcred != NULL) {
672                 newcred = cred = args->pfcred;
673                 privflags |= PRIV_INCREASE;
674                 /* pfcred is not forced to adhere to these settings */
675                 priv_intersect(&CR_LPRIV(cred), &CR_IPRIV(cred));
676                 CR_EPRIV(cred) = CR_PPRIV(cred) = CR_IPRIV(cred);
677                 priv_adjust_PA(cred);
678         }

680         /* The new image gets the inheritable secflags as its secflags */
681         /* XXX: This probably means we have the wrong secflags when exec fails *
682         secflag_promote(pp);

684 #endif /* ! codereview */
685         /* SunOS 4.x buy-back */
686         if ((vp->v_vfsp->vfs_flag & VFS_NOSETUID) &&
687             (vattr.va_mode & (VSUID|VSGID))) {
688                 char path[MAXNAMELEN];
689                 refstr_t *mntpt = NULL;
690                 int ret = -1;

692                 bzero(path, sizeof (path));
693                 zone_hold(pp->p_zone);

695                 ret = vnodetopath(pp->p_zone->zone_rootvp, vp, path,
696                     sizeof (path), cred);

698                 /* fallback to mountpoint if a path can't be found */
699                 if ((ret != 0) || (ret == 0 && path[0] == '\0'))
700                         mntpt = vfs_getmntpoint(vp->v_vfsp);

702                 if (mntpt == NULL)
703                         zcmn_err(pp->p_zone->zone_id, CE_NOTE,
704                             "!uid %d: setuid execution not allowed, "
705                             "file=%s", cred->cr_uid, path);
706                 else
707                         zcmn_err(pp->p_zone->zone_id, CE_NOTE,
708                             "!uid %d: setuid execution not allowed, "
709                             "fs=%s, file=%s", cred->cr_uid,
710                             ZONE_PATH_TRANSLATE(refstr_value(mntpt),
711                             pp->p_zone), exec_file);

713                 if (!INGLOBALZONE(pp)) {
714                         /* zone_rootpath always has trailing / */
715                         if (mntpt == NULL)
716                                 cmn_err(CE_NOTE, "!zone: %s, uid: %d "
717                                     "setuid execution not allowed, file=%s%s",
718                                     pp->p_zone->zone_name, cred->cr_uid,
```



```
719                                     pp->p_zone->zone_rootpath, path + 1);
720                         else
721                                 cmn_err(CE_NOTE, "!zone: %s, uid: %d "
722                                     "setuid execution not allowed, fs=%s, "
723                                     "file=%s", pp->p_zone->zone_name,
724                                     cred->cr_uid, refstr_value(mntpt),
725                                     exec_file);
726                 }

728                 if (mntpt != NULL)
729                         refstr_rele(mntpt);

731                 zone_rele(pp->p_zone);
732         }

734         /*
735          * execsetid() told us whether or not we had to change the
736          * credentials of the process.  In privflags, it told us
737          * whether we gained any privileges or executed a set-uid executable.
738          */
739         setid = (privflags & (PRIV_SETUGID|PRIV_INCREASE|PRIV_FORCED));

741         /*
742          * Use /etc/system variable to determine if the stack
743          * should be marked as executable by default.
744          */
745         if (noexec_user_stack)
746                 args->stk_prot &= ~PROT_EXEC;

748         args->execswp = eswp; /* Save execsw pointer in uarg for exec_func */
749         args->ex_vp = vp;

751         /*
752          * Traditionally, the setid flags told the sub processes whether
753          * the file just executed was set-uid or set-gid; this caused
754          * some confusion as the 'setid' flag did not match the SUGID
755          * process flag which is only set when the uids/gids do not match.
756          * A script set-gid/set-uid to the real uid/gid would start with
757          * /dev/fd/X but an executable would happily trust LD_LIBRARY_PATH.
758          * Now we flag those cases where the calling process cannot
759          * be trusted to influence the newly exec'ed process, either
760          * because it runs with more privileges or when the uids/gids
761          * do in fact not match.
762          * This also makes the runtime linker agree with the on exec
763          * values of SNOCD and SUGID.
764          */
765         setidfl = 0;
766         if (cred->cr_uid != cred->cr_ruid || (cred->cr_rgid != cred->cr_gid &&
767             !supgroupmember(cred->cr_gid, cred))) {
768                 setidfl |= EXECSETID_UGIDS;
769         }
770         if (setid & PRIV_SETUGID)
771                 setidfl |= EXECSETID_SETID;
772         if (setid & PRIV_FORCED)
773                 setidfl |= EXECSETID_PRIVS;

775         execvp = pp->p_exec;
776         if (execvp)
777                 VN_HOLD(execvp);

779         error = (*eswp->exec_func)(vp, uap, args, idatap, level, execsz,
780             setidfl, exec_file, cred, brand_action);
781         rw_exit(eswp->exec_lock);
782         if (error != 0) {
783                 if (execvp)
784                         VN_RELE(execvp);
```

```
785                 /*
786                  * If this process's p_exec has been set to the vp of
787                  * the executable by exec_func, we will return without
788                  * calling VOP_CLOSE because proc_exit will close it
789                  * on exit.
790                  */
791                 if (pp->p_exec == vp)
792                         goto bad_noclose;
793                 else
794                         goto bad;
795         }
796
797         if (level == 0) {
798                 uid_t oruid;
799
800                 if (execvp != NULL) {
801                         /*
802                          * Close the previous executable only if we are
803                          * at level 0.
804                          */
805                         (void) VOP_CLOSE(execvp, FREAD, 1, (offset_t)0,
806                             cred, NULL);
807                 }
808
809                 mutex_enter(&pp->p_crlock);
810
811                 oruid = pp->p_cred->cr_ruid;
812
813                 if (newcred != NULL) {
814                         /*
815                          * Free the old credentials, and set the new ones.
816                          * Do this for both the process and the (single) thread.
817                          */
818                         crfree(pp->p_cred);
819                         pp->p_cred = cred;      /* cred already held for proc */
820                         crhold(cred);           /* hold new cred for thread */
821                         /*
822                          * DTrace accesses t_cred in probe context.  t_cred
823                          * must always be either NULL, or point to a valid,
824                          * allocated cred structure.
825                          */
826                         oldcred = curthread->t_cred;
827                         curthread->t_cred = cred;
828                         crfree(oldcred);
829
830                         if (priv_basic_test >= 0 &&
831                             !PRIV_ISASSERT(&CR_IPRIV(newcred),
832                             priv_basic_test)) {
833                                 pid_t pid = pp->p_pid;
834                                 char *fn = PTOU(pp)->u_comm;
835
836                                 cmn_err(CE_WARN, "%s[%d]: exec: basic_test "
837                                     "privilege removed from E/I", fn, pid);
838                         }
839                 }
840                 /*
841                  * On emerging from a successful exec(), the saved
842                  * uid and gid equal the effective uid and gid.
843                  */
844                 cred->cr_suid = cred->cr_uid;
845                 cred->cr_sgid = cred->cr_gid;
846
847                 /*
848                  * If the real and effective ids do not match, this
849                  * is a setuid process that should not dump core.
850                  * The group comparison is tricky; we prevent the code
```



```
851                  * from flagging SNOCD when executing with an effective gid
852                  * which is a supplementary group.
853                  */
854                 if (cred->cr_ruid != cred->cr_uid ||
855                     (cred->cr_rgid != cred->cr_gid &&
856                     !supgroupmember(cred->cr_gid, cred)) ||
857                     (privflags & PRIV_INCREASE) != 0)
858                         suidflags = PSUIDFLAGS;
859                 else
860                         suidflags = 0;
861
862                 mutex_exit(&pp->p_crlock);
863                 if (newcred != NULL && oruid != newcred->cr_ruid) {
864                         /* Note that the process remains in the same zone. */
865                         mutex_enter(&pidlock);
866                         upcount_dec(oruid, crgetzoneid(newcred));
867                         upcount_inc(newcred->cr_ruid, crgetzoneid(newcred));
868                         mutex_exit(&pidlock);
869                 }
870                 if (suidflags) {
871                         mutex_enter(&pp->p_lock);
872                         pp->p_flag |= suidflags;
873                         mutex_exit(&pp->p_lock);
874                 }
875                 if (setid && (pp->p_proc_flag & P_PR_PTRACE) == 0) {
876                         /*
877                          * If process is traced via /proc, arrange to
878                          * invalidate the associated /proc vnode.
879                          */
880                         if (pp->p_plist || (pp->p_proc_flag & P_PR_TRACE))
881                                 args->traceinval = 1;
882                 }
883                 if (pp->p_proc_flag & P_PR_PTRACE)
884                         psignal(pp, SIGTRAP);
885                 if (args->traceinval)
886                         prinvalidate(&pp->p_user);
887         }
888         if (execvp)
889                 VN_RELE(execvp);
890         return (0);
891
892 bad:
893         (void) VOP_CLOSE(vp, FREAD, 1, (offset_t)0, cred, NULL);
894
895 bad_noclose:
896         if (newcred != NULL)
897                 crfree(newcred);
898         if (error == 0)
899                 error = ENOEXEC;
900
901         if (suidflags) {
902                 mutex_enter(&pp->p_lock);
903                 pp->p_flag |= suidflags;
904                 mutex_exit(&pp->p_lock);
905         }
906         return (error);
907 }
908
909 extern char *execswnames[];
910
911 struct execsw *
912 allocate_execsw(char *name, char *magic, size_t magic_size)
913 {
914         int i, j;
915         char *ename;
916         char *magicp;
```

```
        mutex_enter(&execsw_lock);
        for (i = 0; i < nexectype; i++) {
                if (execswnames[i] == NULL) {
                        ename = kmem_alloc(strlen(name) + 1, KM_SLEEP);
                        (void) strcpy(ename, name);
                        execswnames[i] = ename;
                        /*
                         * Set the magic number last so that we
                         * don't need to hold the execsw_lock in
                         * findexectype().
                         */
                        magicp = kmem_alloc(magic_size, KM_SLEEP);
                        for (j = 0; j < magic_size; j++)
                                magicp[j] = magic[j];
                        execsw[i].exec_magic = magicp;
                        mutex_exit(&execsw_lock);
                        return (&execsw[i]);
                }
        }
        mutex_exit(&execsw_lock);
        return (NULL);
}

/*
 * Find the exec switch table entry with the corresponding magic string.
 */
struct execsw *
findexecsw(char *magic)
{
        struct execsw *eswp;

        for (eswp = execsw; eswp < &execsw[nexectype]; eswp++) {
                ASSERT(eswp->exec_maglen <= MAGIC_BYTES);
                if (magic && eswp->exec_maglen != 0 &&
                    bcmp(magic, eswp->exec_magic, eswp->exec_maglen) == 0)
                        return (eswp);
        }
        return (NULL);
}

/*
 * Find the execsw[] index for the given exec header string by looking for the
 * magic string at a specified offset and length for each kind of executable
 * file format until one matches.  If no execsw[] entry is found, try to
 * autoload a module for this magic string.
 */
struct execsw *
findexec_by_hdr(char *header)
{
        struct execsw *eswp;

        for (eswp = execsw; eswp < &execsw[nexectype]; eswp++) {
                ASSERT(eswp->exec_maglen <= MAGIC_BYTES);
                if (header && eswp->exec_maglen != 0 &&
                    bcmp(&header[eswp->exec_magoff], eswp->exec_magic,
                    eswp->exec_maglen) == 0) {
                        if (hold_execsw(eswp) != 0)
                                return (NULL);
                        return (eswp);
                }
        }
        return (NULL);  /* couldn't find the type */
}

/*
```



```
 * Find the execsw[] index for the given magic string.  If no execsw[] entry
 * is found, try to autoload a module for this magic string.
 */
struct execsw *
findexec_by_magic(char *magic)
{
        struct execsw *eswp;

        for (eswp = execsw; eswp < &execsw[nexectype]; eswp++) {
                ASSERT(eswp->exec_maglen <= MAGIC_BYTES);
                if (magic && eswp->exec_maglen != 0 &&
                    bcmp(magic, eswp->exec_magic, eswp->exec_maglen) == 0) {
                        if (hold_execsw(eswp) != 0)
                                return (NULL);
                        return (eswp);
                }
        }
        return (NULL);  /* couldn't find the type */
}

static int
hold_execsw(struct execsw *eswp)
{
        char *name;

        rw_enter(eswp->exec_lock, RW_READER);
        while (!LOADED_EXEC(eswp)) {
                rw_exit(eswp->exec_lock);
                name = execswnames[eswp-execsw];
                ASSERT(name);
                if (modload("exec", name) == -1)
                        return (-1);
                rw_enter(eswp->exec_lock, RW_READER);
        }
        return (0);
}

static int
execsetid(struct vnode *vp, struct vattr *vattrp, uid_t *uidp, uid_t *gidp,
    priv_set_t *fset, cred_t *cr, const char *pathname)
{
        proc_t *pp = ttoproc(curthread);
        uid_t uid, gid;
        int privflags = 0;

        /*
         * Remember credentials.
         */
        uid = cr->cr_uid;
        gid = cr->cr_gid;

        /* Will try to reset the PRIV_AWARE bit later. */
        if ((CR_FLAGS(cr) & (PRIV_AWARE|PRIV_AWARE_INHERIT)) == PRIV_AWARE)
                privflags |= PRIV_RESET;

        if ((vp->v_vfsp->vfs_flag & VFS_NOSETUID) == 0) {
                /*
                 * If it's a set-uid root program we perform the
                 * forced privilege look-aside. This has three possible
                 * outcomes:
                 *      no look aside information -> treat as before
                 *      look aside in Limit set -> apply forced privs
                 *      look aside not in Limit set -> ignore set-uid root
                 *
                 * Ordinary set-uid root execution only allowed if the limit
                 * set holds all unsafe privileges.
```

```
1049                 */
1050                if (vattrp->va_mode & VSUID) {
1051                        if (vattrp->va_uid == 0) {
1052                                int res = get_forced_privs(cr, pathname, fset);

1054                                switch (res) {
1055                                case -1:
1056                                        if (priv_issubset(&priv_unsafe,
1057                                            &CR_LPRIV(cr))) {
1058                                                uid = vattrp->va_uid;
1059                                                privflags |= PRIV_SETUGID;
1060                                        }
1061                                        break;
1062                                case 0:
1063                                        privflags |= PRIV_FORCED|PRIV_INCREASE;
1064                                        break;
1065                                default:
1066                                        break;
1067                                }
1068                        } else {
1069                                uid = vattrp->va_uid;
1070                                privflags |= PRIV_SETUGID;
1071                        }
1072                }
1073                if (vattrp->va_mode & VSGID) {
1074                        gid = vattrp->va_gid;
1075                        privflags |= PRIV_SETUGID;
1076                }
1077        }

1079        /*
1080         * Do we need to change our credential anyway?
1081         * This is the case when E != I or P != I, as
1082         * we need to do the assignments (with F empty and A full)
1083         * Or when I is not a subset of L; in that case we need to
1084         * enforce L.
1085         *
1086         *              I' = L & I
1087         *
1088         *              E' = P' = (I' + F) & A
1089         * or
1090         *              E' = P' = I'
1091         */
1092        if (!priv_isequalset(&CR_EPRIV(cr), &CR_IPRIV(cr)) ||
1093            !priv_issubset(&CR_IPRIV(cr), &CR_LPRIV(cr)) ||
1094            !priv_isequalset(&CR_PPRIV(cr), &CR_IPRIV(cr)))
1095                privflags |= PRIV_RESET;

1097        /* Child has more privileges than parent */
1098        if (!priv_issubset(&CR_IPRIV(cr), &CR_PPRIV(cr)))
1099                privflags |= PRIV_INCREASE;

1101        /* If MAC-aware flag(s) are on, need to update cred to remove. */
1102        if ((CR_FLAGS(cr) & NET_MAC_AWARE) ||
1103            (CR_FLAGS(cr) & NET_MAC_AWARE_INHERIT))
1104                privflags |= MAC_FLAGS;
1105        /*
1106         * Set setuid/setgid protections if no ptrace() compatibility.
1107         * For privileged processes, honor setuid/setgid even in
1108         * the presence of ptrace() compatibility.
1109         */
1110        if (((pp->p_proc_flag & P_PR_PTRACE) == 0 ||
1111            PRIV_POLICY_ONLY(cr, PRIV_PROC_OWNER, (uid == 0))) &&
1112            (cr->cr_uid != uid ||
1113            cr->cr_gid != gid ||
1114            cr->cr_suid != uid ||
```



```
1115                cr->cr_sgid != gid)) {
1116                    *uidp = uid;
1117                    *gidp = gid;
1118                    privflags |= PRIV_SETID;
1119            }
1120            return (privflags);
1121    }

1123    int
1124    execpermissions(struct vnode *vp, struct vattr *vattrp, struct uarg *args)
1125    {
1126            int error;
1127            proc_t *p = ttoproc(curthread);

1129            vattrp->va_mask = AT_MODE | AT_UID | AT_GID | AT_SIZE;
1130            if (error = VOP_GETATTR(vp, vattrp, ATTR_EXEC, p->p_cred, NULL))
1131                    return (error);
1132            /*
1133             * Check the access mode.
1134             * If VPROC, ask /proc if the file is an object file.
1135             */
1136            if ((error = VOP_ACCESS(vp, VEXEC, 0, p->p_cred, NULL)) != 0 ||
1137                !(vp->v_type == VREG || (vp->v_type == VPROC && pr_isobject(vp))) ||
1138                (vp->v_vfsp->vfs_flag & VFS_NOEXEC) != 0 ||
1139                (vattrp->va_mode & (VEXEC|(VEXEC>>3)|(VEXEC>>6))) == 0) {
1140                    if (error == 0)
1141                            error = EACCES;
1142                    return (error);
1143            }

1145            if ((p->p_plist || (p->p_proc_flag & (P_PR_PTRACE|P_PR_TRACE))) &&
1146                (error = VOP_ACCESS(vp, VREAD, 0, p->p_cred, NULL))) {
1147                    /*
1148                     * If process is under ptrace(2) compatibility,
1149                     * fail the exec(2).
1150                     */
1151                    if (p->p_proc_flag & P_PR_PTRACE)
1152                            goto bad;
1153                    /*
1154                     * Process is traced via /proc.
1155                     * Arrange to invalidate the /proc vnode.
1156                     */
1157                    args->traceinval = 1;
1158            }
1159            return (0);
1160    bad:
1161            if (error == 0)
1162                    error = ENOEXEC;
1163            return (error);
1164    }

1166    /*
1167     * Map a section of an executable file into the user's
1168     * address space.
1169     */
1170    int
1171    execmap(struct vnode *vp, caddr_t addr, size_t len, size_t zfodlen,
1172        off_t offset, int prot, int page, uint_t szc)
1173    {
1174            int error = 0;
1175            off_t oldoffset;
1176            caddr_t zfodbase, oldaddr;
1177            size_t end, oldlen;
1178            size_t zfoddiff;
1179            label_t ljb;
1180            proc_t *p = ttoproc(curthread);
```

```
1182         oldaddr = addr;
1183         addr = (caddr_t)((uintptr_t)addr & (uintptr_t)PAGEMASK);
1184         if (len) {
1185                 oldlen = len;
1186                 len += ((size_t)oldaddr - (size_t)addr);
1187                 oldoffset = offset;
1188                 offset = (off_t)((uintptr_t)offset & PAGEMASK);
1189                 if (page) {
1190                         spgcnt_t  prefltmem, availm, npages;
1191                         int preread;
1192                         uint_t mflag = MAP_PRIVATE | MAP_FIXED;

1194                         if ((prot & (PROT_WRITE | PROT_EXEC)) == PROT_EXEC) {
1195                                 mflag |= MAP_TEXT;
1196                         } else {
1197                                 mflag |= MAP_INITDATA;
1198                         }

1200                         if (valid_usr_range(addr, len, prot, p->p_as,
1201                             p->p_as->a_userlimit) != RANGE_OKAY) {
1202                                 error = ENOMEM;
1203                                 goto bad;
1204                         }
1205                         if (error = VOP_MAP(vp, (offset_t)offset,
1206                             p->p_as, &addr, len, prot, PROT_ALL,
1207                             mflag, CRED(), NULL))
1208                                 goto bad;

1210                         /*
1211                          * If the segment can fit, then we prefault
1212                          * the entire segment in.  This is based on the
1213                          * model that says the best working set of a
1214                          * small program is all of its pages.
1215                          */
1216                         npages = (spgcnt_t)btopr(len);
1217                         prefltmem = freemem - desfree;
1218                         preread =
1219                             (npages < prefltmem && len < PGTHRESH) ? 1 : 0;

1221                         /*
1222                          * If we aren't prefaulting the segment,
1223                          * increment "deficit", if necessary to ensure
1224                          * that pages will become available when this
1225                          * process starts executing.
1226                          */
1227                         availm = freemem - lotsfree;
1228                         if (preread == 0 && npages > availm &&
1229                             deficit < lotsfree) {
1230                                 deficit += MIN((pgcnt_t)(npages - availm),
1231                                     lotsfree - deficit);
1232                         }

1234                         if (preread) {
1235                                 TRACE_2(TR_FAC_PROC, TR_EXECMAP_PREREAD,
1236                                     "execmap preread:freemem %d size %lu",
1237                                     freemem, len);
1238                                 (void) as_fault(p->p_as->a_hat, p->p_as,
1239                                     (caddr_t)addr, len, F_INVAL, S_READ);
1240                         }
1241                 } else {
1242                         if (valid_usr_range(addr, len, prot, p->p_as,
1243                             p->p_as->a_userlimit) != RANGE_OKAY) {
1244                                 error = ENOMEM;
1245                                 goto bad;
1246                         }
```



```
1248                         if (error = as_map(p->p_as, addr, len,
1249                             segvn_create, zfod_argsp))
1250                                 goto bad;
1251                         /*
1252                          * Read in the segment in one big chunk.
1253                          */
1254                         if (error = vn_rdwr(UIO_READ, vp, (caddr_t)oldaddr,
1255                             oldlen, (offset_t)oldoffset, UIO_USERSPACE, 0,
1256                             (rlim64_t)0, CRED(), (ssize_t *)0))
1257                                 goto bad;
1258                         /*
1259                          * Now set protections.
1260                          */
1261                         if (prot != PROT_ZFOD) {
1262                                 (void) as_setprot(p->p_as, (caddr_t)addr,
1263                                     len, prot);
1264                         }
1265                 }
1266         }

1268         if (zfodlen) {
1269                 struct as *as = curproc->p_as;
1270                 struct seg *seg;
1271                 uint_t zprot = 0;

1273                 end = (size_t)addr + len;
1274                 zfodbase = (caddr_t)roundup(end, PAGESIZE);
1275                 zfoddiff = (uintptr_t)zfodbase - end;
1276                 if (zfoddiff) {
1277                         /*
1278                          * Before we go to zero the remaining space on the last
1279                          * page, make sure we have write permission.
1280                          *
1281                          * Normal illumos binaries don't even hit the case
1282                          * where we have to change permission on the last page
1283                          * since their protection is typically either
1284                          *    PROT_USER | PROT_WRITE | PROT_READ
1285                          * or
1286                          *    PROT_ZFOD (same as PROT_ALL).
1287                          *
1288                          * We need to be careful how we zero-fill the last page
1289                          * if the segment protection does not include
1290                          * PROT_WRITE. Using as_setprot() can cause the VM
1291                          * segment code to call segvn_vpage(), which must
1292                          * allocate a page struct for each page in the segment.
1293                          * If we have a very large segment, this may fail, so
1294                          * we have to check for that, even though we ignore
1295                          * other return values from as_setprot.
1296                          */

1298                         AS_LOCK_ENTER(as, &as->a_lock, RW_READER);
1299                         seg = as_segat(curproc->p_as, (caddr_t)end);
1300                         if (seg != NULL)
1301                                 SEGOP_GETPROT(seg, (caddr_t)end, zfoddiff - 1,
1302                                     &zprot);
1303                         AS_LOCK_EXIT(as, &as->a_lock);

1305                         if (seg != NULL && (zprot & PROT_WRITE) == 0) {
1306                                 if (as_setprot(as, (caddr_t)end, zfoddiff - 1,
1307                                     zprot | PROT_WRITE) == ENOMEM) {
1308                                         error = ENOMEM;
1309                                         goto bad;
1310                                 }
1311                         }
```

```
1313                        if (on_fault(&ljb)) {
1314                                no_fault();
1315                                if (seg != NULL && (zprot & PROT_WRITE) == 0)
1316                                        (void) as_setprot(as, (caddr_t)end,
1317                                            zfoddiff - 1, zprot);
1318                                error = EFAULT;
1319                                goto bad;
1320                        }
1321                        uzero((void *)end, zfoddiff);
1322                        no_fault();
1323                        if (seg != NULL && (zprot & PROT_WRITE) == 0)
1324                                (void) as_setprot(as, (caddr_t)end,
1325                                    zfoddiff - 1, zprot);
1326                }
1327                if (zfodlen > zfoddiff) {
1328                        struct segvn_crargs crargs =
1329                            SEGVN_ZFOD_ARGS(PROT_ZFOD, PROT_ALL);

1331                        zfodlen -= zfoddiff;
1332                        if (valid_usr_range(zfodbase, zfodlen, prot, p->p_as,
1333                            p->p_as->a_userlimit) != RANGE_OKAY) {
1334                                error = ENOMEM;
1335                                goto bad;
1336                        }
1337                        if (szc > 0) {
1338                                /*
1339                                 * ASSERT alignment because the mapelfexec()
1340                                 * caller for the szc > 0 case extended zfod
1341                                 * so it's end is pgsz aligned.
1342                                 */
1343                                size_t pgsz = page_get_pagesize(szc);
1344                                ASSERT(IS_P2ALIGNED(zfodbase + zfodlen, pgsz));

1346                                if (IS_P2ALIGNED(zfodbase, pgsz)) {
1347                                        crargs.szc = szc;
1348                                } else {
1349                                        crargs.szc = AS_MAP_HEAP;
1350                                }
1351                        } else {
1352                                crargs.szc = AS_MAP_NO_LPOOB;
1353                        }
1354                        if (error = as_map(p->p_as, (caddr_t)zfodbase,
1355                            zfodlen, segvn_create, &crargs))
1356                                goto bad;
1357                        if (prot != PROT_ZFOD) {
1358                                (void) as_setprot(p->p_as, (caddr_t)zfodbase,
1359                                    zfodlen, prot);
1360                        }
1361                }
1362        }
1363        return (0);
1364 bad:
1365        return (error);
1366 }

1368 void
1369 setexecenv(struct execenv *ep)
1370 {
1371        proc_t *p = ttoproc(curthread);
1372        klwp_t *lwp = ttolwp(curthread);
1373        struct vnode *vp;

1375        p->p_bssbase = ep->ex_bssbase;
1376        p->p_brkbase = ep->ex_brkbase;
1377        p->p_brksize = ep->ex_brksize;
1378        if (p->p_exec)
```



```
1379                VN_RELE(p->p_exec);     /* out with the old */
1380        vp = p->p_exec = ep->ex_vp;
1381        if (vp != NULL)
1382                VN_HOLD(vp);            /* in with the new */

1384        lwp->lwp_sigaltstack.ss_sp = 0;
1385        lwp->lwp_sigaltstack.ss_size = 0;
1386        lwp->lwp_sigaltstack.ss_flags = SS_DISABLE;
1387 }

1389 int
1390 execopen(struct vnode **vpp, int *fdp)
1391 {
1392        struct vnode *vp = *vpp;
1393        file_t *fp;
1394        int error = 0;
1395        int filemode = FREAD;

1397        VN_HOLD(vp);            /* open reference */
1398        if (error = falloc(NULL, filemode, &fp, fdp)) {
1399                VN_RELE(vp);
1400                *fdp = -1;      /* just in case falloc changed value */
1401                return (error);
1402        }
1403        if (error = VOP_OPEN(&vp, filemode, CRED(), NULL)) {
1404                VN_RELE(vp);
1405                setf(*fdp, NULL);
1406                unfalloc(fp);
1407                *fdp = -1;
1408                return (error);
1409        }
1410        *vpp = vp;              /* vnode should not have changed */
1411        fp->f_vnode = vp;
1412        mutex_exit(&fp->f_tlock);
1413        setf(*fdp, fp);
1414        return (0);
1415 }

1417 int
1418 execclose(int fd)
1419 {
1420        return (closeandsetf(fd, NULL));
1421 }


1424 /*
1425  * noexec stub function.
1426  */
1427 /*ARGSUSED*/
1428 int
1429 noexec(
1430     struct vnode *vp,
1431     struct execa *uap,
1432     struct uarg *args,
1433     struct intpdata *idatap,
1434     int level,
1435     long *execsz,
1436     int setid,
1437     caddr_t exec_file,
1438     struct cred *cred)
1439 {
1440        cmn_err(CE_WARN, "missing exec capability for %s", uap->fname);
1441        return (ENOEXEC);
1442 }

1444 /*
```

```
1445  * Support routines for building a user stack.
1446  *
1447  * execve(path, argv, envp) must construct a new stack with the specified
1448  * arguments and environment variables (see exec_args() for a description
1449  * of the user stack layout).  To do this, we copy the arguments and
1450  * environment variables from the old user address space into the kernel,
1451  * free the old as, create the new as, and copy our buffered information
1452  * to the new stack.  Our kernel buffer has the following structure:
1453  *
1454  *      +-----------------------+ <--- stk_base + stk_size
1455  *      | string offsets        |
1456  *      +-----------------------+ <--- stk_offp
1457  *      |                       |
1458  *      | STK_AVAIL() space     |
1459  *      |                       |
1460  *      +-----------------------+ <--- stk_strp
1461  *      | strings               |
1462  *      +-----------------------+ <--- stk_base
1463  *
1464  * When we add a string, we store the string's contents (including the null
1465  * terminator) at stk_strp, and we store the offset of the string relative to
1466  * stk_base at --stk_offp.  At strings are added, stk_strp increases and
1467  * stk_offp decreases.  The amount of space remaining, STK_AVAIL(), is just
1468  * the difference between these pointers.  If we run out of space, we return
1469  * an error and exec_args() starts all over again with a buffer twice as large.
1470  * When we're all done, the kernel buffer looks like this:
1471  *
1472  *      +-----------------------+ <--- stk_base + stk_size
1473  *      | argv[0] offset        |
1474  *      +-----------------------+
1475  *      | ...                   |
1476  *      +-----------------------+
1477  *      | argv[argc-1] offset   |
1478  *      +-----------------------+
1479  *      | envp[0] offset        |
1480  *      +-----------------------+
1481  *      | ...                   |
1482  *      +-----------------------+
1483  *      | envp[envc-1] offset   |
1484  *      +-----------------------+
1485  *      | AT_SUN_PLATFORM offset|
1486  *      +-----------------------+
1487  *      | AT_SUN_EXECNAME offset|
1488  *      +-----------------------+ <--- stk_offp
1489  *      |                       |
1490  *      | STK_AVAIL() space     |
1491  *      |                       |
1492  *      +-----------------------+ <--- stk_strp
1493  *      | AT_SUN_EXECNAME offset|
1494  *      +-----------------------+
1495  *      | AT_SUN_PLATFORM offset|
1496  *      +-----------------------+
1497  *      | envp[envc-1] string   |
1498  *      +-----------------------+
1499  *      | ...                   |
1500  *      +-----------------------+
1501  *      | envp[0] string        |
1502  *      +-----------------------+
1503  *      | argv[argc-1] string   |
1504  *      +-----------------------+
1505  *      | ...                   |
1506  *      +-----------------------+
1507  *      | argv[0] string        |
1508  *      +-----------------------+ <--- stk_base
1509  */
```



```
1511 #define STK_AVAIL(args)         ((char *)(args)->stk_offp - (args)->stk_strp)

1513 /*
1514  * Add a string to the stack.
1515  */
1516 static int
1517 stk_add(uarg_t *args, const char *sp, enum uio_seg segflg)
1518 {
1519         int error;
1520         size_t len;

1522         if (STK_AVAIL(args) < sizeof (int))
1523                 return (E2BIG);
1524         *--args->stk_offp = args->stk_strp - args->stk_base;

1526         if (segflg == UIO_USERSPACE) {
1527                 error = copyinstr(sp, args->stk_strp, STK_AVAIL(args), &len);
1528                 if (error != 0)
1529                         return (error);
1530         } else {
1531                 len = strlen(sp) + 1;
1532                 if (len > STK_AVAIL(args))
1533                         return (E2BIG);
1534                 bcopy(sp, args->stk_strp, len);
1535         }

1537         args->stk_strp += len;

1539         return (0);
1540 }

1542 static int
1543 stk_getptr(uarg_t *args, char *src, char **dst)
1544 {
1545         int error;

1547         if (args->from_model == DATAMODEL_NATIVE) {
1548                 ulong_t ptr;
1549                 error = fulword(src, &ptr);
1550                 *dst = (caddr_t)ptr;
1551         } else {
1552                 uint32_t ptr;
1553                 error = fuword32(src, &ptr);
1554                 *dst = (caddr_t)(uintptr_t)ptr;
1555         }
1556         return (error);
1557 }

1559 static int
1560 stk_putptr(uarg_t *args, char *addr, char *value)
1561 {
1562         if (args->to_model == DATAMODEL_NATIVE)
1563                 return (sulword(addr, (ulong_t)value));
1564         else
1565                 return (suword32(addr, (uint32_t)(uintptr_t)value));
1566 }

1568 static int
1569 stk_copyin(execa_t *uap, uarg_t *args, intpdata_t *intp, void **auxvpp)
1570 {
1571         char *sp;
1572         int argc, error;
1573         int argv_empty = 0;
1574         size_t ptrsize = args->from_ptrsize;
1575         size_t size, pad;
1576         char *argv = (char *)uap->argp;
```

```
1577         char *envp = (char *)uap->envp;
1578
1579         /*
1580          * Copy interpreter's name and argument to argv[0] and argv[1].
1581          */
1582         if (intp != NULL && intp->intp_name != NULL) {
1583                 if ((error = stk_add(args, intp->intp_name, UIO_SYSSPACE)) != 0)
1584                         return (error);
1585                 if (intp->intp_arg != NULL &&
1586                     (error = stk_add(args, intp->intp_arg, UIO_SYSSPACE)) != 0)
1587                         return (error);
1588                 if (args->fname != NULL)
1589                         error = stk_add(args, args->fname, UIO_SYSSPACE);
1590                 else
1591                         error = stk_add(args, uap->fname, UIO_USERSPACE);
1592                 if (error)
1593                         return (error);
1594
1595                 /*
1596                  * Check for an empty argv[].
1597                  */
1598                 if (stk_getptr(args, argv, &sp))
1599                         return (EFAULT);
1600                 if (sp == NULL)
1601                         argv_empty = 1;
1602
1603                 argv += ptrsize;                /* ignore original argv[0] */
1604         }
1605
1606         if (argv_empty == 0) {
1607                 /*
1608                  * Add argv[] strings to the stack.
1609                  */
1610                 for (;;) {
1611                         if (stk_getptr(args, argv, &sp))
1612                                 return (EFAULT);
1613                         if (sp == NULL)
1614                                 break;
1615                         if ((error = stk_add(args, sp, UIO_USERSPACE)) != 0)
1616                                 return (error);
1617                         argv += ptrsize;
1618                 }
1619         }
1620         argc = (int *)(args->stk_base + args->stk_size) - args->stk_offp;
1621         args->arglen = args->stk_strp - args->stk_base;
1622
1623         /*
1624          * Add environ[] strings to the stack.
1625          */
1626         if (envp != NULL) {
1627                 for (;;) {
1628                         char *tmp = args->stk_strp;
1629                         if (stk_getptr(args, envp, &sp))
1630                                 return (EFAULT);
1631                         if (sp == NULL)
1632                                 break;
1633                         if ((error = stk_add(args, sp, UIO_USERSPACE)) != 0)
1634                                 return (error);
1635                         if (args->scrubenv && strncmp(tmp, "LD_", 3) == 0) {
1636                                 /* Undo the copied string */
1637                                 args->stk_strp = tmp;
1638                                 *(args->stk_offp++) = NULL;
1639                         }
1640                         envp += ptrsize;
1641                 }
1642         }
```



```
1643         args->na = (int *)(args->stk_base + args->stk_size) - args->stk_offp;
1644         args->ne = args->na - argc;
1645
1646         /*
1647          * Add AT_SUN_PLATFORM, AT_SUN_EXECNAME, AT_SUN_BRANDNAME, and
1648          * AT_SUN_EMULATOR strings to the stack.
1649          */
1650         if (auxvpp != NULL && *auxvpp != NULL) {
1651                 if ((error = stk_add(args, platform, UIO_SYSSPACE)) != 0)
1652                         return (error);
1653                 if ((error = stk_add(args, args->pathname, UIO_SYSSPACE)) != 0)
1654                         return (error);
1655                 if (args->brandname != NULL &&
1656                     (error = stk_add(args, args->brandname, UIO_SYSSPACE)) != 0)
1657                         return (error);
1658                 if (args->emulator != NULL &&
1659                     (error = stk_add(args, args->emulator, UIO_SYSSPACE)) != 0)
1660                         return (error);
1661         }
1662
1663         /*
1664          * Compute the size of the stack.  This includes all the pointers,
1665          * the space reserved for the aux vector, and all the strings.
1666          * The total number of pointers is args->na (which is argc + envc)
1667          * plus 4 more: (1) a pointer's worth of space for argc; (2) the NULL
1668          * after the last argument (i.e. argv[argc]); (3) the NULL after the
1669          * last environment variable (i.e. envp[envc]); and (4) the NULL after
1670          * all the strings, at the very top of the stack.
1671          */
1672         size = (args->na + 4) * args->to_ptrsize + args->auxsize +
1673             (args->stk_strp - args->stk_base);
1674
1675         /*
1676          * Pad the string section with zeroes to align the stack size.
1677          */
1678         pad = P2NPHASE(size, args->stk_align);
1679
1680         if (STK_AVAIL(args) < pad)
1681                 return (E2BIG);
1682
1683         args->usrstack_size = size + pad;
1684
1685         while (pad-- != 0)
1686                 *args->stk_strp++ = 0;
1687
1688         args->nc = args->stk_strp - args->stk_base;
1689
1690         return (0);
1691 }
1692
1693 static int
1694 stk_copyout(uarg_t *args, char *usrstack, void **auxvpp, user_t *up)
1695 {
1696         size_t ptrsize = args->to_ptrsize;
1697         ssize_t pslen;
1698         char *kstrp = args->stk_base;
1699         char *ustrp = usrstack - args->nc - ptrsize;
1700         char *usp = usrstack - args->usrstack_size;
1701         int *offp = (int *)(args->stk_base + args->stk_size);
1702         int envc = args->ne;
1703         int argc = args->na - envc;
1704         int i;
1705
1706         /*
1707          * Record argc for /proc.
1708          */
```

```
1709         up->u_argc = argc;
1710
1711         /*
1712          * Put argc on the stack.  Note that even though it's an int,
1713          * it always consumes ptrsize bytes (for alignment).
1714          */
1715         if (stk_putptr(args, usp, (char *)(uintptr_t)argc))
1716                 return (-1);
1717
1718         /*
1719          * Add argc space (ptrsize) to usp and record argv for /proc.
1720          */
1721         up->u_argv = (uintptr_t)(usp += ptrsize);
1722
1723         /*
1724          * Put the argv[] pointers on the stack.
1725          */
1726         for (i = 0; i < argc; i++, usp += ptrsize)
1727                 if (stk_putptr(args, usp, &ustrp[*--offp]))
1728                         return (-1);
1729
1730         /*
1731          * Copy arguments to u_psargs.
1732          */
1733         pslen = MIN(args->arglen, PSARGSZ) - 1;
1734         for (i = 0; i < pslen; i++)
1735                 up->u_psargs[i] = (kstrp[i] == '\0' ? ' ' : kstrp[i]);
1736         while (i < PSARGSZ)
1737                 up->u_psargs[i++] = '\0';
1738
1739         /*
1740          * Add space for argv[]'s NULL terminator (ptrsize) to usp and
1741          * record envp for /proc.
1742          */
1743         up->u_envp = (uintptr_t)(usp += ptrsize);
1744
1745         /*
1746          * Put the envp[] pointers on the stack.
1747          */
1748         for (i = 0; i < envc; i++, usp += ptrsize)
1749                 if (stk_putptr(args, usp, &ustrp[*--offp]))
1750                         return (-1);
1751
1752         /*
1753          * Add space for envp[]'s NULL terminator (ptrsize) to usp and
1754          * remember where the stack ends, which is also where auxv begins.
1755          */
1756         args->stackend = usp += ptrsize;
1757
1758         /*
1759          * Put all the argv[], envp[], and auxv strings on the stack.
1760          */
1761         if (copyout(args->stk_base, ustrp, args->nc))
1762                 return (-1);
1763
1764         /*
1765          * Fill in the aux vector now that we know the user stack addresses
1766          * for the AT_SUN_PLATFORM, AT_SUN_EXECNAME, AT_SUN_BRANDNAME and
1767          * AT_SUN_EMULATOR strings.
1768          */
1769         if (auxvpp != NULL && *auxvpp != NULL) {
1770                 if (args->to_model == DATAMODEL_NATIVE) {
1771                         auxv_t **a = (auxv_t **)auxvpp;
1772                         ADDAUX(*a, AT_SUN_PLATFORM, (long)&ustrp[*--offp])
1773                         ADDAUX(*a, AT_SUN_EXECNAME, (long)&ustrp[*--offp])
1774                         if (args->brandname != NULL)
```



```
1775                                 ADDAUX(*a,
1776                                     AT_SUN_BRANDNAME, (long)&ustrp[*--offp])
1777                         if (args->emulator != NULL)
1778                                 ADDAUX(*a,
1779                                     AT_SUN_EMULATOR, (long)&ustrp[*--offp])
1780                 } else {
1781                         auxv32_t **a = (auxv32_t **)auxvpp;
1782                         ADDAUX(*a,
1783                             AT_SUN_PLATFORM, (int)(uintptr_t)&ustrp[*--offp])
1784                         ADDAUX(*a,
1785                             AT_SUN_EXECNAME, (int)(uintptr_t)&ustrp[*--offp])
1786                         if (args->brandname != NULL)
1787                                 ADDAUX(*a, AT_SUN_BRANDNAME,
1788                                     (int)(uintptr_t)&ustrp[*--offp])
1789                         if (args->emulator != NULL)
1790                                 ADDAUX(*a, AT_SUN_EMULATOR,
1791                                     (int)(uintptr_t)&ustrp[*--offp])
1792                 }
1793         }
1794
1795         return (0);
1796 }
1797
1798 /*
1799  * Though the actual stack base is constant, slew the %sp by a random aligned
1800  * amount in [0,aslr_max_stack_skew).  Mostly, this makes life slightly more
1801  * complicated for buffer overflows hoping to overwrite the return address.
1802  *
1803  * On some platforms this helps avoid cache thrashing when identical processes
1804  * simultaneously share caches that don't provide enough associativity
1805  * (e.g. sun4v systems). In this case stack slewing makes the same hot stack
1806  * variables in different processes live in different cache sets increasing
1807  * effective associativity.
1808  */
1809 size_t
1810 exec_get_spslew(void)
1811 {
1812 #ifdef sun4v
1813         static uint_t sp_color_stride = 16;
1814         static uint_t sp_color_mask = 0x1f;
1815         static uint_t sp_current_color = (uint_t)-1;
1816 #endif
1817         size_t off;
1818
1819         ASSERT(ISP2(aslr_max_stack_skew));
1820
1821         if ((aslr_max_stack_skew == 0) ||
1822             !secflag_enabled(curproc, PROC_SEC_ASLR)) {
1823 #ifdef sun4v
1824                 uint_t spcolor = atomic_inc_32_nv(&sp_current_color);
1825                 return ((size_t)((spcolor & sp_color_mask) * SA(sp_color_stride)
1826 #else
1827                 return (0);
1828 #endif
1829         }
1830
1831         (void) random_get_pseudo_bytes((uint8_t *)&off, sizeof (off));
1832         return SA(P2PHASE(off, aslr_max_stack_skew));
1833 }
1834
1835 /*
1836 #endif /* ! codereview */
1837  * Initialize a new user stack with the specified arguments and environment.
1838  * The initial user stack layout is as follows:
1839  *
1840  *      User Stack
```

```
1841  *      +---------------+ <--- curproc->p_usrstack
1842  *      |               |
1843  *      | slew          |
1844  *      |               |
1845  *      +---------------+
1846  *      | NULL          |
1847  *      +---------------+
1848  *      |               |
1849  *      | auxv strings  |
1850  *      |               |
1851  *      +---------------+
1852  *      |               |
1853  *      | envp strings  |
1854  *      |               |
1855  *      +---------------+
1856  *      |               |
1857  *      | argv strings  |
1858  *      |               |
1859  *      +---------------+ <--- ustrp
1860  *      |               |
1861  *      | aux vector    |
1862  *      |               |
1863  *      +---------------+ <--- auxv
1864  *      | NULL          |
1865  *      +---------------+
1866  *      | envp[envc-1]  |
1867  *      +---------------+
1868  *      | ...           |
1869  *      +---------------+
1870  *      | envp[0]       |
1871  *      +---------------+ <--- envp[]
1872  *      | NULL          |
1873  *      +---------------+
1874  *      | argv[argc-1]  |
1875  *      +---------------+
1876  *      | ...           |
1877  *      +---------------+
1878  *      | argv[0]       |
1879  *      +---------------+ <--- argv[]
1880  *      | argc          |
1881  *      +---------------+ <--- stack base
1882  */
1883 int
1884 exec_args(execa_t *uap, uarg_t *args, intpdata_t *intp, void **auxvpp)
1885 {
1886         size_t size;
1887         int error;
1888         proc_t *p = ttoproc(curthread);
1889         user_t *up = PTOU(p);
1890         char *usrstack;
1891         rctl_entity_p_t e;
1892         struct as *as;
1893         extern int use_stk_lpg;
1894         size_t sp_slew;

1896         args->from_model = p->p_model;
1897         if (p->p_model == DATAMODEL_NATIVE) {
1898                 args->from_ptrsize = sizeof (long);
1899         } else {
1900                 args->from_ptrsize = sizeof (int32_t);
1901         }

1903         if (args->to_model == DATAMODEL_NATIVE) {
1904                 args->to_ptrsize = sizeof (long);
1905                 args->ncargs = NCARGS;
1906                 args->stk_align = STACK_ALIGN;
```



```
1907                 if (args->addr32)
1908                         usrstack = (char *)USRSTACK64_32;
1909                 else
1910                         usrstack = (char *)USRSTACK;
1911         } else {
1912                 args->to_ptrsize = sizeof (int32_t);
1913                 args->ncargs = NCARGS32;
1914                 args->stk_align = STACK_ALIGN32;
1915                 usrstack = (char *)USRSTACK32;
1916         }

1918         ASSERT(P2PHASE((uintptr_t)usrstack, args->stk_align) == 0);

1920 #if defined(__sparc)
1921         /*
1922          * Make sure user register windows are empty before
1923          * attempting to make a new stack.
1924          */
1925         (void) flush_user_windows_to_stack(NULL);
1926 #endif

1928         for (size = PAGESIZE; ; size *= 2) {
1929                 args->stk_size = size;
1930                 args->stk_base = kmem_alloc(size, KM_SLEEP);
1931                 args->stk_strp = args->stk_base;
1932                 args->stk_offp = (int *)(args->stk_base + size);
1933                 error = stk_copyin(uap, args, intp, auxvpp);
1934                 if (error == 0)
1935                         break;
1936                 kmem_free(args->stk_base, size);
1937                 if (error != E2BIG && error != ENAMETOOLONG)
1938                         return (error);
1939                 if (size >= args->ncargs)
1940                         return (E2BIG);
1941         }

1943         size = args->usrstack_size;

1945         ASSERT(error == 0);
1946         ASSERT(P2PHASE(size, args->stk_align) == 0);
1947         ASSERT((ssize_t)STK_AVAIL(args) >= 0);

1949         if (size > args->ncargs) {
1950                 kmem_free(args->stk_base, args->stk_size);
1951                 return (E2BIG);
1952         }

1954         /*
1955          * Leave only the current lwp and force the other lwps to exit.
1956          * If another lwp beat us to the punch by calling exit(), bail out.
1957          */
1958         if ((error = exitlwps(0)) != 0) {
1959                 kmem_free(args->stk_base, args->stk_size);
1960                 return (error);
1961         }

1963         /*
1964          * Revoke any doors created by the process.
1965          */
1966         if (p->p_door_list)
1967                 door_exit();

1969         /*
1970          * Release schedctl data structures.
1971          */
1972         if (p->p_pagep)
```

```
1973                schedctl_proc_cleanup();

1975        /*
1976         * Clean up any DTrace helpers for the process.
1977         */
1978        if (p->p_dtrace_helpers != NULL) {
1979                ASSERT(dtrace_helpers_cleanup != NULL);
1980                (*dtrace_helpers_cleanup)();
1981        }

1983        mutex_enter(&p->p_lock);
1984        /*
1985         * Cleanup the DTrace provider associated with this process.
1986         */
1987        if (p->p_dtrace_probes) {
1988                ASSERT(dtrace_fasttrap_exec_ptr != NULL);
1989                dtrace_fasttrap_exec_ptr(p);
1990        }
1991        mutex_exit(&p->p_lock);

1993        /*
1994         * discard the lwpchan cache.
1995         */
1996        if (p->p_lcp != NULL)
1997                lwpchan_destroy_cache(1);

1999        /*
2000         * Delete the POSIX timers.
2001         */
2002        if (p->p_itimer != NULL)
2003                timer_exit();

2005        /*
2006         * Delete the ITIMER_REALPROF interval timer.
2007         * The other ITIMER_* interval timers are specified
2008         * to be inherited across exec().
2009         */
2010        delete_itimer_realprof();

2012        if (AU_AUDITING())
2013                audit_exec(args->stk_base, args->stk_base + args->arglen,
2014                    args->na - args->ne, args->ne, args->pfcred);

2016        /*
2017         * Ensure that we don't change resource associations while we
2018         * change address spaces.
2019         */
2020        mutex_enter(&p->p_lock);
2021        pool_barrier_enter();
2022        mutex_exit(&p->p_lock);

2024        /*
2025         * Destroy the old address space and create a new one.
2026         * From here on, any errors are fatal to the exec()ing process.
2027         * On error we return -1, which means the caller must SIGKILL
2028         * the process.
2029         */
2030        relvm();

2032        mutex_enter(&p->p_lock);
2033        pool_barrier_exit();
2034        mutex_exit(&p->p_lock);

2036        up->u_execsw = args->execswp;

2038        p->p_brkbase = NULL;
```



```
2039        p->p_brksize = 0;
2040        p->p_brkpageszc = 0;
2041        p->p_stksize = 0;
2042        p->p_stkpageszc = 0;
2043        p->p_model = args->to_model;
2044        p->p_usrstack = usrstack;
2045        p->p_stkprot = args->stk_prot;
2046        p->p_datprot = args->dat_prot;

2048        /*
2049         * Reset resource controls such that all controls are again active as
2050         * well as appropriate to the potentially new address model for the
2051         * process.
2052         */
2053        e.rcep_p.proc = p;
2054        e.rcep_t = RCENTITY_PROCESS;
2055        rctl_set_reset(p->p_rctls, p, &e);

2057        /* Too early to call map_pgsz for the heap */
2058        if (use_stk_lpg) {
2059                p->p_stkpageszc = page_szc(map_pgsz(MAPPGSZ_STK, p, 0, 0, 0));
2060        }

2062        mutex_enter(&p->p_lock);
2063        p->p_flag |= SAUTOLPG;  /* kernel controls page sizes */
2064        mutex_exit(&p->p_lock);

  72        /*
  73         * Some platforms may choose to randomize real stack start by adding a
  74         * small slew (not more than a few hundred bytes) to the top of the
  75         * stack. This helps avoid cache thrashing when identical processes
  76         * simultaneously share caches that don't provide enough associativity
  77         * (e.g. sun4v systems). In this case stack slewing makes the same hot
  78         * stack variables in different processes to live in different cache
  79         * sets increasing effective associativity.
  80         */
2066        sp_slew = exec_get_spslew();
2067        ASSERT(P2PHASE(sp_slew, args->stk_align) == 0);
2068        /* Be certain we don't underflow */
2069        VERIFY((curproc->p_usrstack - (size + sp_slew)) < curproc->p_usrstack);
2070 #endif /* ! codereview */
2071        exec_set_sp(size + sp_slew);

2073        as = as_alloc();
2074        p->p_as = as;
2075        as->a_proc = p;
2076        if (p->p_model == DATAMODEL_ILP32 || args->addr32)
2077                as->a_userlimit = (caddr_t)USERLIMIT32;
2078        (void) hat_setup(as->a_hat, HAT_ALLOC);
2079        hat_join_srd(as->a_hat, args->ex_vp);

2081        /*
2082         * Finally, write out the contents of the new stack.
2083         */
2084        error = stk_copyout(args, usrstack - sp_slew, auxvpp, up);
2085        kmem_free(args->stk_base, args->stk_size);
2086        return (error);
2087 }
```

```
**********************************************************
   37306 Wed May 27 19:49:28 2015
new/usr/src/uts/common/os/fork.c
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
______unchanged_portion_omitted_

 925 /*
 926  * create a child proc struct.
 927  */
 928 static int
 929 getproc(proc_t **cpp, pid_t pid, uint_t flags)
 930 {
 931         proc_t          *pp, *cp;
 932         pid_t           newpid;
 933         struct user     *uarea;
 934         extern uint_t   nproc;
 935         struct cred     *cr;
 936         uid_t           ruid;
 937         zoneid_t        zoneid;
 938         task_t          *task;
 939         kproject_t      *proj;
 940         zone_t          *zone;
 941         int             rctlfail = 0;

 943         if (zone_status_get(curproc->p_zone) >= ZONE_IS_SHUTTING_DOWN)
 944                 return (-1);    /* no point in starting new processes */

 946         pp = (flags & GETPROC_KERNEL) ? &p0 : curproc;
 947         task = pp->p_task;
 948         proj = task->tk_proj;
 949         zone = pp->p_zone;

 951         mutex_enter(&pp->p_lock);
 952         mutex_enter(&zone->zone_nlwps_lock);
 953         if (proj != proj0p) {
 954                 if (task->tk_nprocs >= task->tk_nprocs_ctl)
 955                         if (rctl_test(rc_task_nprocs, task->tk_rctls,
 956                             pp, 1, 0) & RCT_DENY)
 957                                 rctlfail = 1;

 959                 if (proj->kpj_nprocs >= proj->kpj_nprocs_ctl)
 960                         if (rctl_test(rc_project_nprocs, proj->kpj_rctls,
 961                             pp, 1, 0) & RCT_DENY)
 962                                 rctlfail = 1;

 964                 if (zone->zone_nprocs >= zone->zone_nprocs_ctl)
 965                         if (rctl_test(rc_zone_nprocs, zone->zone_rctls,
 966                             pp, 1, 0) & RCT_DENY)
 967                                 rctlfail = 1;

 969                 if (rctlfail) {
 970                         mutex_exit(&zone->zone_nlwps_lock);
 971                         mutex_exit(&pp->p_lock);
 972                         atomic_inc_32(&zone->zone_ffcap);
 973                         goto punish;
 974                 }
 975         }
 976         task->tk_nprocs++;
 977         proj->kpj_nprocs++;
 978         zone->zone_nprocs++;
 979         mutex_exit(&zone->zone_nlwps_lock);
 980         mutex_exit(&pp->p_lock);
```



```
 982         cp = kmem_cache_alloc(process_cache, KM_SLEEP);
 983         bzero(cp, sizeof (proc_t));

 985         /*
 986          * Make proc entry for child process
 987          */
 988         mutex_init(&cp->p_splock, NULL, MUTEX_DEFAULT, NULL);
 989         mutex_init(&cp->p_crlock, NULL, MUTEX_DEFAULT, NULL);
 990         mutex_init(&cp->p_pflock, NULL, MUTEX_DEFAULT, NULL);
 991 #if defined(__x86)
 992         mutex_init(&cp->p_ldtlock, NULL, MUTEX_DEFAULT, NULL);
 993 #endif
 994         mutex_init(&cp->p_maplock, NULL, MUTEX_DEFAULT, NULL);
 995         cp->p_stat = SIDL;
 996         cp->p_mstart = gethrtime();
 997         cp->p_as = &kas;
 998         /*
 999          * p_zone must be set before we call pid_allocate since the process
1000          * will be visible after that and code such as prfind_zone will
1001          * look at the p_zone field.
1002          */
1003         cp->p_zone = pp->p_zone;
1004         cp->p_t1_lgrpid = LGRP_NONE;
1005         cp->p_tr_lgrpid = LGRP_NONE;

1007         if ((newpid = pid_allocate(cp, pid, PID_ALLOC_PROC)) == -1) {
1008                 if (nproc == v.v_proc) {
1009                         CPU_STATS_ADDQ(CPU, sys, procovf, 1);
1010                         cmn_err(CE_WARN, "out of processes");
1011                 }
1012                 goto bad;
1013         }

1015         mutex_enter(&pp->p_lock);
1016         cp->p_exec = pp->p_exec;
1017         cp->p_execdir = pp->p_execdir;
1018         mutex_exit(&pp->p_lock);

1020         if (cp->p_exec) {
1021                 VN_HOLD(cp->p_exec);
1022                 /*
1023                  * Each VOP_OPEN() must be paired with a corresponding
1024                  * VOP_CLOSE(). In this case, the executable will be
1025                  * closed for the child in either proc_exit() or gexec().
1026                  */
1027                 if (VOP_OPEN(&cp->p_exec, FREAD, CRED(), NULL) != 0) {
1028                         VN_RELE(cp->p_exec);
1029                         cp->p_exec = NULLVP;
1030                         cp->p_execdir = NULLVP;
1031                         goto bad;
1032                 }
1033         }
1034         if (cp->p_execdir)
1035                 VN_HOLD(cp->p_execdir);

1037         /*
1038          * If not privileged make sure that this user hasn't exceeded
1039          * v.v_maxup processes, and that users collectively haven't
1040          * exceeded v.v_maxupttl processes.
1041          */
1042         mutex_enter(&pidlock);
1043         ASSERT(nproc < v.v_proc);       /* otherwise how'd we get our pid? */
1044         cr = CRED();
1045         ruid = crgetruid(cr);
1046         zoneid = crgetzoneid(cr);
```

```
1047         if (nproc >= v.v_maxup &&       /* short-circuit; usually false */
1048             (nproc >= v.v_maxupttl ||
1049             upcount_get(ruid, zoneid) >= v.v_maxup) &&
1050             secpolicy_newproc(cr) != 0) {
1051                 mutex_exit(&pidlock);
1052                 zcmn_err(zoneid, CE_NOTE,
1053                     "out of per-user processes for uid %d", ruid);
1054                 goto bad;
1055         }

1057         /*
1058          * Everything is cool, put the new proc on the active process list.
1059          * It is already on the pid list and in /proc.
1060          * Increment the per uid process count (upcount).
1061          */
1062         nproc++;
1063         upcount_inc(ruid, zoneid);

1065         cp->p_next = practive;
1066         practive->p_prev = cp;
1067         practive = cp;

1069         cp->p_ignore = pp->p_ignore;
1070         cp->p_siginfo = pp->p_siginfo;
1071         cp->p_flag = pp->p_flag & (SJCTL|SNOWAIT|SNOCD);
1072         cp->p_sessp = pp->p_sessp;
1073         sess_hold(pp);
1074         cp->p_brand = pp->p_brand;
1075         if (PROC_IS_BRANDED(pp))
1076                 BROP(pp)->b_copy_procdata(cp, pp);
1077         cp->p_bssbase = pp->p_bssbase;
1078         cp->p_brkbase = pp->p_brkbase;
1079         cp->p_brksize = pp->p_brksize;
1080         cp->p_brkpageszc = pp->p_brkpageszc;
1081         cp->p_stksize = pp->p_stksize;
1082         cp->p_stkpageszc = pp->p_stkpageszc;
1083         cp->p_stkprot = pp->p_stkprot;
1084         cp->p_datprot = pp->p_datprot;
1085         cp->p_usrstack = pp->p_usrstack;
1086         cp->p_model = pp->p_model;
1087         cp->p_ppid = pp->p_pid;
1088         cp->p_ancpid = pp->p_pid;
1089         cp->p_portcnt = pp->p_portcnt;
1090         /*
1091          * Security flags are preserved on fork, the inherited copy come into
1092          * effect on exec
1093          */
1094         bcopy(&pp->p_secflags, &cp->p_secflags, sizeof (psecflags_t));
1095 #endif /* ! codereview */

1097         /*
1098          * Initialize watchpoint structures
1099          */
1100         avl_create(&cp->p_warea, wa_compare, sizeof (struct watched_area),
1101             offsetof(struct watched_area, wa_link));

1103         /*
1104          * Initialize immediate resource control values.
1105          */
1106         cp->p_stk_ctl = pp->p_stk_ctl;
1107         cp->p_fsz_ctl = pp->p_fsz_ctl;
1108         cp->p_vmem_ctl = pp->p_vmem_ctl;
1109         cp->p_fno_ctl = pp->p_fno_ctl;

1111         /*
1112          * Link up to parent-child-sibling chain.  No need to lock
```



```
1113          * in general since only a call to freeproc() (done by the
1114          * same parent as newproc()) diddles with the child chain.
1115          */
1116         cp->p_sibling = pp->p_child;
1117         if (pp->p_child)
1118                 pp->p_child->p_psibling = cp;

1120         cp->p_parent = pp;
1121         pp->p_child = cp;

1123         cp->p_child_ns = NULL;
1124         cp->p_sibling_ns = NULL;

1126         cp->p_nextorph = pp->p_orphan;
1127         cp->p_nextofkin = pp;
1128         pp->p_orphan = cp;

1130         /*
1131          * Inherit profiling state; do not inherit REALPROF profiling state.
1132          */
1133         cp->p_prof = pp->p_prof;
1134         cp->p_rprof_cyclic = CYCLIC_NONE;

1136         /*
1137          * Inherit pool pointer from the parent.  Kernel processes are
1138          * always bound to the default pool.
1139          */
1140         mutex_enter(&pp->p_lock);
1141         if (flags & GETPROC_KERNEL) {
1142                 cp->p_pool = pool_default;
1143                 cp->p_flag |= SSYS;
1144         } else {
1145                 cp->p_pool = pp->p_pool;
1146         }
1147         atomic_inc_32(&cp->p_pool->pool_ref);
1148         mutex_exit(&pp->p_lock);

1150         /*
1151          * Add the child process to the current task.  Kernel processes
1152          * are always attached to task0.
1153          */
1154         mutex_enter(&cp->p_lock);
1155         if (flags & GETPROC_KERNEL)
1156                 task_attach(task0p, cp);
1157         else
1158                 task_attach(pp->p_task, cp);
1159         mutex_exit(&cp->p_lock);
1160         mutex_exit(&pidlock);

1162         avl_create(&cp->p_ct_held, contract_compar, sizeof (contract_t),
1163             offsetof(contract_t, ct_ctlist));

1165         /*
1166          * Duplicate any audit information kept in the process table
1167          */
1168         if (audit_active)       /* copy audit data to cp */
1169                 audit_newproc(cp);

1171         crhold(cp->p_cred = cr);

1173         /*
1174          * Bump up the counts on the file structures pointed at by the
1175          * parent's file table since the child will point at them too.
1176          */
1177         fcnt_add(P_FINFO(pp), 1);
```

```
1179         if (PTOU(pp)->u_cdir) {
1180                 VN_HOLD(PTOU(pp)->u_cdir);
1181         } else {
1182                 ASSERT(pp == &p0);
1183                 /*
1184                  * We must be at or before vfs_mountroot(); it will take care of
1185                  * assigning our current directory.
1186                  */
1187         }
1188         if (PTOU(pp)->u_rdir)
1189                 VN_HOLD(PTOU(pp)->u_rdir);
1190         if (PTOU(pp)->u_cwd)
1191                 refstr_hold(PTOU(pp)->u_cwd);

1193         /*
1194          * copy the parent's uarea.
1195          */
1196         uarea = PTOU(cp);
1197         bcopy(PTOU(pp), uarea, sizeof (*uarea));
1198         flist_fork(P_FINFO(pp), P_FINFO(cp));

1200         gethrestime(&uarea->u_start);
1201         uarea->u_ticks = ddi_get_lbolt();
1202         uarea->u_mem = rm_asrss(pp->p_as);
1203         uarea->u_acflag = AFORK;

1205         /*
1206          * If inherit-on-fork, copy /proc tracing flags to child.
1207          */
1208         if ((pp->p_proc_flag & P_PR_FORK) != 0) {
1209                 cp->p_proc_flag |= pp->p_proc_flag & (P_PR_TRACE|P_PR_FORK);
1210                 cp->p_sigmask = pp->p_sigmask;
1211                 cp->p_fltmask = pp->p_fltmask;
1212         } else {
1213                 sigemptyset(&cp->p_sigmask);
1214                 premptyset(&cp->p_fltmask);
1215                 uarea->u_systrap = 0;
1216                 premptyset(&uarea->u_entrymask);
1217                 premptyset(&uarea->u_exitmask);
1218         }
1219         /*
1220          * If microstate accounting is being inherited, mark child
1221          */
1222         if ((pp->p_flag & SMSFORK) != 0)
1223                 cp->p_flag |= pp->p_flag & (SMSFORK|SMSACCT);

1225         /*
1226          * Inherit fixalignment flag from the parent
1227          */
1228         cp->p_fixalignment = pp->p_fixalignment;

1230         *cpp = cp;
1231         return (0);

1233 bad:
1234         ASSERT(MUTEX_NOT_HELD(&pidlock));

1236         mutex_destroy(&cp->p_crlock);
1237         mutex_destroy(&cp->p_pflock);
1238 #if defined(__x86)
1239         mutex_destroy(&cp->p_ldtlock);
1240 #endif
1241         if (newpid != -1) {
1242                 proc_entry_free(cp->p_pidp);
1243                 (void) pid_rele(cp->p_pidp);
1244         }
```



```
1245         kmem_cache_free(process_cache, cp);

1247         mutex_enter(&zone->zone_nlwps_lock);
1248         task->tk_nprocs--;
1249         proj->kpj_nprocs--;
1250         zone->zone_nprocs--;
1251         mutex_exit(&zone->zone_nlwps_lock);
1252         atomic_inc_32(&zone->zone_ffnoproc);

1254 punish:
1255         /*
1256          * We most likely got into this situation because some process is
1257          * forking out of control.  As punishment, put it to sleep for a
1258          * bit so it can't eat the machine alive.  Sleep interval is chosen
1259          * to allow no more than one fork failure per cpu per clock tick
1260          * on average (yes, I just made this up).  This has two desirable
1261          * properties: (1) it sets a constant limit on the fork failure
1262          * rate, and (2) the busier the system is, the harsher the penalty
1263          * for abusing it becomes.
1264          */
1265         INCR_COUNT(&fork_fail_pending, &pidlock);
1266         delay(fork_fail_pending / ncpus + 1);
1267         DECR_COUNT(&fork_fail_pending, &pidlock);

1269         return (-1); /* out of memory or proc slots */
1270 }

1272 /*
1273  * Release virtual memory.
1274  * In the case of vfork(), the child was given exclusive access to its
1275  * parent's address space.  The parent is waiting in vfwait() for the
1276  * child to release its exclusive claim via relvm().
1277  */
1278 void
1279 relvm()
1280 {
1281         proc_t *p = curproc;

1283         ASSERT((unsigned)p->p_lwpcnt <= 1);

1285         prrelvm();      /* inform /proc */

1287         if (p->p_flag & SVFORK) {
1288                 proc_t *pp = p->p_parent;
1289                 /*
1290                  * The child process is either exec'ing or exit'ing.
1291                  * The child is now separated from the parent's address
1292                  * space.  The parent process is made dispatchable.
1293                  *
1294                  * This is a delicate locking maneuver, involving
1295                  * both the parent's p_lock and the child's p_lock.
1296                  * As soon as the SVFORK flag is turned off, the
1297                  * parent is free to run, but it must not run until
1298                  * we wake it up using its p_cv because it might
1299                  * exit and we would be referencing invalid memory.
1300                  * Therefore, we hold the parent with its p_lock
1301                  * while protecting our p_flags with our own p_lock.
1302                  */
1303 try_again:
1304                 mutex_enter(&p->p_lock);        /* grab child's lock first */
1305                 prbarrier(p);           /* make sure /proc is blocked out */
1306                 mutex_enter(&pp->p_lock);

1308                 /*
1309                  * Check if parent is locked by /proc.
1310                  */
```

```
1311                if (pp->p_proc_flag & P_PR_LOCK) {
1312                        /*
1313                         * Delay until /proc is done with the parent.
1314                         * We must drop our (the child's) p->p_lock, wait
1315                         * via prbarrier() on the parent, then start over.
1316                         */
1317                        mutex_exit(&p->p_lock);
1318                        prbarrier(pp);
1319                        mutex_exit(&pp->p_lock);
1320                        goto try_again;
1321                }
1322                p->p_flag &= ~SVFORK;
1323                kpreempt_disable();
1324                p->p_as = &kas;

1326                /*
1327                 * notify hat of change in thread's address space
1328                 */
1329                hat_thread_exit(curthread);
1330                kpreempt_enable();

1332                /*
1333                 * child sizes are copied back to parent because
1334                 * child may have grown.
1335                 */
1336                pp->p_brkbase = p->p_brkbase;
1337                pp->p_brksize = p->p_brksize;
1338                pp->p_stksize = p->p_stksize;

1340                /*
1341                 * Copy back the shm accounting information
1342                 * to the parent process.
1343                 */
1344                pp->p_segacct = p->p_segacct;
1345                p->p_segacct = NULL;

1347                /*
1348                 * The parent is no longer waiting for the vfork()d child.
1349                 * Restore the parent's watched pages, if any.  This is
1350                 * safe because we know the parent is not locked by /proc
1351                 */
1352                pp->p_flag &= ~SVFWAIT;
1353                if (avl_numnodes(&pp->p_wpage) != 0) {
1354                        pp->p_as->a_wpage = pp->p_wpage;
1355                        avl_create(&pp->p_wpage, wp_compare,
1356                            sizeof (struct watched_page),
1357                            offsetof(struct watched_page, wp_link));
1358                }
1359                cv_signal(&pp->p_cv);
1360                mutex_exit(&pp->p_lock);
1361                mutex_exit(&p->p_lock);
1362        } else {
1363                if (p->p_as != &kas) {
1364                        struct as *as;

1366                        if (p->p_segacct)
1367                                shmexit(p);

1369                        /*
1370                         * We grab p_lock for the benefit of /proc
1371                         */
1372                        kpreempt_disable();
1373                        mutex_enter(&p->p_lock);
1374                        prbarrier(p);   /* make sure /proc is blocked out */
1375                        as = p->p_as;
1376                        p->p_as = &kas;
```



```
1377                        mutex_exit(&p->p_lock);

1379                        /*
1380                         * notify hat of change in thread's address space
1381                         */
1382                        hat_thread_exit(curthread);
1383                        kpreempt_enable();

1385                        as_free(as);
1386                        p->p_tr_lgrpid = LGRP_NONE;
1387                }
1388        }
1389 }

1391 /*
1392  * Wait for child to exec or exit.
1393  * Called by parent of vfork'ed process.
1394  * See important comments in relvm(), above.
1395  */
1396 void
1397 vfwait(pid_t pid)
1398 {
1399         int signalled = 0;
1400         proc_t *pp = ttoproc(curthread);
1401         proc_t *cp;

1403         /*
1404          * Wait for child to exec or exit.
1405          */
1406         for (;;) {
1407                 mutex_enter(&pidlock);
1408                 cp = prfind(pid);
1409                 if (cp == NULL || cp->p_parent != pp) {
1410                         /*
1411                          * Child has exit()ed.
1412                          */
1413                         mutex_exit(&pidlock);
1414                         break;
1415                 }
1416                 /*
1417                  * Grab the child's p_lock before releasing pidlock.
1418                  * Otherwise, the child could exit and we would be
1419                  * referencing invalid memory.
1420                  */
1421                 mutex_enter(&cp->p_lock);
1422                 mutex_exit(&pidlock);
1423                 if (!(cp->p_flag & SVFORK)) {
1424                         /*
1425                          * Child has exec()ed or is exit()ing.
1426                          */
1427                         mutex_exit(&cp->p_lock);
1428                         break;
1429                 }
1430                 mutex_enter(&pp->p_lock);
1431                 mutex_exit(&cp->p_lock);
1432                 /*
1433                  * We might be waked up spuriously from the cv_wait().
1434                  * We have to do the whole operation over again to be
1435                  * sure the child's SVFORK flag really is turned off.
1436                  * We cannot make reference to the child because it can
1437                  * exit before we return and we would be referencing
1438                  * invalid memory.
1439                  *
1440                  * Because this is potentially a very long-term wait,
1441                  * we call cv_wait_sig() (for its jobcontrol and /proc
1442                  * side-effects) unless there is a current signal, in
```

```
1443                  * which case we use cv_wait() because we cannot return
1444                  * from this function until the child has released the
1445                  * address space.  Calling cv_wait_sig() with a current
1446                  * signal would lead to an indefinite loop here because
1447                  * cv_wait_sig() returns immediately in this case.
1448                  */
1449                 if (signalled)
1450                         cv_wait(&pp->p_cv, &pp->p_lock);
1451                 else
1452                         signalled = !cv_wait_sig(&pp->p_cv, &pp->p_lock);
1453                 mutex_exit(&pp->p_lock);
1454         }

1456         /* restore watchpoints to parent */
1457         if (pr_watch_active(pp)) {
1458                 struct as *as = pp->p_as;
1459                 AS_LOCK_ENTER(as, &as->a_lock, RW_WRITER);
1460                 as_setwatch(as);
1461                 AS_LOCK_EXIT(as, &as->a_lock);
1462         }

1464         mutex_enter(&pp->p_lock);
1465         prbarrier(pp);  /* barrier against /proc locking */
1466         continuelwps(pp);
1467         mutex_exit(&pp->p_lock);
1468 }
```

```
**********************************************************
   26456 Wed May 27 19:49:29 2015
new/usr/src/uts/common/os/grow.c
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
libc: adjust brk(0) to return the existing break, and use it to initialize sbrk(
**********************************************************
```

```
   1 /*
   2  * CDDL HEADER START
   3  *
   4  * The contents of this file are subject to the terms of the
   5  * Common Development and Distribution License (the "License").
   6  * You may not use this file except in compliance with the License.
   7  *
   8  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
   9  * or http://www.opensolaris.org/os/licensing.
  10  * See the License for the specific language governing permissions
  11  * and limitations under the License.
  12  *
  13  * When distributing Covered Code, include this CDDL HEADER in each
  14  * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  15  * If applicable, add the following below this CDDL HEADER, with the
  16  * fields enclosed by brackets "[]" replaced with your own identifying
  17  * information: Portions Copyright [yyyy] [name of copyright owner]
  18  *
  19  * CDDL HEADER END
  20  */

  22 /* Copyright 2013 OmniTI Computer Consulting, Inc. All rights reserved. */

  24 /*
  25  * Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
  26  * Use is subject to license terms.
  27  */

  29 /*      Copyright (c) 1984, 1986, 1987, 1988, 1989 AT&T */
  30 /*        All Rights Reserved   */

  32 #include <sys/types.h>
  33 #include <sys/inttypes.h>
  34 #include <sys/param.h>
  35 #include <sys/sysmacros.h>
  36 #include <sys/systm.h>
  37 #include <sys/signal.h>
  38 #include <sys/user.h>
  39 #include <sys/errno.h>
  40 #include <sys/var.h>
  41 #include <sys/proc.h>
  42 #include <sys/tuneable.h>
  43 #include <sys/debug.h>
  44 #include <sys/cmn_err.h>
  45 #include <sys/cred.h>
  46 #include <sys/vnode.h>
  47 #include <sys/vfs.h>
  48 #include <sys/vm.h>
  49 #include <sys/file.h>
  50 #include <sys/mman.h>
  51 #include <sys/vmparam.h>
  52 #include <sys/fcntl.h>
  53 #include <sys/lwpchan_impl.h>
  54 #include <sys/nbmlock.h>

  56 #include <vm/hat.h>
  57 #include <vm/as.h>
```



```
  58 #include <vm/seg.h>
  59 #include <vm/seg_dev.h>
  60 #include <vm/seg_vn.h>

  62 int use_brk_lpg = 1;
  63 int use_stk_lpg = 1;

  65 static int brk_lpg(caddr_t nva);
  66 static int grow_lpg(caddr_t sp);

  68 intptr_t
  68 int
  69 brk(caddr_t nva)
  70 {
  71         int error;
  72         proc_t *p = curproc;

  74         /*
  75          * As a special case to aid the implementation of sbrk(3C), if given a
  76          * new brk of 0, return the current brk.  We'll hide this in brk(3C).
  77          */
  78         if (nva == 0)
  79                 return ((intptr_t)(p->p_brkbase + p->p_brksize));

  81         /*
  82 #endif /* ! codereview */
  83          * Serialize brk operations on an address space.
  84          * This also serves as the lock protecting p_brksize
  85          * and p_brkpageszc.
  86          */
  87         as_rangelock(p->p_as);
  88         if (use_brk_lpg && (p->p_flag & SAUTOLPG) != 0) {
  89                 error = brk_lpg(nva);
  90         } else {
  91                 error = brk_internal(nva, p->p_brkpageszc);
  92         }
  93         as_rangeunlock(p->p_as);
  94         return ((error != 0 ? set_errno(error) : 0));
  95 }

  97 /*
  98  * Algorithm: call arch-specific map_pgsz to get best page size to use,
  99  * then call brk_internal().
 100  * Returns 0 on success.
 101  */
 102 static int
 103 brk_lpg(caddr_t nva)
 104 {
 105         struct proc *p = curproc;
 106         size_t pgsz, len;
 107         caddr_t addr, brkend;
 108         caddr_t bssbase = p->p_bssbase;
 109         caddr_t brkbase = p->p_brkbase;
 110         int oszc, szc;
 111         int err;

 113         oszc = p->p_brkpageszc;

 115         /*
 116          * If p_brkbase has not yet been set, the first call
 117          * to brk_internal() will initialize it.
 118          */
 119         if (brkbase == 0) {
 120                 return (brk_internal(nva, oszc));
 121         }
```

```
123         len = nva - bssbase;
124
125         pgsz = map_pgsz(MAPPGSZ_HEAP, p, bssbase, len, 0);
126         szc = page_szc(pgsz);
127
128         /*
129          * Covers two cases:
130          * 1. page_szc() returns -1 for invalid page size, so we want to
131          * ignore it in that case.
132          * 2. By design we never decrease page size, as it is more stable.
133          */
134         if (szc <= oszc) {
135                 err = brk_internal(nva, oszc);
136                 /* If failed, back off to base page size. */
137                 if (err != 0 && oszc != 0) {
138                         err = brk_internal(nva, 0);
139                 }
140                 return (err);
141         }
142
143         err = brk_internal(nva, szc);
144         /* If using szc failed, map with base page size and return. */
145         if (err != 0) {
146                 if (szc != 0) {
147                         err = brk_internal(nva, 0);
148                 }
149                 return (err);
150         }
151
152         /*
153          * Round up brk base to a large page boundary and remap
154          * anything in the segment already faulted in beyond that
155          * point.
156          */
157         addr = (caddr_t)P2ROUNDUP((uintptr_t)p->p_bssbase, pgsz);
158         brkend = brkbase + p->p_brksize;
159         len = brkend - addr;
160         /* Check that len is not negative. Update page size code for heap. */
161         if (addr >= p->p_bssbase && brkend > addr && IS_P2ALIGNED(len, pgsz)) {
162                 (void) as_setpagesize(p->p_as, addr, len, szc, B_FALSE);
163                 p->p_brkpageszc = szc;
164         }
165
166         ASSERT(err == 0);
167         return (err);           /* should always be 0 */
168 }
169
170 /*
171  * Returns 0 on success.
172  */
173 int
174 brk_internal(caddr_t nva, uint_t brkszc)
175 {
176         caddr_t ova;                    /* current break address */
177         size_t size;
178         int     error;
179         struct proc *p = curproc;
180         struct as *as = p->p_as;
181         size_t pgsz;
182         uint_t szc;
183         rctl_qty_t as_rctl;
184
185         /*
186          * extend heap to brkszc alignment but use current p->p_brkpageszc
187          * for the newly created segment. This allows the new extension
188          * segment to be concatenated successfully with the existing brk
```



```
189          * segment.
190          */
191         if ((szc = brkszc) != 0) {
192                 pgsz = page_get_pagesize(szc);
193                 ASSERT(pgsz > PAGESIZE);
194         } else {
195                 pgsz = PAGESIZE;
196         }
197
198         mutex_enter(&p->p_lock);
199         as_rctl = rctl_enforced_value(rctlproc_legacy[RLIMIT_DATA],
200             p->p_rctls, p);
201         mutex_exit(&p->p_lock);
202
203         /*
204          * If p_brkbase has not yet been set, the first call
205          * to brk() will initialize it.
206          */
207         if (p->p_brkbase == 0)
208                 p->p_brkbase = nva;
209
210         /*
211          * Before multiple page size support existed p_brksize was the value
212          * not rounded to the pagesize (i.e. it stored the exact user request
213          * for heap size). If pgsz is greater than PAGESIZE calculate the
214          * heap size as the real new heap size by rounding it up to pgsz.
215          * This is useful since we may want to know where the heap ends
216          * without knowing heap pagesize (e.g. some old code) and also if
217          * heap pagesize changes we can update p_brkpageszc but delay adding
218          * new mapping yet still know from p_brksize where the heap really
219          * ends. The user requested heap end is stored in libc variable.
220          */
221         if (pgsz > PAGESIZE) {
222                 caddr_t tnva = (caddr_t)P2ROUNDUP((uintptr_t)nva, pgsz);
223                 size = tnva - p->p_brkbase;
224                 if (tnva < p->p_brkbase || (size > p->p_brksize &&
225                     size > (size_t)as_rctl)) {
226                         szc = 0;
227                         pgsz = PAGESIZE;
228                         size = nva - p->p_brkbase;
229                 }
230         } else {
231                 size = nva - p->p_brkbase;
232         }
233
234         /*
235          * use PAGESIZE to roundup ova because we want to know the real value
236          * of the current heap end in case p_brkpageszc changes since the last
237          * p_brksize was computed.
238          */
239         nva = (caddr_t)P2ROUNDUP((uintptr_t)nva, pgsz);
240         ova = (caddr_t)P2ROUNDUP((uintptr_t)(p->p_brkbase + p->p_brksize),
241             PAGESIZE);
242
243         if ((nva < p->p_brkbase) || (size > p->p_brksize &&
244             size > as_rctl)) {
245                 mutex_enter(&p->p_lock);
246                 (void) rctl_action(rctlproc_legacy[RLIMIT_DATA], p->p_rctls, p,
247                     RCA_SAFE);
248                 mutex_exit(&p->p_lock);
249                 return (ENOMEM);
250         }
251
252         if (nva > ova) {
253                 struct segvn_crargs crargs =
254                     SEGVN_ZFOD_ARGS(PROT_ZFOD, PROT_ALL);
```

```
256                 if (!(p->p_datprot & PROT_EXEC)) {
257                         crargs.prot &= ~PROT_EXEC;
258                 }
259
260                 /*
261                  * Add new zfod mapping to extend UNIX data segment
262                  * AS_MAP_NO_LPOOB means use 0, and don't reapply OOB policies
263                  * via map_pgszcvec(). Use AS_MAP_HEAP to get intermediate
264                  * page sizes if ova is not aligned to szc's pgsz.
265                  */
266                 if (szc > 0) {
267                         caddr_t rbss;
268
269                         rbss = (caddr_t)P2ROUNDUP((uintptr_t)p->p_bssbase,
270                             pgsz);
271                         if (IS_P2ALIGNED(p->p_bssbase, pgsz) || ova > rbss) {
272                                 crargs.szc = p->p_brkpageszc ? p->p_brkpageszc :
273                                     AS_MAP_NO_LPOOB;
274                         } else if (ova == rbss) {
275                                 crargs.szc = szc;
276                         } else {
277                                 crargs.szc = AS_MAP_HEAP;
278                         }
279                 } else {
280                         crargs.szc = AS_MAP_NO_LPOOB;
281                 }
282                 crargs.lgrp_mem_policy_flags = LGRP_MP_FLAG_EXTEND_UP;
283                 error = as_map(as, ova, (size_t)(nva - ova), segvn_create,
284                     &crargs);
285                 if (error) {
286                         return (error);
287                 }
288
289         } else if (nva < ova) {
290                 /*
291                  * Release mapping to shrink UNIX data segment.
292                  */
293                 (void) as_unmap(as, nva, (size_t)(ova - nva));
294         }
295         p->p_brksize = size;
296         return (0);
297 }
298
299 /*
300  * Grow the stack to include sp.  Return 1 if successful, 0 otherwise.
301  * This routine assumes that the stack grows downward.
302  */
303 int
304 grow(caddr_t sp)
305 {
306         struct proc *p = curproc;
307         struct as *as = p->p_as;
308         size_t oldsize = p->p_stksize;
309         size_t newsize;
310         int err;
311
312         /*
313          * Serialize grow operations on an address space.
314          * This also serves as the lock protecting p_stksize
315          * and p_stkpageszc.
316          */
317         as_rangelock(as);
318         if (use_stk_lpg && (p->p_flag & SAUTOLPG) != 0) {
319                 err = grow_lpg(sp);
320         } else {
```



```
321                 err = grow_internal(sp, p->p_stkpageszc);
322         }
323         as_rangeunlock(as);
324
325         if (err == 0 && (newsize = p->p_stksize) > oldsize) {
326                 ASSERT(IS_P2ALIGNED(oldsize, PAGESIZE));
327                 ASSERT(IS_P2ALIGNED(newsize, PAGESIZE));
328                 /*
329                  * Set up translations so the process doesn't have to fault in
330                  * the stack pages we just gave it.
331                  */
332                 (void) as_fault(as->a_hat, as, p->p_usrstack - newsize,
333                     newsize - oldsize, F_INVAL, S_WRITE);
334         }
335         return ((err == 0 ? 1 : 0));
336 }
337
338 /*
339  * Algorithm: call arch-specific map_pgsz to get best page size to use,
340  * then call grow_internal().
341  * Returns 0 on success.
342  */
343 static int
344 grow_lpg(caddr_t sp)
345 {
346         struct proc *p = curproc;
347         size_t pgsz;
348         size_t len, newsize;
349         caddr_t addr, saddr;
350         caddr_t growend;
351         int oszc, szc;
352         int err;
353
354         newsize = p->p_usrstack - sp;
355
356         oszc = p->p_stkpageszc;
357         pgsz = map_pgsz(MAPPGSZ_STK, p, sp, newsize, 0);
358         szc = page_szc(pgsz);
359
360         /*
361          * Covers two cases:
362          * 1. page_szc() returns -1 for invalid page size, so we want to
363          * ignore it in that case.
364          * 2. By design we never decrease page size, as it is more stable.
365          * This shouldn't happen as the stack never shrinks.
366          */
367         if (szc <= oszc) {
368                 err = grow_internal(sp, oszc);
369                 /* failed, fall back to base page size */
370                 if (err != 0 && oszc != 0) {
371                         err = grow_internal(sp, 0);
372                 }
373                 return (err);
374         }
375
376         /*
377          * We've grown sufficiently to switch to a new page size.
378          * So we are going to remap the whole segment with the new page size.
379          */
380         err = grow_internal(sp, szc);
381         /* The grow with szc failed, so fall back to base page size. */
382         if (err != 0) {
383                 if (szc != 0) {
384                         err = grow_internal(sp, 0);
385                 }
386                 return (err);
```

```
387         }
389         /*
390          * Round up stack pointer to a large page boundary and remap
391          * any pgsz pages in the segment already faulted in beyond that
392          * point.
393          */
394         saddr = p->p_usrstack - p->p_stksize;
395         addr = (caddr_t)P2ROUNDUP((uintptr_t)saddr, pgsz);
396         growend = (caddr_t)P2ALIGN((uintptr_t)p->p_usrstack, pgsz);
397         len = growend - addr;
398         /* Check that len is not negative. Update page size code for stack. */
399         if (addr >= saddr && growend > addr && IS_P2ALIGNED(len, pgsz)) {
400                 (void) as_setpagesize(p->p_as, addr, len, szc, B_FALSE);
401                 p->p_stkpageszc = szc;
402         }
404         ASSERT(err == 0);
405         return (err);           /* should always be 0 */
406 }
408 /*
409  * This routine assumes that the stack grows downward.
410  * Returns 0 on success, errno on failure.
411  */
412 int
413 grow_internal(caddr_t sp, uint_t growszc)
414 {
415         struct proc *p = curproc;
416         size_t newsize;
417         size_t oldsize;
418         int    error;
419         size_t pgsz;
420         uint_t szc;
421         struct segvn_crargs crargs = SEGVN_ZFOD_ARGS(PROT_ZFOD, PROT_ALL);
423         ASSERT(sp < p->p_usrstack);
424         sp = (caddr_t)P2ALIGN((uintptr_t)sp, PAGESIZE);
426         /*
427          * grow to growszc alignment but use current p->p_stkpageszc for
428          * the segvn_crargs szc passed to segvn_create. For memcntl to
429          * increase the szc, this allows the new extension segment to be
430          * concatenated successfully with the existing stack segment.
431          */
432         if ((szc = growszc) != 0) {
433                 pgsz = page_get_pagesize(szc);
434                 ASSERT(pgsz > PAGESIZE);
435                 newsize = p->p_usrstack - (caddr_t)P2ALIGN((uintptr_t)sp, pgsz);
436                 if (newsize > (size_t)p->p_stk_ctl) {
437                         szc = 0;
438                         pgsz = PAGESIZE;
439                         newsize = p->p_usrstack - sp;
440                 }
441         } else {
442                 pgsz = PAGESIZE;
443                 newsize = p->p_usrstack - sp;
444         }
446         if (newsize > (size_t)p->p_stk_ctl) {
447                 (void) rctl_action(rctlproc_legacy[RLIMIT_STACK], p->p_rctls, p,
448                     RCA_UNSAFE_ALL);
450                 return (ENOMEM);
451         }
```



```
453         oldsize = p->p_stksize;
454         ASSERT(P2PHASE(oldsize, PAGESIZE) == 0);
456         if (newsize <= oldsize) {       /* prevent the stack from shrinking */
457                 return (0);
458         }
460         if (!(p->p_stkprot & PROT_EXEC)) {
461                 crargs.prot &= ~PROT_EXEC;
462         }
463         /*
464          * extend stack with the proposed new growszc, which is different
465          * than p_stkpageszc only on a memcntl to increase the stack pagesize.
466          * AS_MAP_NO_LPOOB means use 0, and don't reapply OOB policies via
467          * map_pgszcvec(). Use AS_MAP_STACK to get intermediate page sizes
468          * if not aligned to szc's pgsz.
469          */
470         if (szc > 0) {
471                 caddr_t oldsp = p->p_usrstack - oldsize;
472                 caddr_t austk = (caddr_t)P2ALIGN((uintptr_t)p->p_usrstack,
473                     pgsz);
475                 if (IS_P2ALIGNED(p->p_usrstack, pgsz) || oldsp < austk) {
476                         crargs.szc = p->p_stkpageszc ? p->p_stkpageszc :
477                             AS_MAP_NO_LPOOB;
478                 } else if (oldsp == austk) {
479                         crargs.szc = szc;
480                 } else {
481                         crargs.szc = AS_MAP_STACK;
482                 }
483         } else {
484                 crargs.szc = AS_MAP_NO_LPOOB;
485         }
486         crargs.lgrp_mem_policy_flags = LGRP_MP_FLAG_EXTEND_DOWN;
488         if ((error = as_map(p->p_as, p->p_usrstack - newsize, newsize - oldsize,
489             segvn_create, &crargs)) != 0) {
490                 if (error == EAGAIN) {
491                         cmn_err(CE_WARN, "Sorry, no swap space to grow stack "
492                             "for pid %d (%s)", p->p_pid, PTOU(p)->u_comm);
493                 }
494                 return (error);
495         }
496         p->p_stksize = newsize;
497         return (0);
498 }
500 /*
501  * Find address for user to map.  If MAP_FIXED is not specified, we can pick
502  * any address we want, but we will first try the value in *addrp if it is
503  * non-NULL and _MAP_RANDOMIZE is not set.  Thus this is implementing a way to
504  * try and get a preferred address.
 75  * Find address for user to map.
 76  * If MAP_FIXED is not specified, we can pick any address we want, but we will
 77  * first try the value in *addrp if it is non-NULL.  Thus this is implementing
 78  * a way to try and get a preferred address.
505  */
506 int
507 choose_addr(struct as *as, caddr_t *addrp, size_t len, offset_t off,
508     int vacalign, uint_t flags)
509 {
510         caddr_t basep = (caddr_t)(uintptr_t)((uintptr_t)*addrp & PAGEMASK);
511         size_t lenp = len;
513         ASSERT(AS_ISCLAIMGAP(as));      /* searches should be serialized */
514         if (flags & MAP_FIXED) {
```

```
515                 (void) as_unmap(as, *addrp, len);
516                 return (0);
517         } else if (basep != NULL &&
518             ((flags & (MAP_ALIGN | _MAP_RANDOMIZE)) == 0) &&
 91         } else if (basep != NULL && ((flags & MAP_ALIGN) == 0) &&
519             !as_gap(as, len, &basep, &lenp, 0, *addrp)) {
520                 /* User supplied address was available */
521                 *addrp = basep;
522         } else {
523                 /*
524                  * No user supplied address or the address supplied was not
525                  * available.
526                  */
527                 map_addr(addrp, len, off, vacalign, flags);
528         }
529         if (*addrp == NULL)
530                 return (ENOMEM);
531         return (0);
532 }
_____unchanged_portion_omitted_

599 static int
600 smmap_common(caddr_t *addrp, size_t len,
601     int prot, int flags, struct file *fp, offset_t pos)
602 {
603         struct vnode *vp;
604         struct as *as = curproc->p_as;
605         uint_t uprot, maxprot, type;
606         int error;
607         int in_crit = 0;

609         if ((flags & ~(MAP_SHARED | MAP_PRIVATE | MAP_FIXED | _MAP_NEW |
610             _MAP_LOW32 | MAP_NORESERVE | MAP_ANON | MAP_ALIGN |
611             MAP_TEXT | MAP_INITDATA)) != 0) {
612                 /* | MAP_RENAME */      /* not implemented, let user know */
613                 return (EINVAL);
614         }

616         if ((flags & MAP_TEXT) && !(prot & PROT_EXEC)) {
617                 return (EINVAL);
618         }

620         if ((flags & (MAP_TEXT | MAP_INITDATA)) == (MAP_TEXT | MAP_INITDATA)) {
621                 return (EINVAL);
622         }

624         if ((flags & (MAP_FIXED | _MAP_RANDOMIZE)) == (MAP_FIXED | _MAP_RANDOMIZ
625                 return (EINVAL);
626         }

628         /* If it's not a fixed allocation and mmap ASLR is enabled, randomize it
629         if (((flags & MAP_FIXED) == 0) &&
630             secflag_enabled(curproc, PROC_SEC_ASLR))
631                 flags |= _MAP_RANDOMIZE;

633 #endif /* ! codereview */
634 #if defined(__sparc)
635         /*
636          * See if this is an "old mmap call".  If so, remember this
637          * fact and convert the flags value given to mmap to indicate
638          * the specified address in the system call must be used.
639          * _MAP_NEW is turned set by all new uses of mmap.
640          */
641         if ((flags & _MAP_NEW) == 0)
642                 flags |= MAP_FIXED;
643 #endif
```



```
644         flags &= ~_MAP_NEW;

646         type = flags & MAP_TYPE;
647         if (type != MAP_PRIVATE && type != MAP_SHARED)
648                 return (EINVAL);


651         if (flags & MAP_ALIGN) {

652                 if (flags & MAP_FIXED)
653                         return (EINVAL);

655                 /* alignment needs to be a power of 2 >= page size */
656                 if (((uintptr_t)*addrp < PAGESIZE && (uintptr_t)*addrp != 0) ||
657                     !ISP2((uintptr_t)*addrp))
658                         return (EINVAL);
659         }
660         /*
661          * Check for bad lengths and file position.
662          * We let the VOP_MAP routine check for negative lengths
663          * since on some vnode types this might be appropriate.
664          */
665         if (len == 0 || (pos & (u_offset_t)PAGEOFFSET) != 0)
666                 return (EINVAL);

668         maxprot = PROT_ALL;             /* start out allowing all accesses */
669         uprot = prot | PROT_USER;

671         if (fp == NULL) {
672                 ASSERT(flags & MAP_ANON);
673                 /* discard lwpchan mappings, like munmap() */
674                 if ((flags & MAP_FIXED) && curproc->p_lcp != NULL)
675                         lwpchan_delete_mapping(curproc, *addrp, *addrp + len);
676                 as_rangelock(as);
677                 error = zmap(as, addrp, len, uprot, flags, pos);
678                 as_rangeunlock(as);
679                 /*
680                  * Tell machine specific code that lwp has mapped shared memory
681                  */
682                 if (error == 0 && (flags & MAP_SHARED)) {
683                         /* EMPTY */
684                         LWP_MMODEL_SHARED_AS(*addrp, len);
685                 }
686                 return (error);
687         } else if ((flags & MAP_ANON) != 0)
688                 return (EINVAL);

690         vp = fp->f_vnode;

692         /* Can't execute code from "noexec" mounted filesystem. */
693         if ((vp->v_vfsp->vfs_flag & VFS_NOEXEC) != 0)
694                 maxprot &= ~PROT_EXEC;

696         /*
697          * These checks were added as part of large files.
698          *
699          * Return ENXIO if the initial position is negative; return EOVERFLOW
700          * if (offset + len) would overflow the maximum allowed offset for the
701          * type of file descriptor being used.
702          */
703         if (vp->v_type == VREG) {
704                 if (pos < 0)
705                         return (ENXIO);
706                 if ((offset_t)len > (OFFSET_MAX(fp) - pos))
707                         return (EOVERFLOW);
708         }
```

```
710         if (type == MAP_SHARED && (fp->f_flag & FWRITE) == 0) {
711                 /* no write access allowed */
712                 maxprot &= ~PROT_WRITE;
713         }

715         /*
716          * XXX - Do we also adjust maxprot based on protections
717          * of the vnode?  E.g. if no execute permission is given
718          * on the vnode for the current user, maxprot probably
719          * should disallow PROT_EXEC also?  This is different
720          * from the write access as this would be a per vnode
721          * test as opposed to a per fd test for writability.
722          */

724         /*
725          * Verify that the specified protections are not greater than
726          * the maximum allowable protections.  Also test to make sure
727          * that the file descriptor does allows for read access since
728          * "write only" mappings are hard to do since normally we do
729          * the read from the file before the page can be written.
730          */
731         if (((maxprot & uprot) != uprot) || (fp->f_flag & FREAD) == 0)
732                 return (EACCES);

734         /*
735          * If the user specified an address, do some simple checks here
736          */
737         if ((flags & MAP_FIXED) != 0) {
738                 caddr_t userlimit;

740                 /*
741                  * Use the user address.  First verify that
742                  * the address to be used is page aligned.
743                  * Then make some simple bounds checks.
744                  */
745                 if (((uintptr_t)*addrp & PAGEOFFSET) != 0)
746                         return (EINVAL);

748                 userlimit = flags & _MAP_LOW32 ?
749                     (caddr_t)USERLIMIT32 : as->a_userlimit;
750                 switch (valid_usr_range(*addrp, len, uprot, as, userlimit)) {
751                 case RANGE_OKAY:
752                         break;
753                 case RANGE_BADPROT:
754                         return (ENOTSUP);
755                 case RANGE_BADADDR:
756                 default:
757                         return (ENOMEM);
758                 }
759         }

761         if ((prot & (PROT_READ | PROT_WRITE | PROT_EXEC)) &&
762             nbl_need_check(vp)) {
763                 int svmand;
764                 nbl_op_t nop;

766                 nbl_start_crit(vp, RW_READER);
767                 in_crit = 1;
768                 error = nbl_svmand(vp, fp->f_cred, &svmand);
769                 if (error != 0)
770                         goto done;
771                 if ((prot & PROT_WRITE) && (type == MAP_SHARED)) {
772                         if (prot & (PROT_READ | PROT_EXEC)) {
773                                 nop = NBL_READWRITE;
774                         } else {
```



```
775                                 nop = NBL_WRITE;
776                         }
777                 } else {
778                         nop = NBL_READ;
779                 }
780                 if (nbl_conflict(vp, nop, 0, LONG_MAX, svmand, NULL)) {
781                         error = EACCES;
782                         goto done;
783                 }
784         }

786         /* discard lwpchan mappings, like munmap() */
787         if ((flags & MAP_FIXED) && curproc->p_lcp != NULL)
788                 lwpchan_delete_mapping(curproc, *addrp, *addrp + len);

790         /*
791          * Ok, now let the vnode map routine do its thing to set things up.
792          */
793         error = VOP_MAP(vp, pos, as,
794             addrp, len, uprot, maxprot, flags, fp->f_cred, NULL);

796         if (error == 0) {
797                 /*
798                  * Tell machine specific code that lwp has mapped shared memory
799                  */
800                 if (flags & MAP_SHARED) {
801                         /* EMPTY */
802                         LWP_MMODEL_SHARED_AS(*addrp, len);
803                 }
804                 if (vp->v_type == VREG &&
805                     (flags & (MAP_TEXT | MAP_INITDATA)) != 0) {
806                         /*
807                          * Mark this as an executable vnode
808                          */
809                         mutex_enter(&vp->v_lock);
810                         vp->v_flag |= VVMEXEC;
811                         mutex_exit(&vp->v_lock);
812                 }
813         }

815 done:
816         if (in_crit)
817                 nbl_end_crit(vp);
818         return (error);
819 }
```

*unchanged_portion_omitted_*

```
**********************************************************
   69615 Wed May 27 19:49:29 2015
new/usr/src/uts/common/os/mmapobj.c
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
```

```
   1 /*
   2  * CDDL HEADER START
   3  *
   4  * The contents of this file are subject to the terms of the
   5  * Common Development and Distribution License (the "License").
   6  * You may not use this file except in compliance with the License.
   7  *
   8  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
   9  * or http://www.opensolaris.org/os/licensing.
  10  * See the License for the specific language governing permissions
  11  * and limitations under the License.
  12  *
  13  * When distributing Covered Code, include this CDDL HEADER in each
  14  * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  15  * If applicable, add the following below this CDDL HEADER, with the
  16  * fields enclosed by brackets "[]" replaced with your own identifying
  17  * information: Portions Copyright [yyyy] [name of copyright owner]
  18  *
  19  * CDDL HEADER END
  20  */
  21 /*
  22  * Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
  23  * Use is subject to license terms.
  24  * Copyright 2014 Joyent, Inc.  All rights reserved.
  25  */

  27 #include <sys/types.h>
  28 #include <sys/sysmacros.h>
  29 #include <sys/kmem.h>
  30 #include <sys/param.h>
  31 #include <sys/systm.h>
  32 #include <sys/errno.h>
  33 #include <sys/mman.h>
  34 #include <sys/cmn_err.h>
  35 #include <sys/cred.h>
  36 #include <sys/vmsystm.h>
  37 #include <sys/machsystm.h>
  38 #include <sys/debug.h>
  39 #include <vm/as.h>
  40 #include <vm/seg.h>
  41 #include <sys/vmparam.h>
  42 #include <sys/vfs.h>
  43 #include <sys/elf.h>
  44 #include <sys/machelf.h>
  45 #include <sys/corectl.h>
  46 #include <sys/exec.h>
  47 #include <sys/exechdr.h>
  48 #include <sys/autoconf.h>
  49 #include <sys/mem.h>
  50 #include <vm/seg_dev.h>
  51 #include <sys/vmparam.h>
  52 #include <sys/mmapobj.h>
  53 #include <sys/atomic.h>

  55 /*
  56  * Theory statement:
  57  *
  58  * The main driving force behind mmapobj is to interpret and map ELF files
```



```
  59  * inside of the kernel instead of having the linker be responsible for this.
  60  *
  61  * mmapobj also supports the AOUT 4.x binary format as well as flat files in
  62  * a read only manner.
  63  *
  64  * When interpreting and mapping an ELF file, mmapobj will map each PT_LOAD
  65  * or PT_SUNWBSS segment according to the ELF standard.  Refer to the "Linker
  66  * and Libraries Guide" for more information about the standard and mapping
  67  * rules.
  68  *
  69  * Having mmapobj interpret and map objects will allow the kernel to make the
  70  * best decision for where to place the mappings for said objects.  Thus, we
  71  * can make optimizations inside of the kernel for specific platforms or cache
  72  * mapping information to make mapping objects faster.  The cache is ignored
  73  * if ASLR is enabled.
  71  * can make optimizations inside of the kernel for specific platforms or
  72  * cache mapping information to make mapping objects faster.
  74  *
  75  * The lib_va_hash will be one such optimization.  For each ELF object that
  76  * mmapobj is asked to interpret, we will attempt to cache the information
  77  * about the PT_LOAD and PT_SUNWBSS sections to speed up future mappings of
  78  * the same objects.  We will cache up to LIBVA_CACHED_SEGS (see below) program
  79  * headers which should cover a majority of the libraries out there without
  80  * wasting space.  In order to make sure that the cached information is valid,
  81  * we check the passed in vnode's mtime and ctime to make sure the vnode
  82  * has not been modified since the last time we used it.
  83  *
  84  * In addition, the lib_va_hash may contain a preferred starting VA for the
  85  * object which can be useful for platforms which support a shared context.
  86  * This will increase the likelyhood that library text can be shared among
  87  * many different processes.  We limit the reserved VA space for 32 bit objects
  88  * in order to minimize fragmenting the processes address space.
  89  *
  90  * In addition to the above, the mmapobj interface allows for padding to be
  91  * requested before the first mapping and after the last mapping created.
  92  * When padding is requested, no additional optimizations will be made for
  93  * that request.
  94  */

  96 /*
  97  * Threshold to prevent allocating too much kernel memory to read in the
  98  * program headers for an object.  If it requires more than below,
  99  * we will use a KM_NOSLEEP allocation to allocate memory to hold all of the
 100  * program headers which could possibly fail.  If less memory than below is
 101  * needed, then we use a KM_SLEEP allocation and are willing to wait for the
 102  * memory if we need to.
 103  */
 104 size_t mmapobj_alloc_threshold = 65536;

 106 /* Debug stats for test coverage */
 107 #ifdef DEBUG
 108 struct mobj_stats {
 109         uint_t  mobjs_unmap_called;
 110         uint_t  mobjs_remap_devnull;
 111         uint_t  mobjs_lookup_start;
 112         uint_t  mobjs_alloc_start;
 113         uint_t  mobjs_alloc_vmem;
 114         uint_t  mobjs_add_collision;
 115         uint_t  mobjs_get_addr;
 116         uint_t  mobjs_map_flat_no_padding;
 117         uint_t  mobjs_map_flat_padding;
 118         uint_t  mobjs_map_ptload_text;
 119         uint_t  mobjs_map_ptload_initdata;
 120         uint_t  mobjs_map_ptload_preread;
 121         uint_t  mobjs_map_ptload_unaligned_text;
 122         uint_t  mobjs_map_ptload_unaligned_map_fail;
```

```
123         uint_t  mobjs_map_ptload_unaligned_read_fail;
124         uint_t  mobjs_zfoddiff;
125         uint_t  mobjs_zfoddiff_nowrite;
126         uint_t  mobjs_zfodextra;
127         uint_t  mobjs_ptload_failed;
128         uint_t  mobjs_map_elf_no_holes;
129         uint_t  mobjs_unmap_hole;
130         uint_t  mobjs_nomem_header;
131         uint_t  mobjs_inval_header;
132         uint_t  mobjs_overlap_header;
133         uint_t  mobjs_np2_align;
134         uint_t  mobjs_np2_align_overflow;
135         uint_t  mobjs_exec_padding;
136         uint_t  mobjs_exec_addr_mapped;
137         uint_t  mobjs_exec_addr_devnull;
138         uint_t  mobjs_exec_addr_in_use;
139         uint_t  mobjs_lvp_found;
140         uint_t  mobjs_no_loadable_yet;
141         uint_t  mobjs_nothing_to_map;
142         uint_t  mobjs_e2big;
143         uint_t  mobjs_dyn_pad_align;
144         uint_t  mobjs_dyn_pad_noalign;
145         uint_t  mobjs_alloc_start_fail;
146         uint_t  mobjs_lvp_nocache;
147         uint_t  mobjs_extra_padding;
148         uint_t  mobjs_lvp_not_needed;
149         uint_t  mobjs_no_mem_map_sz;
150         uint_t  mobjs_check_exec_failed;
151         uint_t  mobjs_lvp_used;
152         uint_t  mobjs_wrong_model;
153         uint_t  mobjs_noexec_fs;
154         uint_t  mobjs_e2big_et_rel;
155         uint_t  mobjs_et_rel_mapped;
156         uint_t  mobjs_unknown_elf_type;
157         uint_t  mobjs_phent32_too_small;
158         uint_t  mobjs_phent64_too_small;
159         uint_t  mobjs_inval_elf_class;
160         uint_t  mobjs_too_many_phdrs;
161         uint_t  mobjs_no_phsize;
162         uint_t  mobjs_phsize_large;
163         uint_t  mobjs_phsize_xtralarge;
164         uint_t  mobjs_fast_wrong_model;
165         uint_t  mobjs_fast_e2big;
166         uint_t  mobjs_fast;
167         uint_t  mobjs_fast_success;
168         uint_t  mobjs_fast_not_now;
169         uint_t  mobjs_small_file;
170         uint_t  mobjs_read_error;
171         uint_t  mobjs_unsupported;
172         uint_t  mobjs_flat_e2big;
173         uint_t  mobjs_phent_align32;
174         uint_t  mobjs_phent_align64;
175         uint_t  mobjs_lib_va_find_hit;
176         uint_t  mobjs_lib_va_find_delay_delete;
177         uint_t  mobjs_lib_va_find_delete;
178         uint_t  mobjs_lib_va_add_delay_delete;
179         uint_t  mobjs_lib_va_add_delete;
180         uint_t  mobjs_lib_va_create_failure;
181         uint_t  mobjs_min_align;
182 #if defined(__sparc)
183         uint_t  mobjs_aout_uzero_fault;
184         uint_t  mobjs_aout_64bit_try;
185         uint_t  mobjs_aout_noexec;
186         uint_t  mobjs_aout_e2big;
187         uint_t  mobjs_aout_lib;
188         uint_t  mobjs_aout_fixed;
```



```
189         uint_t  mobjs_aout_zfoddiff;
190         uint_t  mobjs_aout_map_bss;
191         uint_t  mobjs_aout_bss_fail;
192         uint_t  mobjs_aout_nlist;
193         uint_t  mobjs_aout_addr_in_use;
194 #endif
195 } mobj_stats;
_____unchanged_portion_omitted_

704 /*
705  * Get the starting address for a given file to be mapped and return it
706  * to the caller.  If we're using lib_va and we need to allocate an address,
707  * we will attempt to allocate it from the global reserved pool such that the
708  * same address can be used in the future for this file.  If we can't use the
709  * reserved address then we just get one that will fit in our address space.
710  *
711  * Returns the starting virtual address for the range to be mapped or NULL
712  * if an error is encountered. If we successfully insert the requested info
713  * into the lib_va hash, then *lvpp will be set to point to this lib_va
714  * structure.  The structure will have a hold on it and thus lib_va_release
715  * needs to be called on it by the caller.  This function will not fill out
716  * lv_mps or lv_num_segs since it does not have enough information to do so.
717  * The caller is responsible for doing this making sure that any modifications
718  * to lv_mps are visible before setting lv_num_segs.
719  */
720 static caddr_t
721 mmapobj_alloc_start_addr(struct lib_va **lvpp, size_t len, int use_lib_va,
722     int randomize, size_t align, vattr_t *vap)
721     size_t align, vattr_t *vap)
723 {
724         proc_t *p = curproc;
725         struct as *as = p->p_as;
726         struct segvn_crargs crargs = SEGVN_ZFOD_ARGS(PROT_USER, PROT_ALL);
727         int error;
728         model_t model;
729         uint_t ma_flags = _MAP_LOW32;
730         caddr_t base = NULL;
731         vmem_t *model_vmem;
732         size_t lib_va_start;
733         size_t lib_va_end;
734         size_t lib_va_len;

736         ASSERT(lvpp != NULL);
737         ASSERT((randomize & use_lib_va) != 1);
738 #endif /* ! codereview */

740         MOBJ_STAT_ADD(alloc_start);
741         model = get_udatamodel();

743         if (model == DATAMODEL_LP64) {
744                 ma_flags = 0;
745                 model_vmem = lib_va_64_arena;
746         } else {
747                 ASSERT(model == DATAMODEL_ILP32);
748                 model_vmem = lib_va_32_arena;
749         }

751         if (align > 1) {
752                 ma_flags |= MAP_ALIGN;
753         }

755         if (randomize != 0)
756                 ma_flags |= _MAP_RANDOMIZE;

758 #endif /* ! codereview */
759         if (use_lib_va) {
```

```
760                 /*
761                  * The first time through, we need to setup the lib_va arenas.
762                  * We call map_addr to find a suitable range of memory to map
763                  * the given library, and we will set the highest address
764                  * in our vmem arena to the end of this adddress range.
765                  * We allow up to half of the address space to be used
766                  * for lib_va addresses but we do not prevent any allocations
767                  * in this range from other allocation paths.
768                  */
769                 if (lib_va_64_arena == NULL && model == DATAMODEL_LP64) {
770                         mutex_enter(&lib_va_init_mutex);
771                         if (lib_va_64_arena == NULL) {
772                                 base = (caddr_t)align;
773                                 as_rangelock(as);
774                                 map_addr(&base, len, 0, 1, ma_flags);
775                                 as_rangeunlock(as);
776                                 if (base == NULL) {
777                                         mutex_exit(&lib_va_init_mutex);
778                                         MOBJ_STAT_ADD(lib_va_create_failure);
779                                         goto nolibva;
780                                 }
781                                 lib_va_end = (size_t)base + len;
782                                 lib_va_len = lib_va_end >> 1;
783                                 lib_va_len = P2ROUNDUP(lib_va_len, PAGESIZE);
784                                 lib_va_start = lib_va_end - lib_va_len;
785
786                                 /*
787                                  * Need to make sure we avoid the address hole.
788                                  * We know lib_va_end is valid but we need to
789                                  * make sure lib_va_start is as well.
790                                  */
791                                 if ((lib_va_end > (size_t)hole_end) &&
792                                     (lib_va_start < (size_t)hole_end)) {
793                                         lib_va_start = P2ROUNDUP(
794                                             (size_t)hole_end, PAGESIZE);
795                                         lib_va_len = lib_va_end - lib_va_start;
796                                 }
797                                 lib_va_64_arena = vmem_create("lib_va_64",
798                                     (void *)lib_va_start, lib_va_len, PAGESIZE,
799                                     NULL, NULL, NULL, 0,
800                                     VM_NOSLEEP | VMC_IDENTIFIER);
801                                 if (lib_va_64_arena == NULL) {
802                                         mutex_exit(&lib_va_init_mutex);
803                                         goto nolibva;
804                                 }
805                         }
806                         model_vmem = lib_va_64_arena;
807                         mutex_exit(&lib_va_init_mutex);
808                 } else if (lib_va_32_arena == NULL &&
809                     model == DATAMODEL_ILP32) {
810                         mutex_enter(&lib_va_init_mutex);
811                         if (lib_va_32_arena == NULL) {
812                                 base = (caddr_t)align;
813                                 as_rangelock(as);
814                                 map_addr(&base, len, 0, 1, ma_flags);
815                                 as_rangeunlock(as);
816                                 if (base == NULL) {
817                                         mutex_exit(&lib_va_init_mutex);
818                                         MOBJ_STAT_ADD(lib_va_create_failure);
819                                         goto nolibva;
820                                 }
821                                 lib_va_end = (size_t)base + len;
822                                 lib_va_len = lib_va_end >> 1;
823                                 lib_va_len = P2ROUNDUP(lib_va_len, PAGESIZE);
824                                 lib_va_start = lib_va_end - lib_va_len;
825                                 lib_va_32_arena = vmem_create("lib_va_32",
```



```
826                                     (void *)lib_va_start, lib_va_len, PAGESIZE,
827                                     NULL, NULL, NULL, 0,
828                                     VM_NOSLEEP | VMC_IDENTIFIER);
829                                 if (lib_va_32_arena == NULL) {
830                                         mutex_exit(&lib_va_init_mutex);
831                                         goto nolibva;
832                                 }
833                         }
834                         model_vmem = lib_va_32_arena;
835                         mutex_exit(&lib_va_init_mutex);
836                 }
837
838                 if (model == DATAMODEL_LP64 || libs_mapped_32 < lib_threshold) {
839                         base = vmem_xalloc(model_vmem, len, align, 0, 0, NULL,
840                             NULL, VM_NOSLEEP | VM_ENDALLOC);
841                         MOBJ_STAT_ADD(alloc_vmem);
842                 }
843
844                 /*
845                  * Even if the address fails to fit in our address space,
846                  * or we can't use a reserved address,
847                  * we should still save it off in lib_va_hash.
848                  */
849                 *lvpp = lib_va_add_hash(base, len, align, vap);
850
851                 /*
852                  * Check for collision on insertion and free up our VA space.
853                  * This is expected to be rare, so we'll just reset base to
854                  * NULL instead of looking it up in the lib_va hash.
855                  */
856                 if (*lvpp == NULL) {
857                         if (base != NULL) {
858                                 vmem_xfree(model_vmem, base, len);
859                                 base = NULL;
860                                 MOBJ_STAT_ADD(add_collision);
861                         }
862                 }
863         }
864
865 nolibva:
866         as_rangelock(as);
867
868         /*
869          * If we don't have an expected base address, or the one that we want
870          * to use is not available or acceptable, go get an acceptable
871          * address range.
872          *
873          * If ASLR is enabled, we should never have used the cache, and should
874          * also start our real work here, in the consequent of the next
875          * condition.
876 #endif /* ! codereview */
877          */
878         if (randomize != 0)
879                 ASSERT(base == NULL);
880
881 #endif /* ! codereview */
882         if (base == NULL || as_gap(as, len, &base, &len, 0, NULL) ||
883             valid_usr_range(base, len, PROT_ALL, as, as->a_userlimit) !=
884             RANGE_OKAY || OVERLAPS_STACK(base + len, p)) {
885                 MOBJ_STAT_ADD(get_addr);
886                 base = (caddr_t)align;
887                 map_addr(&base, len, 0, 1, ma_flags);
888         }
889
890         /*
891          * Need to reserve the address space we're going to use.
```

```
892          * Don't reserve swap space since we'll be mapping over this.
893          */
894         if (base != NULL) {
895                 /* Don't reserve swap space since we'll be mapping over this */
896                 crargs.flags |= MAP_NORESERVE;
897                 error = as_map(as, base, len, segvn_create, &crargs);
898                 if (error) {
899                         base = NULL;
900                 }
901         }
902
903         as_rangeunlock(as);
904         return (base);
905 }
906
907 /*
908  * Map the file associated with vp into the address space as a single
909  * read only private mapping.
910  * Returns 0 for success, and non-zero for failure to map the file.
911  */
912 static int
913 mmapobj_map_flat(vnode_t *vp, mmapobj_result_t *mrp, size_t padding,
914     cred_t *fcred)
915 {
916         int error = 0;
917         struct as *as = curproc->p_as;
918         caddr_t addr = NULL;
919         caddr_t start_addr;
920         size_t len;
921         size_t pad_len;
922         int prot = PROT_USER | PROT_READ;
923         uint_t ma_flags = _MAP_LOW32;
924         vattr_t vattr;
925         struct segvn_crargs crargs = SEGVN_ZFOD_ARGS(PROT_USER, PROT_ALL);
926
927         if (get_udatamodel() == DATAMODEL_LP64) {
928                 ma_flags = 0;
929         }
930
931         vattr.va_mask = AT_SIZE;
932         error = VOP_GETATTR(vp, &vattr, 0, fcred, NULL);
933         if (error) {
934                 return (error);
935         }
936
937         len = vattr.va_size;
938
939         ma_flags |= MAP_PRIVATE;
940         if (padding == 0) {
941                 MOBJ_STAT_ADD(map_flat_no_padding);
942                 error = VOP_MAP(vp, 0, as, &addr, len, prot, PROT_ALL,
943                     ma_flags, fcred, NULL);
944                 if (error == 0) {
945                         mrp[0].mr_addr = addr;
946                         mrp[0].mr_msize = len;
947                         mrp[0].mr_fsize = len;
948                         mrp[0].mr_offset = 0;
949                         mrp[0].mr_prot = prot;
950                         mrp[0].mr_flags = 0;
951                 }
952                 return (error);
953         }
954
955         /* padding was requested so there's more work to be done */
956         MOBJ_STAT_ADD(map_flat_padding);
```



```
958         /* No need to reserve swap space now since it will be reserved later */
959         crargs.flags |= MAP_NORESERVE;
960
961         /* Need to setup padding which can only be in PAGESIZE increments. */
962         ASSERT((padding & PAGEOFFSET) == 0);
963         pad_len = len + (2 * padding);
964
965         as_rangelock(as);
966         map_addr(&addr, pad_len, 0, 1, ma_flags);
967         error = as_map(as, addr, pad_len, segvn_create, &crargs);
968         as_rangeunlock(as);
969         if (error) {
970                 return (error);
971         }
972         start_addr = addr;
973         addr += padding;
974         ma_flags |= MAP_FIXED;
975         error = VOP_MAP(vp, 0, as, &addr, len, prot, PROT_ALL, ma_flags,
976             fcred, NULL);
977         if (error == 0) {
978                 mrp[0].mr_addr = start_addr;
979                 mrp[0].mr_msize = padding;
980                 mrp[0].mr_fsize = 0;
981                 mrp[0].mr_offset = 0;
982                 mrp[0].mr_prot = 0;
983                 mrp[0].mr_flags = MR_PADDING;
984
985                 mrp[1].mr_addr = addr;
986                 mrp[1].mr_msize = len;
987                 mrp[1].mr_fsize = len;
988                 mrp[1].mr_offset = 0;
989                 mrp[1].mr_prot = prot;
990                 mrp[1].mr_flags = 0;
991
992                 mrp[2].mr_addr = addr + P2ROUNDUP(len, PAGESIZE);
993                 mrp[2].mr_msize = padding;
994                 mrp[2].mr_fsize = 0;
995                 mrp[2].mr_offset = 0;
996                 mrp[2].mr_prot = 0;
997                 mrp[2].mr_flags = MR_PADDING;
998         } else {
999                 /* Need to cleanup the as_map from earlier */
1000                (void) as_unmap(as, start_addr, pad_len);
1001        }
1002        return (error);
1003 }
1004
1005 /*
1006  * Map a PT_LOAD or PT_SUNWBSS section of an executable file into the user's
1007  * address space.
1008  * vp - vnode to be mapped in
1009  * addr - start address
1010  * len - length of vp to be mapped
1011  * zfodlen - length of zero filled memory after len above
1012  * offset - offset into file where mapping should start
1013  * prot - protections for this mapping
1014  * fcred - credentials for the file associated with vp at open time.
1015  */
1016 static int
1017 mmapobj_map_ptload(struct vnode *vp, caddr_t addr, size_t len, size_t zfodlen,
1018     off_t offset, int prot, cred_t *fcred)
1019 {
1020         int error = 0;
1021         caddr_t zfodbase, oldaddr;
1022         size_t oldlen;
1023         size_t end;
```

```
1024         size_t zfoddiff;
1025         label_t ljb;
1026         struct as *as = curproc->p_as;
1027         model_t model;
1028         int full_page;

1030         /*
1031          * See if addr and offset are aligned such that we can map in
1032          * full pages instead of partial pages.
1033          */
1034         full_page = (((uintptr_t)addr & PAGEOFFSET) ==
1035             ((uintptr_t)offset & PAGEOFFSET));

1037         model = get_udatamodel();

1039         oldaddr = addr;
1040         addr = (caddr_t)((uintptr_t)addr & (uintptr_t)PAGEMASK);
1041         if (len) {
1042                 spgcnt_t availm, npages;
1043                 int preread;
1044                 uint_t mflag = MAP_PRIVATE | MAP_FIXED;

1046                 if (model == DATAMODEL_ILP32) {
1047                         mflag |= _MAP_LOW32;
1048                 }
1049                 /* We may need to map in extra bytes */
1050                 oldlen = len;
1051                 len += ((size_t)oldaddr & PAGEOFFSET);

1053                 if (full_page) {
1054                         offset = (off_t)((uintptr_t)offset & PAGEMASK);
1055                         if ((prot & (PROT_WRITE | PROT_EXEC)) == PROT_EXEC) {
1056                                 mflag |= MAP_TEXT;
1057                                 MOBJ_STAT_ADD(map_ptload_text);
1058                         } else {
1059                                 mflag |= MAP_INITDATA;
1060                                 MOBJ_STAT_ADD(map_ptload_initdata);
1061                         }

1063                         /*
1064                          * maxprot is passed as PROT_ALL so that mdb can
1065                          * write to this segment.
1066                          */
1067                         if (error = VOP_MAP(vp, (offset_t)offset, as, &addr,
1068                             len, prot, PROT_ALL, mflag, fcred, NULL)) {
1069                                 return (error);
1070                         }

1072                         /*
1073                          * If the segment can fit and is relatively small, then
1074                          * we prefault the entire segment in.  This is based
1075                          * on the model that says the best working set of a
1076                          * small program is all of its pages.
1077                          * We only do this if freemem will not drop below
1078                          * lotsfree since we don't want to induce paging.
1079                          */
1080                         npages = (spgcnt_t)btopr(len);
1081                         availm = freemem - lotsfree;
1082                         preread = (npages < availm && len < PGTHRESH) ? 1 : 0;

1084                         /*
1085                          * If we aren't prefaulting the segment,
1086                          * increment "deficit", if necessary to ensure
1087                          * that pages will become available when this
1088                          * process starts executing.
1089                          */
```



```
1090                         if (preread == 0 && npages > availm &&
1091                             deficit < lotsfree) {
1092                                 deficit += MIN((pgcnt_t)(npages - availm),
1093                                     lotsfree - deficit);
1094                         }

1096                         if (preread) {
1097                                 (void) as_faulta(as, addr, len);
1098                                 MOBJ_STAT_ADD(map_ptload_preread);
1099                         }
1100                 } else {
1101                         /*
1102                          * addr and offset were not aligned such that we could
1103                          * use VOP_MAP, thus we need to as_map the memory we
1104                          * need and then read the data in from disk.
1105                          * This code path is a corner case which should never
1106                          * be taken, but hand crafted binaries could trigger
1107                          * this logic and it needs to work correctly.
1108                          */
1109                         MOBJ_STAT_ADD(map_ptload_unaligned_text);
1110                         as_rangelock(as);
1111                         (void) as_unmap(as, addr, len);

1113                         /*
1114                          * We use zfod_argsp because we need to be able to
1115                          * write to the mapping and then we'll change the
1116                          * protections later if they are incorrect.
1117                          */
1118                         error = as_map(as, addr, len, segvn_create, zfod_argsp);
1119                         as_rangeunlock(as);
1120                         if (error) {
1121                                 MOBJ_STAT_ADD(map_ptload_unaligned_map_fail);
1122                                 return (error);
1123                         }

1125                         /* Now read in the data from disk */
1126                         error = vn_rdwr(UIO_READ, vp, oldaddr, oldlen, offset,
1127                             UIO_USERSPACE, 0, (rlim64_t)0, fcred, NULL);
1128                         if (error) {
1129                                 MOBJ_STAT_ADD(map_ptload_unaligned_read_fail);
1130                                 return (error);
1131                         }

1133                         /*
1134                          * Now set protections.
1135                          */
1136                         if (prot != PROT_ZFOD) {
1137                                 (void) as_setprot(as, addr, len, prot);
1138                         }
1139                 }
1140         }

1142         if (zfodlen) {
1143                 end = (size_t)addr + len;
1144                 zfodbase = (caddr_t)P2ROUNDUP(end, PAGESIZE);
1145                 zfoddiff = (uintptr_t)zfodbase - end;
1146                 if (zfoddiff) {
1147                         /*
1148                          * Before we go to zero the remaining space on the last
1149                          * page, make sure we have write permission.
1150                          *
1151                          * We need to be careful how we zero-fill the last page
1152                          * if the protection does not include PROT_WRITE. Using
1153                          * as_setprot() can cause the VM segment code to call
1154                          * segvn_vpage(), which must allocate a page struct for
1155                          * each page in the segment. If we have a very large
```

```
1156                         * segment, this may fail, so we check for that, even
1157                         * though we ignore other return values from as_setprot.
1158                         */
1159                        MOBJ_STAT_ADD(zfoddiff);
1160                        if ((prot & PROT_WRITE) == 0) {
1161                                if (as_setprot(as, (caddr_t)end, zfoddiff,
1162                                    prot | PROT_WRITE) == ENOMEM)
1163                                        return (ENOMEM);
1164                                MOBJ_STAT_ADD(zfoddiff_nowrite);
1165                        }
1166                        if (on_fault(&ljb)) {
1167                                no_fault();
1168                                if ((prot & PROT_WRITE) == 0) {
1169                                        (void) as_setprot(as, (caddr_t)end,
1170                                            zfoddiff, prot);
1171                                }
1172                                return (EFAULT);
1173                        }
1174                        uzero((void *)end, zfoddiff);
1175                        no_fault();

1177                        /*
1178                         * Remove write protection to return to original state
1179                         */
1180                        if ((prot & PROT_WRITE) == 0) {
1181                                (void) as_setprot(as, (caddr_t)end,
1182                                    zfoddiff, prot);
1183                        }
1184                }
1185                if (zfodlen > zfoddiff) {
1186                        struct segvn_crargs crargs =
1187                            SEGVN_ZFOD_ARGS(prot, PROT_ALL);

1189                        MOBJ_STAT_ADD(zfodextra);
1190                        zfodlen -= zfoddiff;
1191                        crargs.szc = AS_MAP_NO_LPOOB;


1194                        as_rangelock(as);
1195                        (void) as_unmap(as, (caddr_t)zfodbase, zfodlen);
1196                        error = as_map(as, (caddr_t)zfodbase,
1197                            zfodlen, segvn_create, &crargs);
1198                        as_rangeunlock(as);
1199                        if (error) {
1200                                return (error);
1201                        }
1202                }
1203        }
1204        return (0);
1205 }

1207 /*
1208  * Map the ELF file represented by vp into the users address space.  The
1209  * first mapping will start at start_addr and there will be num_elements
1210  * mappings.  The mappings are described by the data in mrp which may be
1211  * modified upon returning from this function.
1212  * Returns 0 for success or errno for failure.
1213  */
1214 static int
1215 mmapobj_map_elf(struct vnode *vp, caddr_t start_addr, mmapobj_result_t *mrp,
1216     int num_elements, cred_t *fcred, ushort_t e_type)
1217 {
1218        int i;
1219        int ret;
1220        caddr_t lo;
1221        caddr_t hi;
```



```
1222        struct as *as = curproc->p_as;

1224        for (i = 0; i < num_elements; i++) {
1225                caddr_t addr;
1226                size_t p_memsz;
1227                size_t p_filesz;
1228                size_t zfodlen;
1229                offset_t p_offset;
1230                size_t dif;
1231                int prot;

1233                /* Always need to adjust mr_addr */
1234                addr = start_addr + (size_t)(mrp[i].mr_addr);
1235                mrp[i].mr_addr =
1236                    (caddr_t)((uintptr_t)addr & (uintptr_t)PAGEMASK);

1238                /* Padding has already been mapped */
1239                if (MR_GET_TYPE(mrp[i].mr_flags) == MR_PADDING) {
1240                        continue;
1241                }
1242                p_memsz = mrp[i].mr_msize;
1243                p_filesz = mrp[i].mr_fsize;
1244                zfodlen = p_memsz - p_filesz;
1245                p_offset = mrp[i].mr_offset;
1246                dif = (uintptr_t)(addr) & PAGEOFFSET;
1247                prot = mrp[i].mr_prot | PROT_USER;
1248                ret = mmapobj_map_ptload(vp, addr, p_filesz, zfodlen,
1249                    p_offset, prot, fcred);
1250                if (ret != 0) {
1251                        MOBJ_STAT_ADD(ptload_failed);
1252                        mmapobj_unmap(mrp, i, num_elements, e_type);
1253                        return (ret);
1254                }

1256                /* Need to cleanup mrp to reflect the actual values used */
1257                mrp[i].mr_msize += dif;
1258                mrp[i].mr_offset = (size_t)addr & PAGEOFFSET;
1259        }

1261        /* Also need to unmap any holes created above */
1262        if (num_elements == 1) {
1263                MOBJ_STAT_ADD(map_elf_no_holes);
1264                return (0);
1265        }
1266        if (e_type == ET_EXEC) {
1267                return (0);
1268        }

1270        as_rangelock(as);
1271        lo = start_addr;
1272        hi = mrp[0].mr_addr;

1274        /* Remove holes made by the rest of the segments */
1275        for (i = 0; i < num_elements - 1; i++) {
1276                lo = (caddr_t)P2ROUNDUP((size_t)(mrp[i].mr_addr) +
1277                    mrp[i].mr_msize, PAGESIZE);
1278                hi = mrp[i + 1].mr_addr;
1279                if (lo < hi) {
1280                        /*
1281                         * If as_unmap fails we just use up a bit of extra
1282                         * space
1283                         */
1284                        (void) as_unmap(as, (caddr_t)lo,
1285                            (size_t)hi - (size_t)lo);
1286                        MOBJ_STAT_ADD(unmap_hole);
1287                }
```

```
1288         }
1289         as_rangeunlock(as);

1291         return (0);
1292 }

1294 /* Ugly hack to get STRUCT_* macros to work below */
1295 struct myphdr {
1296         Phdr            x;      /* native version */
1297 };

1299 struct myphdr32 {
1300         Elf32_Phdr      x;
1301 };

1303 /*
1304  * Calculate and return the number of loadable segments in the ELF Phdr
1305  * represented by phdrbase as well as the len of the total mapping and
1306  * the max alignment that is needed for a given segment.  On success,
1307  * 0 is returned, and *len, *loadable and *align have been filled out.
1308  * On failure, errno will be returned, which in this case is ENOTSUP
1309  * if we were passed an ELF file with overlapping segments.
1310  */
1311 static int
1312 calc_loadable(Ehdr *ehdrp, caddr_t phdrbase, int nphdrs, size_t *len,
1313     int *loadable, size_t *align)
1314 {
1315         int i;
1316         int hsize;
1317         model_t model;
1318         ushort_t e_type = ehdrp->e_type;        /* same offset 32 and 64 bit */
1319         uint_t p_type;
1320         offset_t p_offset;
1321         size_t p_memsz;
1322         size_t p_align;
1323         caddr_t vaddr;
1324         int num_segs = 0;
1325         caddr_t start_addr = NULL;
1326         caddr_t p_end = NULL;
1327         size_t max_align = 0;
1328         size_t min_align = PAGESIZE;    /* needed for vmem_xalloc */
1329         STRUCT_HANDLE(myphdr, mph);
1330 #if defined(__sparc)
1331         extern int vac_size;

1333         /*
1334          * Want to prevent aliasing by making the start address at least be
1335          * aligned to vac_size.
1336          */
1337         min_align = MAX(PAGESIZE, vac_size);
1338 #endif

1340         model = get_udatamodel();
1341         STRUCT_SET_HANDLE(mph, model, (struct myphdr *)phdrbase);

1343         /* hsize alignment should have been checked before calling this func */
1344         if (model == DATAMODEL_LP64) {
1345                 hsize = ehdrp->e_phentsize;
1346                 if (hsize & 7) {
1347                         return (ENOTSUP);
1348                 }
1349         } else {
1350                 ASSERT(model == DATAMODEL_ILP32);
1351                 hsize = ((Elf32_Ehdr *)ehdrp)->e_phentsize;
1352                 if (hsize & 3) {
1353                         return (ENOTSUP);
```



```
1354                 }
1355         }

1357         /*
1358          * Determine the span of all loadable segments and calculate the
1359          * number of loadable segments.
1360          */
1361         for (i = 0; i < nphdrs; i++) {
1362                 p_type = STRUCT_FGET(mph, x.p_type);
1363                 if (p_type == PT_LOAD || p_type == PT_SUNWBSS) {
1364                         vaddr = (caddr_t)(uintptr_t)STRUCT_FGET(mph, x.p_vaddr);
1365                         p_memsz = STRUCT_FGET(mph, x.p_memsz);

1367                         /*
1368                          * Skip this header if it requests no memory to be
1369                          * mapped.
1370                          */
1371                         if (p_memsz == 0) {
1372                                 STRUCT_SET_HANDLE(mph, model,
1373                                     (struct myphdr *)((size_t)STRUCT_BUF(mph) +
1374                                     hsize));
1375                                 MOBJ_STAT_ADD(nomem_header);
1376                                 continue;
1377                         }
1378                         if (num_segs++ == 0) {
1379                                 /*
1380                                  * The p_vaddr of the first PT_LOAD segment
1381                                  * must either be NULL or within the first
1382                                  * page in order to be interpreted.
1383                                  * Otherwise, its an invalid file.
1384                                  */
1385                                 if (e_type == ET_DYN &&
1386                                     ((caddr_t)((uintptr_t)vaddr &
1387                                     (uintptr_t)PAGEMASK) != NULL)) {
1388                                         MOBJ_STAT_ADD(inval_header);
1389                                         return (ENOTSUP);
1390                                 }
1391                                 start_addr = vaddr;
1392                                 /*
1393                                  * For the first segment, we need to map from
1394                                  * the beginning of the file, so we will
1395                                  * adjust the size of the mapping to include
1396                                  * this memory.
1397                                  */
1398                                 p_offset = STRUCT_FGET(mph, x.p_offset);
1399                         } else {
1400                                 p_offset = 0;
1401                         }
1402                         /*
1403                          * Check to make sure that this mapping wouldn't
1404                          * overlap a previous mapping.
1405                          */
1406                         if (vaddr < p_end) {
1407                                 MOBJ_STAT_ADD(overlap_header);
1408                                 return (ENOTSUP);
1409                         }

1411                         p_end = vaddr + p_memsz + p_offset;
1412                         p_end = (caddr_t)P2ROUNDUP((size_t)p_end, PAGESIZE);

1414                         p_align = STRUCT_FGET(mph, x.p_align);
1415                         if (p_align > 1 && p_align > max_align) {
1416                                 max_align = p_align;
1417                                 if (max_align < min_align) {
1418                                         max_align = min_align;
1419                                         MOBJ_STAT_ADD(min_align);
```

```
1420                                }
1421                        }
1422                }
1423                STRUCT_SET_HANDLE(mph, model,
1424                    (struct myphdr *)((size_t)STRUCT_BUF(mph) + hsize));
1425        }

1427        /*
1428         * The alignment should be a power of 2, if it isn't we forgive it
1429         * and round up.  On overflow, we'll set the alignment to max_align
1430         * rounded down to the nearest power of 2.
1431         */
1432        if (max_align > 0 && !ISP2(max_align)) {
1433                MOBJ_STAT_ADD(np2_align);
1434                *align = 2 * (1L << (highbit(max_align) - 1));
1435                if (*align < max_align ||
1436                    (*align > UINT_MAX && model == DATAMODEL_ILP32)) {
1437                        MOBJ_STAT_ADD(np2_align_overflow);
1438                        *align = 1L << (highbit(max_align) - 1);
1439                }
1440        } else {
1441                *align = max_align;
1442        }

1444        ASSERT(*align >= PAGESIZE || *align == 0);

1446        *loadable = num_segs;
1447        *len = p_end - start_addr;
1448        return (0);
1449 }

1451 /*
1452  * Check the address space to see if the virtual addresses to be used are
1453  * available.  If they are not, return errno for failure.  On success, 0
1454  * will be returned, and the virtual addresses for each mmapobj_result_t
1455  * will be reserved.  Note that a reservation could have earlier been made
1456  * for a given segment via a /dev/null mapping.  If that is the case, then
1457  * we can use that VA space for our mappings.
1458  * Note: this function will only be used for ET_EXEC binaries.
1459  */
1460 int
1461 check_exec_addrs(int loadable, mmapobj_result_t *mrp, caddr_t start_addr)
1462 {
1463        int i;
1464        struct as *as = curproc->p_as;
1465        struct segvn_crargs crargs = SEGVN_ZFOD_ARGS(PROT_ZFOD, PROT_ALL);
1466        int ret;
1467        caddr_t myaddr;
1468        size_t mylen;
1469        struct seg *seg;

1471        /* No need to reserve swap space now since it will be reserved later */
1472        crargs.flags |= MAP_NORESERVE;
1473        as_rangelock(as);
1474        for (i = 0; i < loadable; i++) {

1476                myaddr = start_addr + (size_t)mrp[i].mr_addr;
1477                mylen = mrp[i].mr_msize;

1479                /* See if there is a hole in the as for this range */
1480                if (as_gap(as, mylen, &myaddr, &mylen, 0, NULL) == 0) {
1481                        ASSERT(myaddr == start_addr + (size_t)mrp[i].mr_addr);
1482                        ASSERT(mylen == mrp[i].mr_msize);

1484 #ifdef DEBUG
1485                        if (MR_GET_TYPE(mrp[i].mr_flags) == MR_PADDING) {
```



```
1486                                MOBJ_STAT_ADD(exec_padding);
1487                        }
1488 #endif
1489                        ret = as_map(as, myaddr, mylen, segvn_create, &crargs);
1490                        if (ret) {
1491                                as_rangeunlock(as);
1492                                mmapobj_unmap_exec(mrp, i, start_addr);
1493                                return (ret);
1494                        }
1495                } else {
1496                        /*
1497                         * There is a mapping that exists in the range
1498                         * so check to see if it was a "reservation"
1499                         * from /dev/null.  The mapping is from
1500                         * /dev/null if the mapping comes from
1501                         * segdev and the type is neither MAP_SHARED
1502                         * nor MAP_PRIVATE.
1503                         */
1504                        AS_LOCK_ENTER(as, &as->a_lock, RW_READER);
1505                        seg = as_findseg(as, myaddr, 0);
1506                        MOBJ_STAT_ADD(exec_addr_mapped);
1507                        if (seg && seg->s_ops == &segdev_ops &&
1508                            ((SEGOP_GETTYPE(seg, myaddr) &
1509                            (MAP_SHARED | MAP_PRIVATE)) == 0) &&
1510                            myaddr >= seg->s_base &&
1511                            myaddr + mylen <=
1512                            seg->s_base + seg->s_size) {
1513                                MOBJ_STAT_ADD(exec_addr_devnull);
1514                                AS_LOCK_EXIT(as, &as->a_lock);
1515                                (void) as_unmap(as, myaddr, mylen);
1516                                ret = as_map(as, myaddr, mylen, segvn_create,
1517                                    &crargs);
1518                                mrp[i].mr_flags |= MR_RESV;
1519                                if (ret) {
1520                                        as_rangeunlock(as);
1521                                        /* Need to remap what we unmapped */
1522                                        mmapobj_unmap_exec(mrp, i + 1,
1523                                            start_addr);
1524                                        return (ret);
1525                                }
1526                        } else {
1527                                AS_LOCK_EXIT(as, &as->a_lock);
1528                                as_rangeunlock(as);
1529                                mmapobj_unmap_exec(mrp, i, start_addr);
1530                                MOBJ_STAT_ADD(exec_addr_in_use);
1531                                return (EADDRINUSE);
1532                        }
1533                }
1534        }
1535        as_rangeunlock(as);
1536        return (0);
1537 }

1539 /*
1540  * Walk through the ELF program headers and extract all useful information
1541  * for PT_LOAD and PT_SUNWBSS segments into mrp.
1542  * Return 0 on success or error on failure.
1543  */
1544 static int
1545 process_phdrs(Ehdr *ehdrp, caddr_t phdrbase, int nphdrs, mmapobj_result_t *mrp,
 736 process_phdr(Ehdr *ehdrp, caddr_t phdrbase, int nphdrs, mmapobj_result_t *mrp,
1546     vnode_t *vp, uint_t *num_mapped, size_t padding, cred_t *fcred)
1547 {
1548        int i;
1549        caddr_t start_addr = NULL;
1550        caddr_t vaddr;
```

```
1551         size_t len = 0;
1552         size_t lib_len = 0;
1553         int ret;
1554         int prot;
1555         struct lib_va *lvp = NULL;
1556         vattr_t vattr;
1557         struct as *as = curproc->p_as;
1558         int error;
1559         int loadable = 0;
1560         int current = 0;
1561         int use_lib_va = 1;
1562         size_t align = 0;
1563         size_t add_pad = 0;
1564         int hdr_seen = 0;
1565         ushort_t e_type = ehdrp->e_type;        /* same offset 32 and 64 bit */
1566         uint_t p_type;
1567         offset_t p_offset;
1568         size_t p_memsz;
1569         size_t p_filesz;
1570         uint_t p_flags;
1571         int hsize;
1572         model_t model;
1573         STRUCT_HANDLE(myphdr, mph);

1575         model = get_udatamodel();
1576         STRUCT_SET_HANDLE(mph, model, (struct myphdr *)phdrbase);

1578         /*
1579          * Need to make sure that hsize is aligned properly.
1580          * For 32bit processes, 4 byte alignment is required.
1581          * For 64bit processes, 8 byte alignment is required.
1582          * If the alignment isn't correct, we need to return failure
1583          * since it could cause an alignment error panic while walking
1584          * the phdr array.
1585          */
1586         if (model == DATAMODEL_LP64) {
1587                 hsize = ehdrp->e_phentsize;
1588                 if (hsize & 7) {
1589                         MOBJ_STAT_ADD(phent_align64);
1590                         return (ENOTSUP);
1591                 }
1592         } else {
1593                 ASSERT(model == DATAMODEL_ILP32);
1594                 hsize = ((Elf32_Ehdr *)ehdrp)->e_phentsize;
1595                 if (hsize & 3) {
1596                         MOBJ_STAT_ADD(phent_align32);
1597                         return (ENOTSUP);
1598                 }
1599         }

1601         if ((padding != 0) || secflag_enabled(curproc, PROC_SEC_ASLR)) {
 792         if (padding != 0) {
1602                 use_lib_va = 0;
1603         }
1604         if (e_type == ET_DYN) {
1605                 vattr.va_mask = AT_FSID | AT_NODEID | AT_CTIME | AT_MTIME;
1606                 error = VOP_GETATTR(vp, &vattr, 0, fcred, NULL);
1607                 if (error) {
1608                         return (error);
1609                 }
1610                 /* Check to see if we already have a description for this lib */
1611                 if (!secflag_enabled(curproc, PROC_SEC_ASLR))
1612 #endif /* ! codereview */
1613                         lvp = lib_va_find(&vattr);

1615                 if (lvp != NULL) {
```



```
1616                         MOBJ_STAT_ADD(lvp_found);
1617                         if (use_lib_va) {
1618                                 start_addr = mmapobj_lookup_start_addr(lvp);
1619                                 if (start_addr == NULL) {
1620                                         lib_va_release(lvp);
1621                                         return (ENOMEM);
1622                                 }
1623                         }

1625                         /*
1626                          * loadable may be zero if the original allocator
1627                          * of lvp hasn't finished setting it up but the rest
1628                          * of the fields will be accurate.
1629                          */
1630                         loadable = lvp->lv_num_segs;
1631                         len = lvp->lv_len;
1632                         align = lvp->lv_align;
1633                 }
1634         }

1636         /*
1637          * Determine the span of all loadable segments and calculate the
1638          * number of loadable segments, the total len spanned by the mappings
1639          * and the max alignment, if we didn't get them above.
1640          */
1641         if (loadable == 0) {
1642                 MOBJ_STAT_ADD(no_loadable_yet);
1643                 ret = calc_loadable(ehdrp, phdrbase, nphdrs, &len,
1644                     &loadable, &align);
1645                 if (ret != 0) {
1646                         /*
1647                          * Since it'd be an invalid file, we shouldn't have
1648                          * cached it previously.
1649                          */
1650                         ASSERT(lvp == NULL);
1651                         return (ret);
1652                 }
1653 #ifdef DEBUG
1654                 if (lvp) {
1655                         ASSERT(len == lvp->lv_len);
1656                         ASSERT(align == lvp->lv_align);
1657                 }
1658 #endif
1659         }

1661         /* Make sure there's something to map. */
1662         if (len == 0 || loadable == 0) {
1663                 /*
1664                  * Since it'd be an invalid file, we shouldn't have
1665                  * cached it previously.
1666                  */
1667                 ASSERT(lvp == NULL);
1668                 MOBJ_STAT_ADD(nothing_to_map);
1669                 return (ENOTSUP);
1670         }

1672         lib_len = len;
1673         if (padding != 0) {
1674                 loadable += 2;
1675         }
1676         if (loadable > *num_mapped) {
1677                 *num_mapped = loadable;
1678                 /* cleanup previous reservation */
1679                 if (start_addr) {
1680                         (void) as_unmap(as, start_addr, lib_len);
1681                 }
```

```
1682                MOBJ_STAT_ADD(e2big);
1683                if (lvp) {
1684                        lib_va_release(lvp);
1685                }
1686                return (E2BIG);
1687        }

1689        /*
1690         * We now know the size of the object to map and now we need to
1691         * get the start address to map it at.  It's possible we already
1692         * have it if we found all the info we need in the lib_va cache.
1693         */
1694        if (e_type == ET_DYN && start_addr == NULL) {
1695                /*
1696                 * Need to make sure padding does not throw off
1697                 * required alignment.  We can only specify an
1698                 * alignment for the starting address to be mapped,
1699                 * so we round padding up to the alignment and map
1700                 * from there and then throw out the extra later.
1701                 */
1702                if (padding != 0) {
1703                        if (align > 1) {
1704                                add_pad = P2ROUNDUP(padding, align);
1705                                len += add_pad;
1706                                MOBJ_STAT_ADD(dyn_pad_align);
1707                        } else {
1708                                MOBJ_STAT_ADD(dyn_pad_noalign);
1709                                len += padding; /* at beginning */
1710                        }
1711                        len += padding; /* at end of mapping */
1712                }
1713                /*
1714                 * At this point, if lvp is non-NULL, then above we
1715                 * already found it in the cache but did not get
1716                 * the start address since we were not going to use lib_va.
1717                 * Since we know that lib_va will not be used, it's safe
1718                 * to call mmapobj_alloc_start_addr and know that lvp
1719                 * will not be modified.
1720                 */
1721                ASSERT(lvp ? use_lib_va == 0 : 1);
1722                start_addr = mmapobj_alloc_start_addr(&lvp, len,
1723                    use_lib_va,
1724                    secflag_enabled(curproc, PROC_SEC_ASLR),
1725                    align, &vattr);
 802                use_lib_va, align, &vattr);
1726                if (start_addr == NULL) {
1727                        if (lvp) {
1728                                lib_va_release(lvp);
1729                        }
1730                        MOBJ_STAT_ADD(alloc_start_fail);
1731                        return (ENOMEM);
1732                }
1733                /*
1734                 * If we can't cache it, no need to hang on to it.
1735                 * Setting lv_num_segs to non-zero will make that
1736                 * field active and since there are too many segments
1737                 * to cache, all future users will not try to use lv_mps.
1738                 */
1739                if (lvp != NULL && loadable > LIBVA_CACHED_SEGS && use_lib_va) {
1740                        lvp->lv_num_segs = loadable;
1741                        lib_va_release(lvp);
1742                        lvp = NULL;
1743                        MOBJ_STAT_ADD(lvp_nocache);
1744                }
1745                /*
1746                 * Free the beginning of the mapping if the padding
```



```
1747                 * was not aligned correctly.
1748                 */
1749                if (padding != 0 && add_pad != padding) {
1750                        (void) as_unmap(as, start_addr,
1751                            add_pad - padding);
1752                        start_addr += (add_pad - padding);
1753                        MOBJ_STAT_ADD(extra_padding);
1754                }
1755        }

1757        /*
1758         * At this point, we have reserved the virtual address space
1759         * for our mappings.  Now we need to start filling out the mrp
1760         * array to describe all of the individual mappings we are going
1761         * to return.
1762         * For ET_EXEC there has been no memory reservation since we are
1763         * using fixed addresses.  While filling in the mrp array below,
1764         * we will have the first segment biased to start at addr 0
1765         * and the rest will be biased by this same amount.  Thus if there
1766         * is padding, the first padding will start at addr 0, and the next
1767         * segment will start at the value of padding.
1768         */

1770        /* We'll fill out padding later, so start filling in mrp at index 1 */
1771        if (padding != 0) {
1772                current = 1;
1773        }

1775        /* If we have no more need for lvp let it go now */
1776        if (lvp != NULL && use_lib_va == 0) {
1777                lib_va_release(lvp);
1778                MOBJ_STAT_ADD(lvp_not_needed);
1779                lvp = NULL;
1780        }

1782        /* Now fill out the mrp structs from the program headers */
1783        STRUCT_SET_HANDLE(mph, model, (struct myphdr *)phdrbase);
1784        for (i = 0; i < nphdrs; i++) {
1785                p_type = STRUCT_FGET(mph, x.p_type);
1786                if (p_type == PT_LOAD || p_type == PT_SUNWBSS) {
1787                        vaddr = (caddr_t)(uintptr_t)STRUCT_FGET(mph, x.p_vaddr);
1788                        p_memsz = STRUCT_FGET(mph, x.p_memsz);
1789                        p_filesz = STRUCT_FGET(mph, x.p_filesz);
1790                        p_offset = STRUCT_FGET(mph, x.p_offset);
1791                        p_flags = STRUCT_FGET(mph, x.p_flags);

1793                        /*
1794                         * Skip this header if it requests no memory to be
1795                         * mapped.
1796                         */
1797                        if (p_memsz == 0) {
1798                                STRUCT_SET_HANDLE(mph, model,
1799                                    (struct myphdr *)((size_t)STRUCT_BUF(mph) +
1800                                    hsize));
1801                                MOBJ_STAT_ADD(no_mem_map_sz);
1802                                continue;
1803                        }

1805                        prot = 0;
1806                        if (p_flags & PF_R)
1807                                prot |= PROT_READ;
1808                        if (p_flags & PF_W)
1809                                prot |= PROT_WRITE;
1810                        if (p_flags & PF_X)
1811                                prot |= PROT_EXEC;
```

```
1813                        ASSERT(current < loadable);
1814                        mrp[current].mr_msize = p_memsz;
1815                        mrp[current].mr_fsize = p_filesz;
1816                        mrp[current].mr_offset = p_offset;
1817                        mrp[current].mr_prot = prot;

1819                        if (hdr_seen == 0 && p_filesz != 0) {
1820                                mrp[current].mr_flags = MR_HDR_ELF;
1821                                /*
1822                                 * We modify mr_offset because we
1823                                 * need to map the ELF header as well, and if
1824                                 * we didn't then the header could be left out
1825                                 * of the mapping that we will create later.
1826                                 * Since we're removing the offset, we need to
1827                                 * account for that in the other fields as well
1828                                 * since we will be mapping the memory from 0
1829                                 * to p_offset.
1830                                 */
1831                                if (e_type == ET_DYN) {
1832                                        mrp[current].mr_offset = 0;
1833                                        mrp[current].mr_msize += p_offset;
1834                                        mrp[current].mr_fsize += p_offset;
1835                                } else {
1836                                        ASSERT(e_type == ET_EXEC);
1837                                        /*
1838                                         * Save off the start addr which will be
1839                                         * our bias for the rest of the
1840                                         * ET_EXEC mappings.
1841                                         */
1842                                        start_addr = vaddr - padding;
1843                                }
1844                                mrp[current].mr_addr = (caddr_t)padding;
1845                                hdr_seen = 1;
1846                        } else {
1847                                if (e_type == ET_EXEC) {
1848                                        /* bias mr_addr */
1849                                        mrp[current].mr_addr =
1850                                            vaddr - (size_t)start_addr;
1851                                } else {
1852                                        mrp[current].mr_addr = vaddr + padding;
1853                                }
1854                                mrp[current].mr_flags = 0;
1855                        }
1856                        current++;
1857                }

1859                /* Move to next phdr */
1860                STRUCT_SET_HANDLE(mph, model,
1861                    (struct myphdr *)((size_t)STRUCT_BUF(mph) +
1862                    hsize));
1863        }

1865        /* Now fill out the padding segments */
1866        if (padding != 0) {
1867                mrp[0].mr_addr = NULL;
1868                mrp[0].mr_msize = padding;
1869                mrp[0].mr_fsize = 0;
1870                mrp[0].mr_offset = 0;
1871                mrp[0].mr_prot = 0;
1872                mrp[0].mr_flags = MR_PADDING;

1874                /* Setup padding for the last segment */
1875                ASSERT(current == loadable - 1);
1876                mrp[current].mr_addr = (caddr_t)lib_len + padding;
1877                mrp[current].mr_msize = padding;
1878                mrp[current].mr_fsize = 0;
```



```
1879                mrp[current].mr_offset = 0;
1880                mrp[current].mr_prot = 0;
1881                mrp[current].mr_flags = MR_PADDING;
1882        }

1884        /*
1885         * Need to make sure address ranges desired are not in use or
1886         * are previously allocated reservations from /dev/null.  For
1887         * ET_DYN, we already made sure our address range was free.
1888         */
1889        if (e_type == ET_EXEC) {
1890                ret = check_exec_addrs(loadable, mrp, start_addr);
1891                if (ret != 0) {
1892                        ASSERT(lvp == NULL);
1893                        MOBJ_STAT_ADD(check_exec_failed);
1894                        return (ret);
1895                }
1896        }

1898        /* Finish up our business with lvp. */
1899        if (lvp) {
1900                ASSERT(e_type == ET_DYN);
1901                if (lvp->lv_num_segs == 0 && loadable <= LIBVA_CACHED_SEGS) {
1902                        bcopy(mrp, lvp->lv_mps,
1903                            loadable * sizeof (mmapobj_result_t));
1904                        membar_producer();
1905                }
1906                /*
1907                 * Setting lv_num_segs to a non-zero value indicates that
1908                 * lv_mps is now valid and can be used by other threads.
1909                 * So, the above stores need to finish before lv_num_segs
1910                 * is updated. lv_mps is only valid if lv_num_segs is
1911                 * greater than LIBVA_CACHED_SEGS.
1912                 */
1913                lvp->lv_num_segs = loadable;
1914                lib_va_release(lvp);
1915                MOBJ_STAT_ADD(lvp_used);
1916        }

1918        /* Now that we have mrp completely filled out go map it */
1919        ret = mmapobj_map_elf(vp, start_addr, mrp, loadable, fcred, e_type);
1920        if (ret == 0) {
1921                *num_mapped = loadable;
1922        }

1924        return (ret);
1925 }

1927 /*
1928  * Take the ELF file passed in, and do the work of mapping it.
1929  * num_mapped in - # elements in user buffer
1930  * num_mapped out - # sections mapped and length of mrp array if
1931  *                      no errors.
1932  */
1933 static int
1934 doelfwork(Ehdr *ehdrp, vnode_t *vp, mmapobj_result_t *mrp,
1935     uint_t *num_mapped, size_t padding, cred_t *fcred)
1936 {
1937        int error;
1938        offset_t phoff;
1939        int nphdrs;
1940        unsigned char ei_class;
1941        unsigned short phentsize;
1942        ssize_t phsizep;
1943        caddr_t phbasep;
1944        int to_map;
```

```
1945         model_t model;

1947         ei_class = ehdrp->e_ident[EI_CLASS];
1948         model = get_udatamodel();
1949         if ((model == DATAMODEL_ILP32 && ei_class == ELFCLASS64) ||
1950             (model == DATAMODEL_LP64 && ei_class == ELFCLASS32)) {
1951                 MOBJ_STAT_ADD(wrong_model);
1952                 return (ENOTSUP);
1953         }

1955         /* Can't execute code from "noexec" mounted filesystem. */
1956         if (ehdrp->e_type == ET_EXEC &&
1957             (vp->v_vfsp->vfs_flag & VFS_NOEXEC) != 0) {
1958                 MOBJ_STAT_ADD(noexec_fs);
1959                 return (EACCES);
1960         }

1962         /*
1963          * Relocatable and core files are mapped as a single flat file
1964          * since no interpretation is done on them by mmapobj.
1965          */
1966         if (ehdrp->e_type == ET_REL || ehdrp->e_type == ET_CORE) {
1967                 to_map = padding ? 3 : 1;
1968                 if (*num_mapped < to_map) {
1969                         *num_mapped = to_map;
1970                         MOBJ_STAT_ADD(e2big_et_rel);
1971                         return (E2BIG);
1972                 }
1973                 error = mmapobj_map_flat(vp, mrp, padding, fcred);
1974                 if (error == 0) {
1975                         *num_mapped = to_map;
1976                         mrp[padding ? 1 : 0].mr_flags = MR_HDR_ELF;
1977                         MOBJ_STAT_ADD(et_rel_mapped);
1978                 }
1979                 return (error);
1980         }

1982         /* Check for an unknown ELF type */
1983         if (ehdrp->e_type != ET_EXEC && ehdrp->e_type != ET_DYN) {
1984                 MOBJ_STAT_ADD(unknown_elf_type);
1985                 return (ENOTSUP);
1986         }

1988         if (ei_class == ELFCLASS32) {
1989                 Elf32_Ehdr *e32hdr = (Elf32_Ehdr *)ehdrp;
1990                 ASSERT(model == DATAMODEL_ILP32);
1991                 nphdrs = e32hdr->e_phnum;
1992                 phentsize = e32hdr->e_phentsize;
1993                 if (phentsize < sizeof (Elf32_Phdr)) {
1994                         MOBJ_STAT_ADD(phent32_too_small);
1995                         return (ENOTSUP);
1996                 }
1997                 phoff = e32hdr->e_phoff;
1998         } else if (ei_class == ELFCLASS64) {
1999                 Elf64_Ehdr *e64hdr = (Elf64_Ehdr *)ehdrp;
2000                 ASSERT(model == DATAMODEL_LP64);
2001                 nphdrs = e64hdr->e_phnum;
2002                 phentsize = e64hdr->e_phentsize;
2003                 if (phentsize < sizeof (Elf64_Phdr)) {
2004                         MOBJ_STAT_ADD(phent64_too_small);
2005                         return (ENOTSUP);
2006                 }
2007                 phoff = e64hdr->e_phoff;
2008         } else {
2009                 /* fallthrough case for an invalid ELF class */
2010                 MOBJ_STAT_ADD(inval_elf_class);
```



```
2011                 return (ENOTSUP);
2012         }

2014         /*
2015          * nphdrs should only have this value for core files which are handled
2016          * above as a single mapping.  If other file types ever use this
2017          * sentinel, then we'll add the support needed to handle this here.
2018          */
2019         if (nphdrs == PN_XNUM) {
2020                 MOBJ_STAT_ADD(too_many_phdrs);
2021                 return (ENOTSUP);
2022         }

2024         phsizep = nphdrs * phentsize;

2026         if (phsizep == 0) {
2027                 MOBJ_STAT_ADD(no_phsize);
2028                 return (ENOTSUP);
2029         }

2031         /* Make sure we only wait for memory if it's a reasonable request */
2032         if (phsizep > mmapobj_alloc_threshold) {
2033                 MOBJ_STAT_ADD(phsize_large);
2034                 if ((phbasep = kmem_alloc(phsizep, KM_NOSLEEP)) == NULL) {
2035                         MOBJ_STAT_ADD(phsize_xtralarge);
2036                         return (ENOMEM);
2037                 }
2038         } else {
2039                 phbasep = kmem_alloc(phsizep, KM_SLEEP);
2040         }

2042         if ((error = vn_rdwr(UIO_READ, vp, phbasep, phsizep,
2043             (offset_t)phoff, UIO_SYSSPACE, 0, (rlim64_t)0,
2044             fcred, NULL)) != 0) {
2045                 kmem_free(phbasep, phsizep);
2046                 return (error);
2047         }

2049         /* Now process the phdr's */
2050         error = process_phdrs(ehdrp, phbasep, nphdrs, mrp, vp, num_mapped,
1127         error = process_phdr(ehdrp, phbasep, nphdrs, mrp, vp, num_mapped,
2051             padding, fcred);
2052         kmem_free(phbasep, phsizep);
2053         return (error);
2054 }
_____unchanged_portion_omitted_
2288 #endif

2290 /*
2291  * These are the two types of files that we can interpret and we want to read
2292  * in enough info to cover both types when looking at the initial header.
2293  */
2294 #define MAX_HEADER_SIZE (MAX(sizeof (Ehdr), sizeof (struct exec)))

2296 /*
2297  * Map vp passed in in an interpreted manner.  ELF and AOUT files will be
2298  * interpreted and mapped appropriately for execution.
2299  * num_mapped in - # elements in mrp
2300  * num_mapped out - # sections mapped and length of mrp array if
2301  *                  no errors or E2BIG returned.
2302  *
2303  * Returns 0 on success, errno value on failure.
2304  */
2305 static int
2306 mmapobj_map_interpret(vnode_t *vp, mmapobj_result_t *mrp,
2307     uint_t *num_mapped, size_t padding, cred_t *fcred)
```

```
2308 {
2309         int error = 0;
2310         vattr_t vattr;
2311         struct lib_va *lvp;
2312         caddr_t start_addr;
2313         model_t model;

2315         /*
2316          * header has to be aligned to the native size of ulong_t in order
2317          * to avoid an unaligned access when dereferencing the header as
2318          * a ulong_t.  Thus we allocate our array on the stack of type
2319          * ulong_t and then have header, which we dereference later as a char
2320          * array point at lheader.
2321          */
2322         ulong_t lheader[(MAX_HEADER_SIZE / (sizeof (ulong_t))) + 1];
2323         caddr_t header = (caddr_t)&lheader;

2325         vattr.va_mask = AT_FSID | AT_NODEID | AT_CTIME | AT_MTIME | AT_SIZE;
2326         error = VOP_GETATTR(vp, &vattr, 0, fcred, NULL);
2327         if (error) {
2328                 return (error);
2329         }

2331         /*
2332          * Check lib_va to see if we already have a full description
2333          * for this library.  This is the fast path and only used for
2334          * ET_DYN ELF files (dynamic libraries).
2335          */
2336         if (padding == 0 && !secflag_enabled(curproc, PROC_SEC_ASLR) &&
2337             ((lvp = lib_va_find(&vattr)) != NULL)) {
1413         if (padding == 0 && (lvp = lib_va_find(&vattr)) != NULL) {
2338                 int num_segs;

2340                 model = get_udatamodel();
2341                 if ((model == DATAMODEL_ILP32 &&
2342                     lvp->lv_flags & LV_ELF64) ||
2343                     (model == DATAMODEL_LP64 &&
2344                     lvp->lv_flags & LV_ELF32)) {
2345                         lib_va_release(lvp);
2346                         MOBJ_STAT_ADD(fast_wrong_model);
2347                         return (ENOTSUP);
2348                 }
2349                 num_segs = lvp->lv_num_segs;
2350                 if (*num_mapped < num_segs) {
2351                         *num_mapped = num_segs;
2352                         lib_va_release(lvp);
2353                         MOBJ_STAT_ADD(fast_e2big);
2354                         return (E2BIG);
2355                 }

2357                 /*
2358                  * Check to see if we have all the mappable program headers
2359                  * cached.
2360                  */
2361                 if (num_segs <= LIBVA_CACHED_SEGS && num_segs != 0) {
2362                         MOBJ_STAT_ADD(fast);
2363                         start_addr = mmapobj_lookup_start_addr(lvp);
2364                         if (start_addr == NULL) {
2365                                 lib_va_release(lvp);
2366                                 return (ENOMEM);
2367                         }

2369                         bcopy(lvp->lv_mps, mrp,
2370                             num_segs * sizeof (mmapobj_result_t));

2372                         error = mmapobj_map_elf(vp, start_addr, mrp,
```



```
2373                             num_segs, fcred, ET_DYN);

2375                         lib_va_release(lvp);
2376                         if (error == 0) {
2377                                 *num_mapped = num_segs;
2378                                 MOBJ_STAT_ADD(fast_success);
2379                         }
2380                         return (error);
2381                 }
2382                 MOBJ_STAT_ADD(fast_not_now);

2384                 /* Release it for now since we'll look it up below */
2385                 lib_va_release(lvp);
2386         }

2388         /*
2389          * Time to see if this is a file we can interpret.  If it's smaller
2390          * than this, then we can't interpret it.
2391          */
2392         if (vattr.va_size < MAX_HEADER_SIZE) {
2393                 MOBJ_STAT_ADD(small_file);
2394                 return (ENOTSUP);
2395         }

2397         if ((error = vn_rdwr(UIO_READ, vp, header, MAX_HEADER_SIZE, 0,
2398             UIO_SYSSPACE, 0, (rlim64_t)0, fcred, NULL)) != 0) {
2399                 MOBJ_STAT_ADD(read_error);
2400                 return (error);
2401         }

2403         /* Verify file type */
2404         if (header[EI_MAG0] == ELFMAG0 && header[EI_MAG1] == ELFMAG1 &&
2405             header[EI_MAG2] == ELFMAG2 && header[EI_MAG3] == ELFMAG3) {
2406                 return (doelfwork((Ehdr *)lheader, vp, mrp, num_mapped,
2407                     padding, fcred));
2408         }

2410 #if defined(__sparc)
2411         /* On sparc, check for 4.X AOUT format */
2412         switch (((struct exec *)header)->a_magic) {
2413         case OMAGIC:
2414         case ZMAGIC:
2415         case NMAGIC:
2416                 return (doaoutwork(vp, mrp, num_mapped,
2417                     (struct exec *)lheader, fcred));
2418         }
2419 #endif

2421         /* Unsupported type */
2422         MOBJ_STAT_ADD(unsupported);
2423         return (ENOTSUP);
2424 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   63093 Wed May 27 19:49:30 2015
new/usr/src/uts/common/os/policy.c
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
______unchanged_portion_omitted_

1731 /* Process security flags */
1732 int
1733 secpolicy_psecflags(const cred_t *cr, proc_t *tp, proc_t *sp)
1734 {
1735         if (PRIV_POLICY(cr, PRIV_PROC_SECFLAGS, B_FALSE, EPERM, NULL) != 0)
1736                 return (EPERM);

1738         if (!prochasprocperm(tp, sp, cr))
1739                 return (EPERM);

1741         return (0);
1742 }

1744 #endif /* ! codereview */
1745 /*
1746  * Processor set binding.
1747  */
1748 int
1749 secpolicy_pbind(const cred_t *cr)
1750 {
1751         if (PRIV_POLICY_ONLY(cr, PRIV_SYS_RES_CONFIG, B_FALSE))
1752                 return (secpolicy_pset(cr));
1753         return (PRIV_POLICY(cr, PRIV_SYS_RES_BIND, B_FALSE, EPERM, NULL));
1754 }

1756 int
1757 secpolicy_ponline(const cred_t *cr)
1758 {
1759         return (PRIV_POLICY(cr, PRIV_SYS_RES_CONFIG, B_FALSE, EPERM, NULL));
1760 }

1762 int
1763 secpolicy_pool(const cred_t *cr)
1764 {
1765         return (PRIV_POLICY(cr, PRIV_SYS_RES_CONFIG, B_FALSE, EPERM, NULL));
1766 }

1768 int
1769 secpolicy_blacklist(const cred_t *cr)
1770 {
1771         return (PRIV_POLICY(cr, PRIV_SYS_RES_CONFIG, B_FALSE, EPERM, NULL));
1772 }

1774 /*
1775  * Catch all system configuration.
1776  */
1777 int
1778 secpolicy_sys_config(const cred_t *cr, boolean_t checkonly)
1779 {
1780         if (checkonly) {
1781                 return (PRIV_POLICY_ONLY(cr, PRIV_SYS_CONFIG, B_FALSE) ? 0 :
1782                     EPERM);
1783         } else {
1784                 return (PRIV_POLICY(cr, PRIV_SYS_CONFIG, B_FALSE, EPERM, NULL));
1785         }
1786 }
```



```
1788 /*
1789  * Zone administration (halt, reboot, etc.) from within zone.
1790  */
1791 int
1792 secpolicy_zone_admin(const cred_t *cr, boolean_t checkonly)
1793 {
1794         if (checkonly) {
1795                 return (PRIV_POLICY_ONLY(cr, PRIV_SYS_ADMIN, B_FALSE) ? 0 :
1796                     EPERM);
1797         } else {
1798                 return (PRIV_POLICY(cr, PRIV_SYS_ADMIN, B_FALSE, EPERM,
1799                     NULL));
1800         }
1801 }

1803 /*
1804  * Zone configuration (create, halt, enter).
1805  */
1806 int
1807 secpolicy_zone_config(const cred_t *cr)
1808 {
1809         /*
1810          * Require all privileges to avoid possibility of privilege
1811          * escalation.
1812          */
1813         return (secpolicy_require_set(cr, PRIV_FULLSET, NULL, KLPDARG_NONE));
1814 }

1816 /*
1817  * Various other system configuration calls
1818  */
1819 int
1820 secpolicy_coreadm(const cred_t *cr)
1821 {
1822         return (PRIV_POLICY(cr, PRIV_SYS_ADMIN, B_FALSE, EPERM, NULL));
1823 }

1825 int
1826 secpolicy_systeminfo(const cred_t *cr)
1827 {
1828         return (PRIV_POLICY(cr, PRIV_SYS_ADMIN, B_FALSE, EPERM, NULL));
1829 }

1831 int
1832 secpolicy_dispadm(const cred_t *cr)
1833 {
1834         return (PRIV_POLICY(cr, PRIV_SYS_CONFIG, B_FALSE, EPERM, NULL));
1835 }

1837 int
1838 secpolicy_settime(const cred_t *cr)
1839 {
1840         return (PRIV_POLICY(cr, PRIV_SYS_TIME, B_FALSE, EPERM, NULL));
1841 }

1843 /*
1844  * For realtime users: high resolution clock.
1845  */
1846 int
1847 secpolicy_clock_highres(const cred_t *cr)
1848 {
1849         return (PRIV_POLICY(cr, PRIV_PROC_CLOCK_HIGHRES, B_FALSE, EPERM,
1850             NULL));
1851 }
```

```
1853 /*
1854  * drv_priv() is documented as callable from interrupt context, not that
1855  * anyone ever does, but still.  No debugging or auditing can be done when
1856  * it is called from interrupt context.
1857  * returns 0 on succes, EPERM on failure.
1858  */
1859 int
1860 drv_priv(cred_t *cr)
1861 {
1862         return (PRIV_POLICY(cr, PRIV_SYS_DEVICES, B_FALSE, EPERM, NULL));
1863 }

1865 int
1866 secpolicy_sys_devices(const cred_t *cr)
1867 {
1868         return (PRIV_POLICY(cr, PRIV_SYS_DEVICES, B_FALSE, EPERM, NULL));
1869 }

1871 int
1872 secpolicy_excl_open(const cred_t *cr)
1873 {
1874         return (PRIV_POLICY(cr, PRIV_SYS_DEVICES, B_FALSE, EBUSY, NULL));
1875 }

1877 int
1878 secpolicy_rctlsys(const cred_t *cr, boolean_t is_zone_rctl)
1879 {
1880         /* zone.* rctls can only be set from the global zone */
1881         if (is_zone_rctl && priv_policy_global(cr) != 0)
1882                 return (EPERM);
1883         return (PRIV_POLICY(cr, PRIV_SYS_RESOURCE, B_FALSE, EPERM, NULL));
1884 }

1886 int
1887 secpolicy_resource(const cred_t *cr)
1888 {
1889         return (PRIV_POLICY(cr, PRIV_SYS_RESOURCE, B_FALSE, EPERM, NULL));
1890 }

1892 int
1893 secpolicy_resource_anon_mem(const cred_t *cr)
1894 {
1895         return (PRIV_POLICY_ONLY(cr, PRIV_SYS_RESOURCE, B_FALSE));
1896 }

1898 /*
1899  * Processes with a real uid of 0 escape any form of accounting, much
1900  * like before.
1901  */
1902 int
1903 secpolicy_newproc(const cred_t *cr)
1904 {
1905         if (cr->cr_ruid == 0)
1906                 return (0);

1908         return (PRIV_POLICY(cr, PRIV_SYS_RESOURCE, B_FALSE, EPERM, NULL));
1909 }

1911 /*
1912  * Networking
1913  */
1914 int
1915 secpolicy_net_rawaccess(const cred_t *cr)
1916 {
1917         return (PRIV_POLICY(cr, PRIV_NET_RAWACCESS, B_FALSE, EACCES, NULL));
1918 }
```



```
1920 int
1921 secpolicy_net_observability(const cred_t *cr)
1922 {
1923         return (PRIV_POLICY(cr, PRIV_NET_OBSERVABILITY, B_FALSE, EACCES, NULL));
1924 }

1926 /*
1927  * Need this privilege for accessing the ICMP device
1928  */
1929 int
1930 secpolicy_net_icmpaccess(const cred_t *cr)
1931 {
1932         return (PRIV_POLICY(cr, PRIV_NET_ICMPACCESS, B_FALSE, EACCES, NULL));
1933 }

1935 /*
1936  * There are a few rare cases where the kernel generates ioctls() from
1937  * interrupt context with a credential of kcred rather than NULL.
1938  * In those cases, we take the safe and cheap test.
1939  */
1940 int
1941 secpolicy_net_config(const cred_t *cr, boolean_t checkonly)
1942 {
1943         if (checkonly) {
1944                 return (PRIV_POLICY_ONLY(cr, PRIV_SYS_NET_CONFIG, B_FALSE) ?
1945                     0 : EPERM);
1946         } else {
1947                 return (PRIV_POLICY(cr, PRIV_SYS_NET_CONFIG, B_FALSE, EPERM,
1948                     NULL));
1949         }
1950 }


1953 /*
1954  * PRIV_SYS_NET_CONFIG is a superset of PRIV_SYS_IP_CONFIG.
1955  *
1956  * There are a few rare cases where the kernel generates ioctls() from
1957  * interrupt context with a credential of kcred rather than NULL.
1958  * In those cases, we take the safe and cheap test.
1959  */
1960 int
1961 secpolicy_ip_config(const cred_t *cr, boolean_t checkonly)
1962 {
1963         if (PRIV_POLICY_ONLY(cr, PRIV_SYS_NET_CONFIG, B_FALSE))
1964                 return (secpolicy_net_config(cr, checkonly));

1966         if (checkonly) {
1967                 return (PRIV_POLICY_ONLY(cr, PRIV_SYS_IP_CONFIG, B_FALSE) ?
1968                     0 : EPERM);
1969         } else {
1970                 return (PRIV_POLICY(cr, PRIV_SYS_IP_CONFIG, B_FALSE, EPERM,
1971                     NULL));
1972         }
1973 }

1975 /*
1976  * PRIV_SYS_NET_CONFIG is a superset of PRIV_SYS_DL_CONFIG.
1977  */
1978 int
1979 secpolicy_dl_config(const cred_t *cr)
1980 {
1981         if (PRIV_POLICY_ONLY(cr, PRIV_SYS_NET_CONFIG, B_FALSE))
1982                 return (secpolicy_net_config(cr, B_FALSE));
1983         return (PRIV_POLICY(cr, PRIV_SYS_DL_CONFIG, B_FALSE, EPERM, NULL));
1984 }
```

```
1986 /*
1987  * PRIV_SYS_DL_CONFIG is a superset of PRIV_SYS_IPTUN_CONFIG.
1988  */
1989 int
1990 secpolicy_iptun_config(const cred_t *cr)
1991 {
1992         if (PRIV_POLICY_ONLY(cr, PRIV_SYS_NET_CONFIG, B_FALSE))
1993                 return (secpolicy_net_config(cr, B_FALSE));
1994         if (PRIV_POLICY_ONLY(cr, PRIV_SYS_DL_CONFIG, B_FALSE))
1995                 return (secpolicy_dl_config(cr));
1996         return (PRIV_POLICY(cr, PRIV_SYS_IPTUN_CONFIG, B_FALSE, EPERM, NULL));
1997 }

1999 /*
2000  * Map IP pseudo privileges to actual privileges.
2001  * So we don't need to recompile IP when we change the privileges.
2002  */
2003 int
2004 secpolicy_ip(const cred_t *cr, int netpriv, boolean_t checkonly)
2005 {
2006         int priv = PRIV_ALL;

2008         switch (netpriv) {
2009         case OP_CONFIG:
2010                 priv = PRIV_SYS_IP_CONFIG;
2011                 break;
2012         case OP_RAW:
2013                 priv = PRIV_NET_RAWACCESS;
2014                 break;
2015         case OP_PRIVPORT:
2016                 priv = PRIV_NET_PRIVADDR;
2017                 break;
2018         }
2019         ASSERT(priv != PRIV_ALL);
2020         if (checkonly)
2021                 return (PRIV_POLICY_ONLY(cr, priv, B_FALSE) ? 0 : EPERM);
2022         else
2023                 return (PRIV_POLICY(cr, priv, B_FALSE, EPERM, NULL));
2024 }

2026 /*
2027  * Map network pseudo privileges to actual privileges.
2028  * So we don't need to recompile IP when we change the privileges.
2029  */
2030 int
2031 secpolicy_net(const cred_t *cr, int netpriv, boolean_t checkonly)
2032 {
2033         int priv = PRIV_ALL;

2035         switch (netpriv) {
2036         case OP_CONFIG:
2037                 priv = PRIV_SYS_NET_CONFIG;
2038                 break;
2039         case OP_RAW:
2040                 priv = PRIV_NET_RAWACCESS;
2041                 break;
2042         case OP_PRIVPORT:
2043                 priv = PRIV_NET_PRIVADDR;
2044                 break;
2045         }
2046         ASSERT(priv != PRIV_ALL);
2047         if (checkonly)
2048                 return (PRIV_POLICY_ONLY(cr, priv, B_FALSE) ? 0 : EPERM);
2049         else
2050                 return (PRIV_POLICY(cr, priv, B_FALSE, EPERM, NULL));
```



```
2051 }

2053 /*
2054  * Checks for operations that are either client-only or are used by
2055  * both clients and servers.
2056  */
2057 int
2058 secpolicy_nfs(const cred_t *cr)
2059 {
2060         return (PRIV_POLICY(cr, PRIV_SYS_NFS, B_FALSE, EPERM, NULL));
2061 }

2063 /*
2064  * Special case for opening rpcmod: have NFS privileges or network
2065  * config privileges.
2066  */
2067 int
2068 secpolicy_rpcmod_open(const cred_t *cr)
2069 {
2070         if (PRIV_POLICY_ONLY(cr, PRIV_SYS_NFS, B_FALSE))
2071                 return (secpolicy_nfs(cr));
2072         else
2073                 return (secpolicy_net_config(cr, NULL));
2074 }

2076 int
2077 secpolicy_chroot(const cred_t *cr)
2078 {
2079         return (PRIV_POLICY(cr, PRIV_PROC_CHROOT, B_FALSE, EPERM, NULL));
2080 }

2082 int
2083 secpolicy_tasksys(const cred_t *cr)
2084 {
2085         return (PRIV_POLICY(cr, PRIV_PROC_TASKID, B_FALSE, EPERM, NULL));
2086 }

2088 int
2089 secpolicy_pfexec_register(const cred_t *cr)
2090 {
2091         return (PRIV_POLICY(cr, PRIV_SYS_ADMIN, B_TRUE, EPERM, NULL));
2092 }

2094 /*
2095  * Basic privilege checks.
2096  */
2097 int
2098 secpolicy_basic_exec(const cred_t *cr, vnode_t *vp)
2099 {
2100         FAST_BASIC_CHECK(cr, PRIV_PROC_EXEC);

2102         return (priv_policy_va(cr, PRIV_PROC_EXEC, B_FALSE, EPERM, NULL,
2103             KLPDARG_VNODE, vp, (char *)NULL, KLPDARG_NOMORE));
2104 }

2106 int
2107 secpolicy_basic_fork(const cred_t *cr)
2108 {
2109         FAST_BASIC_CHECK(cr, PRIV_PROC_FORK);

2111         return (PRIV_POLICY(cr, PRIV_PROC_FORK, B_FALSE, EPERM, NULL));
2112 }

2114 int
2115 secpolicy_basic_proc(const cred_t *cr)
2116 {
```

```
2117         FAST_BASIC_CHECK(cr, PRIV_PROC_SESSION);

2119         return (PRIV_POLICY(cr, PRIV_PROC_SESSION, B_FALSE, EPERM, NULL));
2120 }

2122 /*
2123  * Slightly complicated because we don't want to trigger the policy too
2124  * often.  First we shortcircuit access to "self" (tp == sp) or if
2125  * we don't have the privilege but if we have permission
2126  * just return (0) and we don't flag the privilege as needed.
2127  * Else, we test for the privilege because we either have it or need it.
2128  */
2129 int
2130 secpolicy_basic_procinfo(const cred_t *cr, proc_t *tp, proc_t *sp)
2131 {
2132         if (tp == sp ||
2133             !HAS_PRIVILEGE(cr, PRIV_PROC_INFO) && prochasprocperm(tp, sp, cr)) {
2134                 return (0);
2135         } else {
2136                 return (PRIV_POLICY(cr, PRIV_PROC_INFO, B_FALSE, EPERM, NULL));
2137         }
2138 }

2140 int
2141 secpolicy_basic_link(const cred_t *cr)
2142 {
2143         FAST_BASIC_CHECK(cr, PRIV_FILE_LINK_ANY);

2145         return (PRIV_POLICY(cr, PRIV_FILE_LINK_ANY, B_FALSE, EPERM, NULL));
2146 }

2148 int
2149 secpolicy_basic_net_access(const cred_t *cr)
2150 {
2151         FAST_BASIC_CHECK(cr, PRIV_NET_ACCESS);

2153         return (PRIV_POLICY(cr, PRIV_NET_ACCESS, B_FALSE, EACCES, NULL));
2154 }

2156 /* ARGSUSED */
2157 int
2158 secpolicy_basic_file_read(const cred_t *cr, vnode_t *vp, const char *pn)
2159 {
2160         FAST_BASIC_CHECK(cr, PRIV_FILE_READ);

2162         return (priv_policy_va(cr, PRIV_FILE_READ, B_FALSE, EACCES, NULL,
2163             KLPDARG_VNODE, vp, (char *)pn, KLPDARG_NOMORE));
2164 }

2166 /* ARGSUSED */
2167 int
2168 secpolicy_basic_file_write(const cred_t *cr, vnode_t *vp, const char *pn)
2169 {
2170         FAST_BASIC_CHECK(cr, PRIV_FILE_WRITE);

2172         return (priv_policy_va(cr, PRIV_FILE_WRITE, B_FALSE, EACCES, NULL,
2173             KLPDARG_VNODE, vp, (char *)pn, KLPDARG_NOMORE));
2174 }

2176 /*
2177  * Additional device protection.
2178  *
2179  * Traditionally, a device has specific permissions on the node in
2180  * the filesystem which govern which devices can be opened by what
2181  * processes.  In certain cases, it is desirable to add extra
2182  * restrictions, as writing to certain devices is identical to
```



```
2183  * having a complete run of the system.
2184  *
2185  * This mechanism is called the device policy.
2186  *
2187  * When a device is opened, its policy entry is looked up in the
2188  * policy cache and checked.
2189  */
2190 int
2191 secpolicy_spec_open(const cred_t *cr, struct vnode *vp, int oflag)
2192 {
2193         devplcy_t *plcy;
2194         int err;
2195         struct snode *csp = VTOS(common_specvp(vp));
2196         priv_set_t pset;

2198         mutex_enter(&csp->s_lock);

2200         if (csp->s_plcy == NULL || csp->s_plcy->dp_gen != devplcy_gen) {
2201                 plcy = devpolicy_find(vp);
2202                 if (csp->s_plcy)
2203                         dpfree(csp->s_plcy);
2204                 csp->s_plcy = plcy;
2205                 ASSERT(plcy != NULL);
2206         } else
2207                 plcy = csp->s_plcy;

2209         if (plcy == nullpolicy) {
2210                 mutex_exit(&csp->s_lock);
2211                 return (0);
2212         }

2214         dphold(plcy);

2216         mutex_exit(&csp->s_lock);

2218         if (oflag & FWRITE)
2219                 pset = plcy->dp_wrp;
2220         else
2221                 pset = plcy->dp_rdp;
2222         /*
2223          * Special case:
2224          * PRIV_SYS_NET_CONFIG is a superset of PRIV_SYS_IP_CONFIG.
2225          * If PRIV_SYS_NET_CONFIG is present and PRIV_SYS_IP_CONFIG is
2226          * required, replace PRIV_SYS_IP_CONFIG with PRIV_SYS_NET_CONFIG
2227          * in the required privilege set before doing the check.
2228          */
2229         if (priv_ismember(&pset, PRIV_SYS_IP_CONFIG) &&
2230             priv_ismember(&CR_OEPRIV(cr), PRIV_SYS_NET_CONFIG) &&
2231             !priv_ismember(&CR_OEPRIV(cr), PRIV_SYS_IP_CONFIG)) {
2232                 priv_delset(&pset, PRIV_SYS_IP_CONFIG);
2233                 priv_addset(&pset, PRIV_SYS_NET_CONFIG);
2234         }

2236         err = secpolicy_require_set(cr, &pset, "devpolicy", KLPDARG_NONE);
2237         dpfree(plcy);

2239         return (err);
2240 }

2242 int
2243 secpolicy_modctl(const cred_t *cr, int cmd)
2244 {
2245         switch (cmd) {
2246         case MODINFO:
2247         case MODGETMAJBIND:
2248         case MODGETPATH:
```

```
2249         case MODGETPATHLEN:
2250         case MODGETNAME:
2251         case MODGETFBNAME:
2252         case MODGETDEVPOLICY:
2253         case MODGETDEVPOLICYBYNAME:
2254         case MODDEVT2INSTANCE:
2255         case MODSIZEOF_DEVID:
2256         case MODGETDEVID:
2257         case MODSIZEOF_MINORNAME:
2258         case MODGETMINORNAME:
2259         case MODGETDEVFSPATH_LEN:
2260         case MODGETDEVFSPATH:
2261         case MODGETDEVFSPATH_MI_LEN:
2262         case MODGETDEVFSPATH_MI:
2263                 /* Unprivileged */
2264                 return (0);
2265         case MODLOAD:
2266         case MODSETDEVPOLICY:
2267                 return (secpolicy_require_set(cr, PRIV_FULLSET, NULL,
2268                     KLPDARG_NONE));
2269         default:
2270                 return (secpolicy_sys_config(cr, B_FALSE));
2271         }
2272 }

2274 int
2275 secpolicy_console(const cred_t *cr)
2276 {
2277         return (PRIV_POLICY(cr, PRIV_SYS_DEVICES, B_FALSE, EPERM, NULL));
2278 }

2280 int
2281 secpolicy_power_mgmt(const cred_t *cr)
2282 {
2283         return (PRIV_POLICY(cr, PRIV_SYS_DEVICES, B_FALSE, EPERM, NULL));
2284 }

2286 /*
2287  * Simulate terminal input; another escalation of privileges avenue.
2288  */

2290 int
2291 secpolicy_sti(const cred_t *cr)
2292 {
2293         return (secpolicy_require_set(cr, PRIV_FULLSET, NULL, KLPDARG_NONE));
2294 }

2296 boolean_t
2297 secpolicy_net_reply_equal(const cred_t *cr)
2298 {
2299         return (PRIV_POLICY(cr, PRIV_SYS_CONFIG, B_FALSE, EPERM, NULL));
2300 }

2302 int
2303 secpolicy_swapctl(const cred_t *cr)
2304 {
2305         return (PRIV_POLICY(cr, PRIV_SYS_CONFIG, B_FALSE, EPERM, NULL));
2306 }

2308 int
2309 secpolicy_cpc_cpu(const cred_t *cr)
2310 {
2311         return (PRIV_POLICY(cr, PRIV_CPC_CPU, B_FALSE, EACCES, NULL));
2312 }

2314 /*
```



```
2315  * secpolicy_contract_identity
2316  *
2317  * Determine if the subject may set the process contract FMRI value
2318  */
2319 int
2320 secpolicy_contract_identity(const cred_t *cr)
2321 {
2322         return (PRIV_POLICY(cr, PRIV_CONTRACT_IDENTITY, B_FALSE, EPERM, NULL));
2323 }

2325 /*
2326  * secpolicy_contract_observer
2327  *
2328  * Determine if the subject may observe a specific contract's events.
2329  */
2330 int
2331 secpolicy_contract_observer(const cred_t *cr, struct contract *ct)
2332 {
2333         if (contract_owned(ct, cr, B_FALSE))
2334                 return (0);
2335         return (PRIV_POLICY(cr, PRIV_CONTRACT_OBSERVER, B_FALSE, EPERM, NULL));
2336 }

2338 /*
2339  * secpolicy_contract_observer_choice
2340  *
2341  * Determine if the subject may observe any contract's events.  Just
2342  * tests privilege and audits on success.
2343  */
2344 boolean_t
2345 secpolicy_contract_observer_choice(const cred_t *cr)
2346 {
2347         return (PRIV_POLICY_CHOICE(cr, PRIV_CONTRACT_OBSERVER, B_FALSE));
2348 }

2350 /*
2351  * secpolicy_contract_event
2352  *
2353  * Determine if the subject may request critical contract events or
2354  * reliable contract event delivery.
2355  */
2356 int
2357 secpolicy_contract_event(const cred_t *cr)
2358 {
2359         return (PRIV_POLICY(cr, PRIV_CONTRACT_EVENT, B_FALSE, EPERM, NULL));
2360 }

2362 /*
2363  * secpolicy_contract_event_choice
2364  *
2365  * Determine if the subject may retain contract events in its critical
2366  * set when a change in other terms would normally require a change in
2367  * the critical set.  Just tests privilege and audits on success.
2368  */
2369 boolean_t
2370 secpolicy_contract_event_choice(const cred_t *cr)
2371 {
2372         return (PRIV_POLICY_CHOICE(cr, PRIV_CONTRACT_EVENT, B_FALSE));
2373 }

2375 /*
2376  * secpolicy_gart_access
2377  *
2378  * Determine if the subject has sufficient priveleges to make ioctls to agpgart
2379  * device.
2380  */
```

```
2381 int
2382 secpolicy_gart_access(const cred_t *cr)
2383 {
2384         return (PRIV_POLICY(cr, PRIV_GRAPHICS_ACCESS, B_FALSE, EPERM, NULL));
2385 }

2387 /*
2388  * secpolicy_gart_map
2389  *
2390  * Determine if the subject has sufficient priveleges to map aperture range
2391  * through agpgart driver.
2392  */
2393 int
2394 secpolicy_gart_map(const cred_t *cr)
2395 {
2396         if (PRIV_POLICY_ONLY(cr, PRIV_GRAPHICS_ACCESS, B_FALSE)) {
2397                 return (PRIV_POLICY(cr, PRIV_GRAPHICS_ACCESS, B_FALSE, EPERM,
2398                     NULL));
2399         } else {
2400                 return (PRIV_POLICY(cr, PRIV_GRAPHICS_MAP, B_FALSE, EPERM,
2401                     NULL));
2402         }
2403 }

2405 /*
2406  * secpolicy_zinject
2407  *
2408  * Determine if the subject can inject faults in the ZFS fault injection
2409  * framework.  Requires all privileges.
2410  */
2411 int
2412 secpolicy_zinject(const cred_t *cr)
2413 {
2414         return (secpolicy_require_set(cr, PRIV_FULLSET, NULL, KLPDARG_NONE));
2415 }

2417 /*
2418  * secpolicy_zfs
2419  *
2420  * Determine if the subject has permission to manipulate ZFS datasets
2421  * (not pools).  Equivalent to the SYS_MOUNT privilege.
2422  */
2423 int
2424 secpolicy_zfs(const cred_t *cr)
2425 {
2426         return (PRIV_POLICY(cr, PRIV_SYS_MOUNT, B_FALSE, EPERM, NULL));
2427 }

2429 /*
2430  * secpolicy_idmap
2431  *
2432  * Determine if the calling process has permissions to register an SID
2433  * mapping daemon and allocate ephemeral IDs.
2434  */
2435 int
2436 secpolicy_idmap(const cred_t *cr)
2437 {
2438         return (PRIV_POLICY(cr, PRIV_FILE_SETID, B_TRUE, EPERM, NULL));
2439 }

2441 /*
2442  * secpolicy_ucode_update
2443  *
2444  * Determine if the subject has sufficient privilege to update microcode.
2445  */
2446 int
```



```
2447 secpolicy_ucode_update(const cred_t *scr)
2448 {
2449         return (PRIV_POLICY(scr, PRIV_ALL, B_FALSE, EPERM, NULL));
2450 }

2452 /*
2453  * secpolicy_sadopen
2454  *
2455  * Determine if the subject has sufficient privilege to access /dev/sad/admin.
2456  * /dev/sad/admin appear in global zone and exclusive-IP zones only.
2457  * In global zone, sys_config is required.
2458  * In exclusive-IP zones, sys_ip_config is required.
2459  * Note that sys_config is prohibited in non-global zones.
2460  */
2461 int
2462 secpolicy_sadopen(const cred_t *credp)
2463 {
2464         priv_set_t pset;

2466         priv_emptyset(&pset);

2468         if (crgetzoneid(credp) == GLOBAL_ZONEID)
2469                 priv_addset(&pset, PRIV_SYS_CONFIG);
2470         else
2471                 priv_addset(&pset, PRIV_SYS_IP_CONFIG);

2473         return (secpolicy_require_set(credp, &pset, "devpolicy", KLPDARG_NONE));
2474 }


2477 /*
2478  * Add privileges to a particular privilege set; this is called when the
2479  * current sets of privileges are not sufficient.  I.e., we should always
2480  * call the policy override functions from here.
2481  * What we are allowed to have is in the Observed Permitted set; so
2482  * we compute the difference between that and the newset.
2483  */
2484 int
2485 secpolicy_require_privs(const cred_t *cr, const priv_set_t *nset)
2486 {
2487         priv_set_t rqd;

2489         rqd = CR_OPPRIV(cr);

2491         priv_inverse(&rqd);
2492         priv_intersect(nset, &rqd);

2494         return (secpolicy_require_set(cr, &rqd, NULL, KLPDARG_NONE));
2495 }

2497 /*
2498  * secpolicy_smb
2499  *
2500  * Determine if the cred_t has PRIV_SYS_SMB privilege, indicating
2501  * that it has permission to access the smbsrv kernel driver.
2502  * PRIV_POLICY checks the privilege and audits the check.
2503  *
2504  * Returns:
2505  * 0       Driver access is allowed.
2506  * EPERM   Driver access is NOT permitted.
2507  */
2508 int
2509 secpolicy_smb(const cred_t *cr)
2510 {
2511         return (PRIV_POLICY(cr, PRIV_SYS_SMB, B_FALSE, EPERM, NULL));
2512 }
```

```
2514 /*
2515  * secpolicy_vscan
2516  *
2517  * Determine if cred_t has the necessary privileges to access a file
2518  * for virus scanning and update its extended system attributes.
2519  * PRIV_FILE_DAC_SEARCH, PRIV_FILE_DAC_READ - file access
2520  * PRIV_FILE_FLAG_SET - set extended system attributes
2521  *
2522  * PRIV_POLICY checks the privilege and audits the check.
2523  *
2524  * Returns:
2525  * 0      file access for virus scanning allowed.
2526  * EPERM  file access for virus scanning is NOT permitted.
2527  */
2528 int
2529 secpolicy_vscan(const cred_t *cr)
2530 {
2531         if ((PRIV_POLICY(cr, PRIV_FILE_DAC_SEARCH, B_FALSE, EPERM, NULL)) ||
2532             (PRIV_POLICY(cr, PRIV_FILE_DAC_READ, B_FALSE, EPERM, NULL)) ||
2533             (PRIV_POLICY(cr, PRIV_FILE_FLAG_SET, B_FALSE, EPERM, NULL))) {
2534                 return (EPERM);
2535         }
2536
2537         return (0);
2538 }
2539
2540 /*
2541  * secpolicy_smbfs_login
2542  *
2543  * Determines if the caller can add and delete the smbfs login
2544  * password in the the nsmb kernel module for the CIFS client.
2545  *
2546  * Returns:
2547  * 0       access is allowed.
2548  * EPERM   access is NOT allowed.
2549  */
2550 int
2551 secpolicy_smbfs_login(const cred_t *cr, uid_t uid)
2552 {
2553         uid_t cruid = crgetruid(cr);
2554
2555         if (cruid == uid)
2556                 return (0);
2557         return (PRIV_POLICY(cr, PRIV_PROC_OWNER, B_FALSE,
2558             EPERM, NULL));
2559 }
2560
2561 /*
2562  * secpolicy_xvm_control
2563  *
2564  * Determines if a caller can control the xVM hypervisor and/or running
2565  * domains (x86 specific).
2566  *
2567  * Returns:
2568  * 0       access is allowed.
2569  * EPERM   access is NOT allowed.
2570  */
2571 int
2572 secpolicy_xvm_control(const cred_t *cr)
2573 {
2574         if (PRIV_POLICY(cr, PRIV_XVM_CONTROL, B_FALSE, EPERM, NULL))
2575                 return (EPERM);
2576         return (0);
2577 }
```



```
2579 /*
2580  * secpolicy_ppp_config
2581  *
2582  * Determine if the subject has sufficient privileges to configure PPP and
2583  * PPP-related devices.
2584  */
2585 int
2586 secpolicy_ppp_config(const cred_t *cr)
2587 {
2588         if (PRIV_POLICY_ONLY(cr, PRIV_SYS_NET_CONFIG, B_FALSE))
2589                 return (secpolicy_net_config(cr, B_FALSE));
2590         return (PRIV_POLICY(cr, PRIV_SYS_PPP_CONFIG, B_FALSE, EPERM, NULL));
2591 }
```

```
**********************************************************
   20825 Wed May 27 19:49:30 2015
new/usr/src/uts/common/os/priv_defs
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
```

```
   1 /*
   2  * CDDL HEADER START
   3  *
   4  * The contents of this file are subject to the terms of the
   5  * Common Development and Distribution License (the "License").
   6  * You may not use this file except in compliance with the License.
   7  *
   8  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
   9  * or http://www.opensolaris.org/os/licensing.
  10  * See the License for the specific language governing permissions
  11  * and limitations under the License.
  12  *
  13  * When distributing Covered Code, include this CDDL HEADER in each
  14  * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  15  * If applicable, add the following below this CDDL HEADER, with the
  16  * fields enclosed by brackets "[]" replaced with your own identifying
  17  * information: Portions Copyright [yyyy] [name of copyright owner]
  18  *
  19  * CDDL HEADER END
  20  */
  21 /*
  22  * Copyright (c) 2003, 2010, Oracle and/or its affiliates. All rights reserved.
  23  * Copyright 2013, Joyent, Inc. All rights reserved.
  24  *
  25 INSERT COMMENT
  26  */

  28 #
  29 # Privileges can be added to this file at any location, not
  30 # necessarily at the end.  For patches, it is probably best to
  31 # add the new privilege at the end; for ordinary releases privileges
  32 # should be ordered alphabetically.
  33 #

  35 privilege PRIV_CONTRACT_EVENT

  37         Allows a process to request critical events without limitation.
  38         Allows a process to request reliable delivery of all events on
  39         any event queue.

  41 privilege PRIV_CONTRACT_IDENTITY

  43         Allows a process to set the service FMRI value of a process
  44         contract template.

  46 privilege PRIV_CONTRACT_OBSERVER

  48         Allows a process to observe contract events generated by
  49         contracts created and owned by users other than the process's
  50         effective user ID.
  51         Allows a process to open contract event endpoints belonging to
  52         contracts created and owned by users other than the process's
  53         effective user ID.

  55 privilege PRIV_CPC_CPU

  57         Allow a process to access per-CPU hardware performance counters.
```



```
  59 privilege PRIV_DTRACE_KERNEL

  61         Allows DTrace kernel-level tracing.

  63 privilege PRIV_DTRACE_PROC

  65         Allows DTrace process-level tracing.
  66         Allows process-level tracing probes to be placed and enabled in
  67         processes to which the user has permissions.

  69 privilege PRIV_DTRACE_USER

  71         Allows DTrace user-level tracing.
  72         Allows use of the syscall and profile DTrace providers to
  73         examine processes to which the user has permissions.

  75 privilege PRIV_FILE_CHOWN

  77         Allows a process to change a file's owner user ID.
  78         Allows a process to change a file's group ID to one other than
  79         the process' effective group ID or one of the process'
  80         supplemental group IDs.

  82 privilege PRIV_FILE_CHOWN_SELF

  84         Allows a process to give away its files; a process with this
  85         privilege will run as if {_POSIX_CHOWN_RESTRICTED} is not
  86         in effect.

  88 privilege PRIV_FILE_DAC_EXECUTE

  90         Allows a process to execute an executable file whose permission
  91         bits or ACL do not allow the process execute permission.

  93 privilege PRIV_FILE_DAC_READ

  95         Allows a process to read a file or directory whose permission
  96         bits or ACL do not allow the process read permission.

  98 privilege PRIV_FILE_DAC_SEARCH

 100         Allows a process to search a directory whose permission bits or
 101         ACL do not allow the process search permission.

 103 privilege PRIV_FILE_DAC_WRITE

 105         Allows a process to write a file or directory whose permission
 106         bits or ACL do not allow the process write permission.
 107         In order to write files owned by uid 0 in the absence of an
 108         effective uid of 0 ALL privileges are required.

 110 privilege PRIV_FILE_DOWNGRADE_SL

 112         Allows a process to set the sensitivity label of a file or
 113         directory to a sensitivity label that does not dominate the
 114         existing sensitivity label.
 115         This privilege is interpreted only if the system is configured
 116         with Trusted Extensions.

 118 privilege PRIV_FILE_FLAG_SET

 120         Allows a process to set immutable, nounlink or appendonly
 121         file attributes.

 123 basic privilege PRIV_FILE_LINK_ANY
```

```
125         Allows a process to create hardlinks to files owned by a uid
126         different from the process' effective uid.

128 privilege PRIV_FILE_OWNER

130         Allows a process which is not the owner of a file or directory
131         to perform the following operations that are normally permitted
132         only for the file owner: modify that file's access and
133         modification times; remove or rename a file or directory whose
134         parent directory has the ''save text image after execution''
135         (sticky) bit set; mount a ''namefs'' upon a file; modify
136         permission bits or ACL except for the set-uid and set-gid
137         bits.

139 basic privilege PRIV_FILE_READ

141         Allows a process to read objects in the filesystem.

143 privilege PRIV_FILE_SETID

145         Allows a process to change the ownership of a file or write to
146         a file without the set-user-ID and set-group-ID bits being
147         cleared.
148         Allows a process to set the set-group-ID bit on a file or
149         directory whose group is not the process' effective group or
150         one of the process' supplemental groups.
151         Allows a process to set the set-user-ID bit on a file with
152         different ownership in the presence of PRIV_FILE_OWNER.
153         Additional restrictions apply when creating or modifying a
154         set-uid 0 file.

156 privilege PRIV_FILE_UPGRADE_SL

158         Allows a process to set the sensitivity label of a file or
159         directory to a sensitivity label that dominates the existing
160         sensitivity label.
161         This privilege is interpreted only if the system is configured
162         with Trusted Extensions.

164 basic privilege PRIV_FILE_WRITE

166         Allows a process to modify objects in the filesystem.

168 privilege PRIV_GRAPHICS_ACCESS

170         Allows a process to make privileged ioctls to graphics devices.
171         Typically only xserver process needs to have this privilege.
172         A process with this privilege is also allowed to perform
173         privileged graphics device mappings.

175 privilege PRIV_GRAPHICS_MAP

177         Allows a process to perform privileged mappings through a
178         graphics device.

180 privilege PRIV_IPC_DAC_READ

182         Allows a process to read a System V IPC
183         Message Queue, Semaphore Set, or Shared Memory Segment whose
184         permission bits do not allow the process read permission.
185         Allows a process to read remote shared memory whose
186         permission bits do not allow the process read permission.

188 privilege PRIV_IPC_DAC_WRITE

190         Allows a process to write a System V IPC
```



```
191         Message Queue, Semaphore Set, or Shared Memory Segment whose
192         permission bits do not allow the process write permission.
193         Allows a process to read remote shared memory whose
194         permission bits do not allow the process write permission.
195         Additional restrictions apply if the owner of the object has uid 0
196         and the effective uid of the current process is not 0.

198 privilege PRIV_IPC_OWNER

200         Allows a process which is not the owner of a System
201         V IPC Message Queue, Semaphore Set, or Shared Memory Segment to
202         remove, change ownership of, or change permission bits of the
203         Message Queue, Semaphore Set, or Shared Memory Segment.
204         Additional restrictions apply if the owner of the object has uid 0
205         and the effective uid of the current process is not 0.

207 basic privilege PRIV_NET_ACCESS

209         Allows a process to open a TCP, UDP, SDP or SCTP network endpoint.

211 privilege PRIV_NET_BINDMLP

213         Allow a process to bind to a port that is configured as a
214         multi-level port(MLP) for the process's zone. This privilege
215         applies to both shared address and zone-specific address MLPs.
216         See tnzonecfg(4) from the Trusted Extensions manual pages for
217         information on configuring MLP ports.
218         This privilege is interpreted only if the system is configured
219         with Trusted Extensions.

221 privilege PRIV_NET_ICMPACCESS

223         Allows a process to send and receive ICMP packets.

225 privilege PRIV_NET_MAC_AWARE

227         Allows a process to set NET_MAC_AWARE process flag by using
228         setpflags(2). This privilege also allows a process to set
229         SO_MAC_EXEMPT socket option by using setsockopt(3SOCKET).
230         The NET_MAC_AWARE process flag and the SO_MAC_EXEMPT socket
231         option both allow a local process to communicate with an
232         unlabeled peer if the local process' label dominates the
233         peer's default label, or if the local process runs in the
234         global zone.
235         This privilege is interpreted only if the system is configured
236         with Trusted Extensions.

238 privilege PRIV_NET_MAC_IMPLICIT

240         Allows a process to set SO_MAC_IMPLICIT option by using
241         setsockopt(3SOCKET).  This allows a privileged process to
242         transmit implicitly-labeled packets to a peer.
243         This privilege is interpreted only if the system is configured
244         with Trusted Extensions.

246 privilege PRIV_NET_OBSERVABILITY

248         Allows a process to access /dev/lo0 and the devices in /dev/ipnet/
249         while not requiring them to need PRIV_NET_RAWACCESS.

251 privilege PRIV_NET_PRIVADDR

253         Allows a process to bind to a privileged port
254         number. The privilege port numbers are 1-1023 (the traditional
255         UNIX privileged ports) as well as those ports marked as
256         "udp/tcp_extra_priv_ports" with the exception of the ports
```

```
257         reserved for use by NFS.
259 privilege PRIV_NET_RAWACCESS
261         Allows a process to have direct access to the network layer.
263 unsafe privilege PRIV_PROC_AUDIT
265         Allows a process to generate audit records.
266         Allows a process to get its own audit pre-selection information.
268 privilege PRIV_PROC_CHROOT
270         Allows a process to change its root directory.
272 privilege PRIV_PROC_CLOCK_HIGHRES
274         Allows a process to use high resolution timers.
276 basic privilege PRIV_PROC_EXEC
278         Allows a process to call execve().
280 basic privilege PRIV_PROC_FORK
282         Allows a process to call fork1()/forkall()/vfork()
284 basic privilege PRIV_PROC_INFO
286         Allows a process to examine the status of processes other
287         than those it can send signals to.  Processes which cannot
288         be examined cannot be seen in /proc and appear not to exist.
290 privilege PRIV_PROC_LOCK_MEMORY
292         Allows a process to lock pages in physical memory.
294 privilege PRIV_PROC_OWNER
296         Allows a process to send signals to other processes, inspect
297         and modify process state to other processes regardless of
298         ownership.  When modifying another process, additional
299         restrictions apply:  the effective privilege set of the
300         attaching process must be a superset of the target process'
301         effective, permitted and inheritable sets; the limit set must
302         be a superset of the target's limit set; if the target process
303         has any uid set to 0 all privilege must be asserted unless the
304         effective uid is 0.
305         Allows a process to bind arbitrary processes to CPUs.
307 privilege PRIV_PROC_PRIOUP
309         Allows a process to elevate its priority above its current level.
311 privilege PRIV_PROC_PRIOCNTL
313         Allows all that PRIV_PROC_PRIOUP allows.
314         Allows a process to change its scheduling class to any scheduling class,
315         including the RT class.
317 privilege PRIV_PROC_SECFLAGS
319         Allows a process to manipulate the secflags of processes (subject to,
320         additionally, the ability to signal that process)
322 #endif /* ! codereview */
```



```
323 basic privilege PRIV_PROC_SESSION
325         Allows a process to send signals or trace processes outside its
326         session.
328 unsafe privilege PRIV_PROC_SETID
330         Allows a process to set its uids at will.
331         Assuming uid 0 requires all privileges to be asserted.
333 privilege PRIV_PROC_TASKID
335         Allows a process to assign a new task ID to the calling process.
337 privilege PRIV_PROC_ZONE
339         Allows a process to trace or send signals to processes in
340         other zones.
342 privilege PRIV_SYS_ACCT
344         Allows a process to enable and disable and manage accounting through
345         acct(2), getacct(2), putacct(2) and wracct(2).
347 privilege PRIV_SYS_ADMIN
349         Allows a process to perform system administration tasks such
350         as setting node and domain name and specifying nscd and coreadm
351         settings.
353 privilege PRIV_SYS_AUDIT
355         Allows a process to start the (kernel) audit daemon.
356         Allows a process to view and set audit state (audit user ID,
357         audit terminal ID, audit sessions ID, audit pre-selection mask).
358         Allows a process to turn off and on auditing.
359         Allows a process to configure the audit parameters (cache and
360         queue sizes, event to class mappings, policy options).
362 privilege PRIV_SYS_CONFIG
364         Allows a process to perform various system configuration tasks.
365         Allows a process to add and remove swap devices; when adding a swap
366         device, a process must also have sufficient privileges to read from
367         and write to the swap device.
369 privilege PRIV_SYS_DEVICES
371         Allows a process to successfully call a kernel module that
372         calls the kernel drv_priv(9F) function to check for allowed
373         access.
374         Allows a process to open the real console device directly.
375         Allows a process to open devices that have been exclusively opened.
377 privilege PRIV_SYS_IPC_CONFIG
379         Allows a process to increase the size of a System V IPC Message
380         Queue buffer.
382 privilege PRIV_SYS_LINKDIR
384         Allows a process to unlink and link directories.
386 privilege PRIV_SYS_MOUNT
388         Allows filesystem specific administrative procedures, such as
```

```
389         filesystem configuration ioctls, quota calls and creation/deletion
390         of snapshots.
391         Allows a process to mount and unmount filesystems which would
392         otherwise be restricted (i.e., most filesystems except
393         namefs).
394         A process performing a mount operation needs to have
395         appropriate access to the device being mounted (read-write for
396         "rw" mounts, read for "ro" mounts).
397         A process performing any of the aforementioned
398         filesystem operations needs to have read/write/owner
399         access to the mount point.
400         Only regular files and directories can serve as mount points
401         for processes which do not have all zone privileges asserted.
402         Unless a process has all zone privileges, the mount(2)
403         system call will force the "nosuid" and "restrict" options, the
404         latter only for autofs mountpoints.
405         Regardless of privileges, a process running in a non-global zone may
406         only control mounts performed from within said zone.
407         Outside the global zone, the "nodevices" option is always forced.

409 privilege PRIV_SYS_IPTUN_CONFIG

411         Allows a process to configure IP tunnel links.

413 privilege PRIV_SYS_DL_CONFIG

415         Allows a process to configure all classes of datalinks, including
416         configuration allowed by PRIV_SYS_IPTUN_CONFIG.

418 privilege PRIV_SYS_IP_CONFIG

420         Allows a process to configure a system's IP interfaces and routes.
421         Allows a process to configure network parameters using ndd.
422         Allows a process access to otherwise restricted information using ndd.
423         Allows a process to configure IPsec.
424         Allows a process to pop anchored STREAMs modules with matching zoneid.

426 privilege PRIV_SYS_NET_CONFIG

428         Allows all that PRIV_SYS_IP_CONFIG, PRIV_SYS_DL_CONFIG, and
429         PRIV_SYS_PPP_CONFIG allow.
430         Allows a process to push the rpcmod STREAMs module.
431         Allows a process to INSERT/REMOVE STREAMs modules on locations other
432         than the top of the module stack.

434 privilege PRIV_SYS_NFS

436         Allows a process to perform Sun private NFS specific system calls.
437         Allows a process to bind to ports reserved by NFS: ports 2049 (nfs)
438         and port 4045 (lockd).

440 privilege PRIV_SYS_PPP_CONFIG

442         Allows a process to create and destroy PPP (sppp) interfaces.
443         Allows a process to configure PPP tunnels (sppptun).

445 privilege PRIV_SYS_RES_BIND

447         Allows a process to bind processes to processor sets.

449 privilege PRIV_SYS_RES_CONFIG

451         Allows all that PRIV_SYS_RES_BIND allows.
452         Allows a process to create and delete processor sets, assign
453         CPUs to processor sets and override the PSET_NOESCAPE property.
454         Allows a process to change the operational status of CPUs in
```



```
455         the system using p_online(2).
456         Allows a process to configure resource pools and to bind
457         processes to pools

459 unsafe privilege PRIV_SYS_RESOURCE

461         Allows a process to modify the resource limits specified
462         by setrlimit(2) and setrctl(2) without restriction.
463         Allows a process to exceed the per-user maximum number of
464         processes.
465         Allows a process to extend or create files on a filesystem that
466         has less than minfree space in reserve.

468 privilege PRIV_SYS_SMB

470         Allows a process to access the Sun private SMB kernel module.
471         Allows a process to bind to ports reserved by NetBIOS and SMB:
472         ports 137 (NBNS), 138 (NetBIOS Datagram Service), 139 (NetBIOS
473         Session Service and SMB-over-NBT) and 445 (SMB-over-TCP).

475 privilege PRIV_SYS_SUSER_COMPAT

477         Allows a process to successfully call a third party loadable module
478         that calls the kernel suser() function to check for allowed access.
479         This privilege exists only for third party loadable module
480         compatibility and is not used by Solaris proper.

482 privilege PRIV_SYS_TIME

484         Allows a process to manipulate system time using any of the
485         appropriate system calls: stime, adjtime, ntp_adjtime and
486         the IA specific RTC calls.

488 privilege PRIV_SYS_TRANS_LABEL

490         Allows a process to translate labels that are not dominated
491         by the process' sensitivity label to and from an external
492         string form.
493         This privilege is interpreted only if the system is configured
494         with Trusted Extensions.

496 privilege PRIV_VIRT_MANAGE

498         Allows a process to manage virtualized environments such as
499         xVM(5).

501 privilege PRIV_WIN_COLORMAP

503         Allows a process to override colormap restrictions.
504         Allows a process to install or remove colormaps.
505         Allows a process to retrieve colormap cell entries allocated
506         by other processes.
507         This privilege is interpreted only if the system is configured
508         with Trusted Extensions.

510 privilege PRIV_WIN_CONFIG

512         Allows a process to configure or destroy resources that are
513         permanently retained by the X server.
514         Allows a process to use SetScreenSaver to set the screen
515         saver timeout value.
516         Allows a process to use ChangeHosts to modify the display
517         access control list.
518         Allows a process to use GrabServer.
519         Allows a process to use the SetCloseDownMode request which
520         may retain window, pixmap, colormap, property, cursor, font,
```

```
521         or graphic context resources.
522         This privilege is interpreted only if the system is configured
523         with Trusted Extensions.

525 privilege PRIV_WIN_DAC_READ

527         Allows a process to read from a window resource that it does
528         not own (has a different user ID).
529         This privilege is interpreted only if the system is configured
530         with Trusted Extensions.

532 privilege PRIV_WIN_DAC_WRITE

534         Allows a process to write to or create a window resource that
535         it does not own (has a different user ID). A newly created
536         window property is created with the window's user ID.
537         This privilege is interpreted only if the system is configured
538         with Trusted Extensions.

540 privilege PRIV_WIN_DEVICES

542         Allows a process to perform operations on window input devices.
543         Allows a process to get and set keyboard and pointer controls.
544         Allows a process to modify pointer button and key mappings.
545         This privilege is interpreted only if the system is configured
546         with Trusted Extensions.

548 privilege PRIV_WIN_DGA

550         Allows a process to use the direct graphics access (DGA) X protocol
551         extensions. Direct process access to the frame buffer is still
552         required. Thus the process must have MAC and DAC privileges that
553         allow access to the frame buffer, or the frame buffer must be
554         allocated to the process.
555         This privilege is interpreted only if the system is configured
556         with Trusted Extensions.

558 privilege PRIV_WIN_DOWNGRADE_SL

560         Allows a process to set the sensitivity label of a window resource
561         to a sensitivity label that does not dominate the existing
562         sensitivity label.
563         This privilege is interpreted only if the system is configured
564         with Trusted Extensions.

566 privilege PRIV_WIN_FONTPATH

568         Allows a process to set a font path.
569         This privilege is interpreted only if the system is configured
570         with Trusted Extensions.

572 privilege PRIV_WIN_MAC_READ

574         Allows a process to read from a window resource whose sensitivity
575         label is not equal to the process sensitivity label.
576         This privilege is interpreted only if the system is configured
577         with Trusted Extensions.

579 privilege PRIV_WIN_MAC_WRITE

581         Allows a process to create a window resource whose sensitivity
582         label is not equal to the process sensitivity label.
583         A newly created window property is created with the window's
584         sensitivity label.
585         This privilege is interpreted only if the system is configured
586         with Trusted Extensions.
```



```
588 privilege PRIV_WIN_SELECTION

590         Allows a process to request inter-window data moves without the
591         intervention of the selection confirmer.
592         This privilege is interpreted only if the system is configured
593         with Trusted Extensions.

595 privilege PRIV_WIN_UPGRADE_SL

597         Allows a process to set the sensitivity label of a window
598         resource to a sensitivity label that dominates the existing
599         sensitivity label.
600         This privilege is interpreted only if the system is configured
601         with Trusted Extensions.

603 privilege PRIV_XVM_CONTROL

605         Allows a process access to the xVM(5) control devices for
606         managing guest domains and the hypervisor. This privilege is
607         used only if booted into xVM on x86 platforms.

609 set PRIV_EFFECTIVE

611         Set of privileges currently in effect.

613 set PRIV_INHERITABLE

615         Set of privileges that comes into effect on exec.

617 set PRIV_PERMITTED

619         Set of privileges that can be put into the effective set without
620         restriction.

622 set PRIV_LIMIT

624         Set of privileges that determines the absolute upper bound of
625         privileges this process and its off-spring can obtain.
```

```
**********************************************************
    4354 Wed May 27 19:49:31 2015
new/usr/src/uts/common/os/proc.c
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
______unchanged_portion_omitted_

163 boolean_t
164 secflag_enabled(proc_t *p, uint_t flag)
165 {
166         return ((p->p_secflags.psf_effective & flag) != 0);
167 }

169 void
170 secflag_set(proc_t *p, uint_t flag)
171 {
172         p->p_secflags.psf_inherit = flag;
173 }

175 void
176 secflag_enable(proc_t *p, uint_t flag) {
177         p->p_secflags.psf_inherit |= flag;
178 }

180 void
181 secflag_disable(proc_t *p, uint_t flag) {
182         p->p_secflags.psf_inherit &= ~flag;
183 }

185 void
186 secflag_promote(proc_t *p) {
187         p->p_secflags.psf_effective = p->p_secflags.psf_inherit;
188 }
189 #endif /* ! codereview */
```

```
**********************************************************
   46404 Wed May 27 19:49:31 2015
new/usr/src/uts/common/os/sysent.c
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright (c) 1988, 2010, Oracle and/or its affiliates. All rights reserved.
 * Copyright 2012 Milan Jurik. All rights reserved.
 * Copyright (c) 2013, OmniTI Computer Consulting, Inc. All rights reserved.
 * Copyright (c) 2015, Joyent, Inc.
 */

/*      Copyright (c) 1984, 1986, 1987, 1988, 1989 AT&T */
/*        All Rights Reserved   */

#include <sys/param.h>
#include <sys/types.h>
#include <sys/systm.h>
#include <sys/systrace.h>
#include <sys/procfs.h>
#include <sys/mman.h>
#include <sys/int_types.h>
#include <c2/audit.h>
#include <sys/stat.h>
#include <sys/times.h>
#include <sys/statfs.h>
#include <sys/stropts.h>
#include <sys/statvfs.h>
#include <sys/utsname.h>
#include <sys/timex.h>
#include <sys/socket.h>
#include <sys/sendfile.h>

struct hrtsysa;
struct mmaplf32a;

/*
 * This table is the switch used to transfer to the appropriate
 * routine for processing a system call.  Each row contains the
 * number of arguments expected, a switch that tells systrap()
 * in trap.c whether a setjmp() is not necessary, and a pointer
 * to the routine.
```



```
 */

int     access(char *, int);
int     alarm(int);
int     auditsys(struct auditcalls *, rval_t *);
int64_t brandsys(int, uintptr_t, uintptr_t, uintptr_t, uintptr_t, uintptr_t,
    uintptr_t);
int     brk(caddr_t);
int     chdir(char *);
int     chmod(char *, int);
int     chown(char *, uid_t, gid_t);
int     chroot(char *);
int     cladm(int, int, void *);
int     close(int);
int     exece(const char *, const char **, const char **);
int     faccessat(int, char *, int, int);
int     fchmodat(int, char *, int, int);
int     fchownat(int, char *, uid_t, gid_t, int);
int     fcntl(int, int, intptr_t);
int64_t vfork();
int64_t forksys(int, int);
int     fstat(int, struct stat *);
int     fdsync(int, int);
int64_t getgid();
int     ucredsys(int, int, void *);
int64_t getpid();
int64_t getuid();
time_t  gtime();
int     getloadavg(int *, int);
int     rusagesys(int, void *, void *, void *, void *);
int     getpagesizes(int, size_t *, int);
int     gtty(int, intptr_t);
#if defined(__i386) || defined(__amd64)
int     hrtsys(struct hrtsysa *, rval_t *);
#endif /* __i386 || __amd64 */
int     ioctl(int, int, intptr_t);
int     kill();
int     labelsys(int, void *, void *, void *, void *, void *);
int     link(char *, char *);
int     linkat(int, char *, int, char *, int);
off32_t lseek32(int32_t, off32_t, int32_t);
off_t   lseek64(int, off_t, int);
int     lgrpsys(int, long, void *);
int     mmapobjsys(int, uint_t, mmapobj_result_t *, uint_t *, void *);
int     mknod(char *, mode_t, dev_t);
int     mknodat(int, char *, mode_t, dev_t);
int     mount(long *, rval_t *);
int     nice(int);
int     nullsys();
int     open(char *, int, int);
int     openat(int, char *, int, int);
int     pause();
long    pcsample(void *, long);
int     privsys(int, priv_op_t, priv_ptype_t, void *, size_t, int);
int     profil(unsigned short *, size_t, ulong_t, uint_t);
ssize_t pread(int, void *, size_t, off_t);
int     psecflags();
#endif /* ! codereview */
ssize_t pwrite(int, void *, size_t, off_t);
ssize_t read(int, void *, size_t);
int     rename(char *, char *);
int     renameat(int, char *, int, char *);
void    rexit(int);
int     semsys();
int     setgid(gid_t);
int     setpgrp(int, int, int);
```

```
125 int     setuid(uid_t);
126 uintptr_t       shmsys();
127 uint64_t        sidsys(int, int, int, int);
128 int     sigprocmask(int, sigset_t *, sigset_t *);
129 int     sigsuspend(sigset_t);
130 int     sigaltstack(struct sigaltstack *, struct sigaltstack *);
131 int     sigaction(int, struct sigaction *, struct sigaction *);
132 int     sigpending(int, sigset_t *);
133 int     sigresend(int, siginfo_t *, sigset_t *);
134 int     sigtimedwait(sigset_t *, siginfo_t *, timespec_t *);
135 int     getsetcontext(int, void *);
136 int     stat(char *, struct stat *);
137 int     fstatat(int, char *, struct stat *, int);
138 int     stime(time_t);
139 int     stty(int, intptr_t);
140 int     syssync();
141 int     sysacct(char *);
142 clock_t times(struct tms *);
143 long    ulimit(int, long);
144 int     getrlimit32(int, struct rlimit32 *);
145 int     setrlimit32(int, struct rlimit32 *);
146 int     umask(int);
147 int     umount2(char *, int);
148 int     unlink(char *);
149 int     unlinkat(int, char *, int);
150 int     utimesys(int, uintptr_t, uintptr_t, uintptr_t, uintptr_t);
151 int64_t utssys32(void *, int, int, void *);
152 int64_t utssys64(void *, long, int, void *);
153 int     uucopy(const void *, void *, size_t);
154 ssize_t uucopystr(const char *, char *, size_t);
155 ssize_t write(int, void *, size_t);
156 ssize_t readv(int, struct iovec *, int);
157 ssize_t writev(int, struct iovec *, int);
158 ssize_t preadv(int, struct iovec *, int, off_t, off_t);
159 ssize_t pwritev(int, struct iovec *, int, off_t, off_t);
160 int     syslwp_park(int, uintptr_t, uintptr_t);
161 int     rmdir(char *);
162 int     mkdir(char *, int);
163 int     mkdirat(int, char *, int);
164 int     getdents32(int, void *, size_t);
165 int     statfs32(char *, struct statfs32 *, int32_t, int32_t);
166 int     fstatfs32(int32_t, struct statfs32 *, int32_t, int32_t);
167 int     sysfs(int, long, long);
168 int     getmsg(int, struct strbuf *, struct strbuf *, int *);
169 int     pollsys(pollfd_t *, nfds_t, timespec_t *, sigset_t *);
170 int     putmsg(int, struct strbuf *, struct strbuf *, int);
171 int     uadmin();
172 int     lstat(char *, struct stat *);
173 int     symlink(char *, char *);
174 int     symlinkat(char *, int, char *);
175 ssize_t readlink(char *, char *, size_t);
176 ssize_t readlinkat(int, char *, char *, size_t);
177 int     resolvepath(char *, char *, size_t);
178 int     setgroups(int, gid_t *);
179 int     getgroups(int, gid_t *);
180 int     fchdir(int);
181 int     fchown(int, uid_t, uid_t);
182 int     fchmod(int, int);
183 int     getcwd(char *, size_t);
184 int     statvfs(char *, struct statvfs *);
185 int     fstatvfs(int, struct statvfs *);
186 offset_t llseek32(int32_t, uint32_t, uint32_t, int);

188 #if (defined(__i386) && !defined(__amd64)) || defined(__i386_COMPAT)
189 int     sysi86(short, uintptr_t, uintptr_t, uintptr_t);
190 #endif
```



```
192 int     acl(const char *, int, int, void *);
193 int     facl(int, int, int, void *);
194 long    priocntlsys(int, procset_t *, int, caddr_t, caddr_t);
195 int     waitsys(idtype_t, id_t, siginfo_t *, int);
196 int     sigsendsys(procset_t *, int);
197 int     mincore(caddr_t, size_t, char *);
198 caddr_t smmap64(caddr_t, size_t, int, int, int, off_t);
199 caddr_t smmap32(caddr32_t, size32_t, int, int, int, off32_t);
200 int     smmaplf32(struct mmaplf32a *, rval_t *);
201 int     mprotect(caddr_t, size_t, int);
202 int     munmap(caddr_t, size_t);
203 int     uname(struct utsname *);
204 int     lchown(char *, uid_t, gid_t);
205 int     getpmsg(int, struct strbuf *, struct strbuf *, int *, int *);
206 int     putpmsg(int, struct strbuf *, struct strbuf *, int, int);
207 int     memcntl(caddr_t, size_t, int, caddr_t, int, int);
208 long    sysconfig(int);
209 int     adjtime(struct timeval *, struct timeval *);
210 long    systeminfo(int, char *, long);
211 int     setegid(gid_t);
212 int     seteuid(uid_t);

214 int     setreuid(uid_t, uid_t);
215 int     setregid(gid_t, gid_t);
216 int     install_utrap(utrap_entry_t type, utrap_handler_t, utrap_handler_t *);
217 #ifdef __sparc
218 int     sparc_utrap_install(utrap_entry_t type, utrap_handler_t,
219     utrap_handler_t, utrap_handler_t *, utrap_handler_t *);
220 #endif

222 int     syslwp_create(ucontext_t *, int, id_t *);
223 void    syslwp_exit();
224 int     syslwp_suspend(id_t);
225 int     syslwp_continue(id_t);
226 int     syslwp_private(int, int, uintptr_t);
227 int     lwp_detach(id_t);
228 int     lwp_info(timestruc_t *);
229 int     lwp_kill(id_t, int);
230 int     lwp_self();
231 int64_t lwp_sigmask(int, uint_t, uint_t, uint_t, uint_t);
232 int     yield();
233 int     lwp_wait(id_t, id_t *);
234 int     lwp_mutex_timedlock(lwp_mutex_t *, timespec_t *, uintptr_t);
235 int     lwp_mutex_wakeup(lwp_mutex_t *, int);
236 int     lwp_mutex_unlock(lwp_mutex_t *);
237 int     lwp_mutex_trylock(lwp_mutex_t *, uintptr_t);
238 int     lwp_mutex_register(lwp_mutex_t *, caddr_t);
239 int     lwp_rwlock_sys(int, lwp_rwlock_t *, timespec_t *);
240 int     lwp_sema_post(lwp_sema_t *);
241 int     lwp_sema_timedwait(lwp_sema_t *, timespec_t *, int);
242 int     lwp_sema_trywait(lwp_sema_t *);
243 int     lwp_cond_wait(lwp_cond_t *, lwp_mutex_t *, timespec_t *, int);
244 int     lwp_cond_signal(lwp_cond_t *);
245 int     lwp_cond_broadcast(lwp_cond_t *);
246 caddr_t schedctl();

248 long    pathconf(char *, int);
249 long    fpathconf(int, int);
250 int     processor_bind(idtype_t, id_t, processorid_t, processorid_t *);
251 int     processor_info(processorid_t, processor_info_t *);
252 int     p_online(processorid_t, int);

254 /*
255  *      POSIX .4 system calls *
256  */
```

```
257 int     clock_gettime(clockid_t, timespec_t *);
258 int     clock_settime(clockid_t, timespec_t *);
259 int     clock_getres(clockid_t, timespec_t *);
260 int     timer_create(clockid_t, struct sigevent *, timer_t *);
261 int     timer_delete(timer_t);
262 int     timer_settime(timer_t, int, itimerspec_t *, itimerspec_t *);
263 int     timer_gettime(timer_t, itimerspec_t *);
264 int     timer_getoverrun(timer_t);
265 int     nanosleep(timespec_t *, timespec_t *);
266 int     sigqueue(pid_t, int, void *, int, int);
267 int     signotify(int, siginfo_t *, signotify_id_t *);

269 int     getdents64(int, void *, size_t);
270 int     stat64(char *, struct stat64 *);
271 int     lstat64(char *, struct stat64 *);
272 int     fstatat64(int, char *, struct stat64 *, int);
273 int     fstat64(int, struct stat64 *);
274 int     statvfs64(char *, struct statvfs64 *);
275 int     fstatvfs64(int, struct statvfs64 *);
276 int     setrlimit64(int, struct rlimit64 *);
277 int     getrlimit64(int, struct rlimit64 *);
278 int     pread64(int, void *, size32_t, uint32_t, uint32_t);
279 int     pwrite64(int, void *, size32_t, uint32_t, uint32_t);
280 int     open64(char *, int, int);
281 int     openat64(int, char *, int, int);

283 /*
284  * NTP syscalls
285  */

287 int ntp_gettime(struct ntptimeval *);
288 int ntp_adjtime(struct timex *);

290 /*
291  *      ++++++++++++++++++++++++
292  *      ++  SunOS4.1 Buyback  ++
293  *      ++++++++++++++++++++++++
294  *
295  *      fchroot, vhangup, gettimeofday
296  */

298 int     fchroot(int);
299 int     vhangup();
300 int     gettimeofday(struct timeval *);
301 int     getitimer(uint_t, struct itimerval *);
302 int     setitimer(uint_t, struct itimerval *, struct itimerval *);

304 int     corectl(int, uintptr_t, uintptr_t, uintptr_t);
305 int     modctl(int, uintptr_t, uintptr_t, uintptr_t, uintptr_t, uintptr_t);
306 int64_t loadable_syscall();
307 int64_t indir();

309 long    tasksys(int, projid_t, uint_t, void *, size_t);
310 long    rctlsys(int, char *, void *, void *, size_t, int);

312 long    zone();

314 int     so_socket(int, int, int, char *, int);
315 int     so_socketpair(int[2]);
316 int     bind(int, struct sockaddr *, socklen_t, int);
317 int     listen(int, int, int);
318 int     accept(int, struct sockaddr *, socklen_t *, int, int);
319 int     connect(int, struct sockaddr *, socklen_t, int);
320 int     shutdown(int, int, int);
321 ssize_t recv(int, void *, size_t, int);
322 ssize_t recvfrom(int, void *, size_t, int, struct sockaddr *, socklen_t *);
```



```
323 ssize_t recvmsg(int, struct nmsghdr *, int);
324 ssize_t send(int, void *, size_t, int);
325 ssize_t sendmsg(int, struct nmsghdr *, int);
326 ssize_t sendto(int, void *, size_t, int, struct sockaddr *, socklen_t);
327 int     getpeername(int, struct sockaddr *, socklen_t *, int);
328 int     getsockname(int, struct sockaddr *, socklen_t *, int);
329 int     getsockopt(int, int, int, void *, socklen_t *, int);
330 int     setsockopt(int, int, int, void *, socklen_t *, int);
331 int     sockconfig(int, void *, void *, void *, void *);
332 ssize_t sendfilev(int, int, const struct sendfilevec *, int, size_t *);
333 int     getrandom(void *, size_t, int);

335 typedef int64_t (*llfcn_t)();   /* for casting one-word returns */

337 /*
338  * Sysent initialization macros.
339  *      These take the name string of the system call even though that isn't
340  *      currently used in the sysent entry.  This might be useful someday.
341  *
342  * Initialization macro for system calls which take their args in the C style.
343  * These system calls return the longlong_t return value and must call
344  * set_errno() to return an error.  For SPARC, narg must be at most six.
345  * For more args, use the SYSENT_AP() routine.
346  *
347  * We are able to return two distinct values to userland via the rval_t.
348  * At this time, that corresponds to one 64-bit quantity, or two 32-bit
349  * quantities.  The kernel does not currently need to return two 64-bit
350  * values, or one 128 bit value(!), but we may do one day, so the calling
351  * sequence between userland and the kernel should permit it.
352  *
353  * The interpretation of rval_t is provided by the sy_flags field
354  * which is used to determine how to arrange the results in registers
355  * (or on the stack) for return userland.
356  */
357         /* returns a 64-bit quantity for both ABIs */
358 #define SYSENT_C(name, call, narg)      \
359         { (narg), SE_64RVAL, NULL, NULL, (llfcn_t)(call) }

361         /* returns one 32-bit value for both ABIs: r_val1 */
362 #define SYSENT_CI(name, call, narg)     \
363         { (narg), SE_32RVAL1, NULL, NULL, (llfcn_t)(call) }

365         /* returns 2 32-bit values: r_val1 & r_val2 */
366 #define SYSENT_2CI(name, call, narg)    \
367         { (narg), SE_32RVAL1|SE_32RVAL2, NULL, NULL, (llfcn_t)(call) }

369 /*
370  * Initialization macro for system calls which take their args in the standard
371  * Unix style of a pointer to the arg structure and a pointer to the rval_t.
372  *
373  * Deprecated wherever possible (slower on some architectures, and trickier
374  * to maintain two flavours).
375  */
376 #define SYSENT_AP(name, call, narg)     \
377         { (narg), SE_64RVAL, (call), NULL, syscall_ap }

379 /*
380  * Conditional constructors to build the tables without #ifdef clutter
381  */
382 #if defined(_LP64)
383 #define IF_LP64(true, false)    true
384 #else
385 #define IF_LP64(true, false)    false
386 #endif

388 #if defined(__sparc)
```

```
389 #define IF_sparc(true, false)   true
390 #else
391 #define IF_sparc(true, false)   false
392 #endif

394 #if defined(__i386) && !defined(__amd64)
395 #define IF_i386(true, false)    true
396 #else
397 #define IF_i386(true, false)    false
398 #endif

400 #if defined(__i386) || defined(__amd64)
401 #define IF_x86(true, false)     true
402 #else
403 #define IF_x86(true, false)     false
404 #endif

406 #if (defined(__i386) && !defined(__amd64)) || defined(__i386_COMPAT)
407 #define IF_386_ABI(true, false) true
408 #else
409 #define IF_386_ABI(true, false) false
410 #endif

412 /*
413  * Define system calls that return a native 'long' quantity i.e. a 32-bit
414  * or 64-bit integer - depending on how the kernel is itself compiled
415  * e.g. read(2) returns 'ssize_t' in the kernel and in userland.
416  */
417 #define SYSENT_CL(name, call, narg)     \
418         IF_LP64(SYSENT_C(name, call, narg), SYSENT_CI(name, call, narg))

420 /*
421  * Initialization macro for loadable native system calls.
422  */
423 #define SYSENT_LOADABLE()       \
424         { 0, SE_LOADABLE, (int (*)())nosys, NULL, loadable_syscall }

426 /*
427  * Initialization macro for loadable 32-bit compatibility system calls.
428  */
429 #define SYSENT_LOADABLE32()     SYSENT_LOADABLE()

431 #define SYSENT_NOSYS()          SYSENT_C("nosys", nosys, 0)

433 struct sysent nosys_ent = SYSENT_NOSYS();

435 /*
436  * Native sysent table.
437  */
438 struct sysent sysent[NSYSCALL] =
439 {
440         /*  0 */ IF_LP64(
441                         SYSENT_NOSYS(),
442                         SYSENT_C("indir",       indir,          1)),
443         /*  1 */ SYSENT_CI("exit",              rexit,          1),
444         /*  2 */ SYSENT_CI("psecflags",         psecflags,      3),
115         /*  2 */ SYSENT_LOADABLE(),                     /* (was forkall) */
445         /*  3 */ SYSENT_CL("read",              read,           3),
446         /*  4 */ SYSENT_CL("write",             write,          3),
447         /*  5 */ SYSENT_CI("open",              open,           3),
448         /*  6 */ SYSENT_CI("close",             close,          1),
449         /*  7 */ SYSENT_CI("linkat",            linkat,         5),
450         /*  8 */ SYSENT_LOADABLE(),                     /* (was creat) */
451         /*  9 */ SYSENT_CI("link",              link,           2),
452         /* 10 */ SYSENT_CI("unlink",            unlink,         1),
453         /* 11 */ SYSENT_CI("symlinkat",         symlinkat,      3),
```



```
454         /* 12 */ SYSENT_CI("chdir",             chdir,          1),
455         /* 13 */ SYSENT_CL("time",              gtime,          0),
456         /* 14 */ SYSENT_CI("mknod",             mknod,          3),
457         /* 15 */ SYSENT_CI("chmod",             chmod,          2),
458         /* 16 */ SYSENT_CI("chown",             chown,          3),
459         /* 17 */ SYSENT_CI("brk",               brk,            1),
460         /* 18 */ SYSENT_CI("stat",              stat,           2),
461         /* 19 */ IF_LP64(
462                         SYSENT_CL("lseek",      lseek64,        3),
463                         SYSENT_CL("lseek",      lseek32,        3)),
464         /* 20 */ SYSENT_2CI("getpid",           getpid,         0),
465         /* 21 */ SYSENT_AP("mount",             mount,          8),
466         /* 22 */ SYSENT_CL("readlinkat",        readlinkat,     4),
467         /* 23 */ SYSENT_CI("setuid",            setuid,         1),
468         /* 24 */ SYSENT_2CI("getuid",           getuid,         0),
469         /* 25 */ SYSENT_CI("stime",             stime,          1),
470         /* 26 */ SYSENT_CL("pcsample",          pcsample,       2),
471         /* 27 */ SYSENT_CI("alarm",             alarm,          1),
472         /* 28 */ SYSENT_CI("fstat",             fstat,          2),
473         /* 29 */ SYSENT_CI("pause",             pause,          0),
474         /* 30 */ SYSENT_LOADABLE(),                    /* (was utime) */
475         /* 31 */ SYSENT_CI("stty",              stty,           2),
476         /* 32 */ SYSENT_CI("gtty",              gtty,           2),
477         /* 33 */ SYSENT_CI("access",            access,         2),
478         /* 34 */ SYSENT_CI("nice",              nice,           1),
479         /* 35 */ IF_LP64(
480                         SYSENT_NOSYS(),
481                         SYSENT_CI("statfs",     statfs32,       4)),
482         /* 36 */ SYSENT_CI("sync",              syssync,        0),
483         /* 37 */ SYSENT_CI("kill",              kill,           2),
484         /* 38 */ IF_LP64(
485                         SYSENT_NOSYS(),
486                         SYSENT_CI("fstatfs",    fstatfs32,      4)),
487         /* 39 */ SYSENT_CI("setpgrp",           setpgrp,        3),
488         /* 40 */ SYSENT_CI("uucopystr",         uucopystr,      3),
489         /* 41 */ SYSENT_LOADABLE(),                    /* (was dup) */
490         /* 42 */ SYSENT_LOADABLE(),                    /* pipe */
491         /* 43 */ SYSENT_CL("times",             times,          1),
492         /* 44 */ SYSENT_CI("prof",              profil,         4),
493         /* 45 */ SYSENT_CI("faccessat",         faccessat,      4),
494         /* 46 */ SYSENT_CI("setgid",            setgid,         1),
495         /* 47 */ SYSENT_2CI("getgid",           getgid,         0),
496         /* 48 */ SYSENT_CI("mknodat",           mknodat,        4),
497         /* 49 */ SYSENT_LOADABLE(),                    /* msgsys */
498         /* 50 */ IF_x86(
499                         SYSENT_CI("sysi86",     sysi86,         4),
500                         SYSENT_LOADABLE()),             /* (was sys3b) */
501         /* 51 */ SYSENT_LOADABLE(),                    /* sysacct */
502         /* 52 */ SYSENT_LOADABLE(),                    /* shmsys */
503         /* 53 */ SYSENT_LOADABLE(),                    /* semsys */
504         /* 54 */ SYSENT_CI("ioctl",             ioctl,          3),
505         /* 55 */ SYSENT_CI("uadmin",            uadmin,         3),
506         /* 56 */ SYSENT_CI("fchownat",          fchownat,       5),
507         /* 57 */ IF_LP64(
508                         SYSENT_2CI("utssys",    utssys64,       4),
509                         SYSENT_2CI("utssys",    utssys32,       4)),
510         /* 58 */ SYSENT_CI("fdsync",            fdsync,         2),
511         /* 59 */ SYSENT_CI("execve",            execve,         3),
512         /* 60 */ SYSENT_CI("umask",             umask,          1),
513         /* 61 */ SYSENT_CI("chroot",            chroot,         1),
514         /* 62 */ SYSENT_CI("fcntl",             fcntl,          3),
515         /* 63 */ SYSENT_CI("ulimit",            ulimit,         2),
516         /* 64 */ SYSENT_CI("renameat",          renameat,       4),
517         /* 65 */ SYSENT_CI("unlinkat",          unlinkat,       3),
518         /* 66 */ SYSENT_CI("fstatat",           fstatat,        4),
519         /* 67 */ IF_LP64(
```

```
520                     SYSENT_NOSYS(),
521                     SYSENT_CI("fstatat64",  fstatat64,      4)),
522         /* 68 */ SYSENT_CI("openat",           openat,         4),
523         /* 69 */ IF_LP64(
524                     SYSENT_NOSYS(),
525                     SYSENT_CI("openat64",   openat64,       4)),
526         /* 70 */ SYSENT_CI("tasksys",          tasksys,        5),
527         /* 71 */ SYSENT_LOADABLE(),            /* acctctl */
528         /* 72 */ SYSENT_LOADABLE(),            /* exacct */
529         /* 73 */ SYSENT_CI("getpagesizes",     getpagesizes,   3),
530         /* 74 */ SYSENT_CI("rctlsys",          rctlsys,        6),
531         /* 75 */ SYSENT_2CI("sidsys",          sidsys,         4),
532         /* 76 */ SYSENT_LOADABLE(),                    /* (was fsat) */
533         /* 77 */ SYSENT_CI("lwp_park",         syslwp_park,    3),
534         /* 78 */ SYSENT_CL("sendfilev",        sendfilev,      5),
535         /* 79 */ SYSENT_CI("rmdir",            rmdir,          1),
536         /* 80 */ SYSENT_CI("mkdir",            mkdir,          2),
537         /* 81 */ IF_LP64(
538                     SYSENT_CI("getdents",   getdents64,     3),
539                     SYSENT_CI("getdents",   getdents32,     3)),
540         /* 82 */ SYSENT_CI("privsys",          privsys,        6),
541         /* 83 */ SYSENT_CI("ucredsys",         ucredsys,       3),
542         /* 84 */ SYSENT_CI("sysfs",            sysfs,          3),
543         /* 85 */ SYSENT_CI("getmsg",           getmsg,         4),
544         /* 86 */ SYSENT_CI("putmsg",           putmsg,         4),
545         /* 87 */ SYSENT_LOADABLE(),                    /* (was poll) */
546         /* 88 */ SYSENT_CI("lstat",            lstat,          2),
547         /* 89 */ SYSENT_CI("symlink",          symlink,        2),
548         /* 90 */ SYSENT_CL("readlink",         readlink,       3),
549         /* 91 */ SYSENT_CI("setgroups",        setgroups,      2),
550         /* 92 */ SYSENT_CI("getgroups",        getgroups,      2),
551         /* 93 */ SYSENT_CI("fchmod",           fchmod,         2),
552         /* 94 */ SYSENT_CI("fchown",           fchown,         3),
553         /* 95 */ SYSENT_CI("sigprocmask",      sigprocmask,    3),
554         /* 96 */ SYSENT_CI("sigsuspend",       sigsuspend,     1),
555         /* 97 */ SYSENT_CI("sigaltstack",      sigaltstack,    2),
556         /* 98 */ SYSENT_CI("sigaction",        sigaction,      3),
557         /* 99 */ SYSENT_CI("sigpending",       sigpending,     2),
558         /* 100 */ SYSENT_CI("getsetcontext",   getsetcontext,  2),
559         /* 101 */ SYSENT_CI("fchmodat",        fchmodat,       4),
560         /* 102 */ SYSENT_CI("mkdirat",         mkdirat,        3),
561         /* 103 */ SYSENT_CI("statvfs",         statvfs,        2),
562         /* 104 */ SYSENT_CI("fstatvfs",        fstatvfs,       2),
563         /* 105 */ SYSENT_CI("getloadavg",      getloadavg,     2),
564         /* 106 */ SYSENT_LOADABLE(),           /* nfssys */
565         /* 107 */ SYSENT_CI("waitsys",         waitsys,        4),
566         /* 108 */ SYSENT_CI("sigsendset",      sigsendsys,     2),
567         /* 109 */ IF_x86(
568                     SYSENT_AP("hrtsys",     hrtsys,         5),
569                     SYSENT_LOADABLE()),
570         /* 110 */ SYSENT_CI("utimesys",        utimesys,       5),
571         /* 111 */ SYSENT_CI("sigresend",       sigresend,      3),
572         /* 112 */ SYSENT_CL("priocntlsys",     priocntlsys,    5),
573         /* 113 */ SYSENT_CL("pathconf",        pathconf,       2),
574         /* 114 */ SYSENT_CI("mincore",         mincore,        3),
575         /* 115 */ IF_LP64(
576                     SYSENT_CL("mmap",       smmap64,        6),
577                     SYSENT_CL("mmap",       smmap32,        6)),
578         /* 116 */ SYSENT_CI("mprotect",        mprotect,       3),
579         /* 117 */ SYSENT_CI("munmap",          munmap,         2),
580         /* 118 */ SYSENT_CL("fpathconf",       fpathconf,      2),
581         /* 119 */ SYSENT_2CI("vfork",          vfork,          0),
582         /* 120 */ SYSENT_CI("fchdir",          fchdir,         1),
583         /* 121 */ SYSENT_CL("readv",           readv,          3),
584         /* 122 */ SYSENT_CL("writev",          writev,         3),
585         /* 123 */ SYSENT_CL("preadv",          preadv,         5),
```



```
586         /* 124 */ SYSENT_CL("pwritev",         pwritev,        5),
587         /* 125 */ SYSENT_LOADABLE(),                   /* (was fxstat) */
588         /* 126 */ SYSENT_CI("getrandom",       getrandom,      3),
589         /* 127 */ SYSENT_CI("mmapobj",         mmapobjsys,     5),
590         /* 128 */ IF_LP64(
591                     SYSENT_CI("setrlimit",  setrlimit64,    2),
592                     SYSENT_CI("setrlimit",  setrlimit32,    2)),
593         /* 129 */ IF_LP64(
594                     SYSENT_CI("getrlimit",  getrlimit64,    2),
595                     SYSENT_CI("getrlimit",  getrlimit32,    2)),
596         /* 130 */ SYSENT_CI("lchown",          lchown,         3),
597         /* 131 */ SYSENT_CI("memcntl",         memcntl,        6),
598         /* 132 */ SYSENT_CI("getpmsg",         getpmsg,        5),
599         /* 133 */ SYSENT_CI("putpmsg",         putpmsg,        5),
600         /* 134 */ SYSENT_CI("rename",          rename,         2),
601         /* 135 */ SYSENT_CI("uname",           uname,          1),
602         /* 136 */ SYSENT_CI("setegid",         setegid,        1),
603         /* 137 */ SYSENT_CL("sysconfig",       sysconfig,      1),
604         /* 138 */ SYSENT_CI("adjtime",         adjtime,        2),
605         /* 139 */ SYSENT_CL("systeminfo",      systeminfo,     3),
606         /* 140 */ SYSENT_LOADABLE(),           /* sharefs */
607         /* 141 */ SYSENT_CI("seteuid",         seteuid,        1),
608         /* 142 */ SYSENT_2CI("forksys",        forksys,        2),
609         /* 143 */ SYSENT_LOADABLE(),                   /* (was fork1) */
610         /* 144 */ SYSENT_CI("sigtimedwait",    sigtimedwait,   3),
611         /* 145 */ SYSENT_CI("lwp_info",        lwp_info,       1),
612         /* 146 */ SYSENT_CI("yield",           yield,          0),
613         /* 147 */ SYSENT_LOADABLE(),               /* (was lwp_sema_wait) */
614         /* 148 */ SYSENT_CI("lwp_sema_post",   lwp_sema_post,  1),
615         /* 149 */ SYSENT_CI("lwp_sema_trywait", lwp_sema_trywait, 1),
616         /* 150 */ SYSENT_CI("lwp_detach",      lwp_detach,     1),
617         /* 151 */ SYSENT_CI("corectl",         corectl,        4),
618         /* 152 */ SYSENT_CI("modctl",          modctl,         6),
619         /* 153 */ SYSENT_CI("fchroot",         fchroot,        1),
620         /* 154 */ SYSENT_LOADABLE(),                   /* (was utimes) */
621         /* 155 */ SYSENT_CI("vhangup",         vhangup,        0),
622         /* 156 */ SYSENT_CI("gettimeofday",    gettimeofday,   1),
623         /* 157 */ SYSENT_CI("getitimer",       getitimer,      2),
624         /* 158 */ SYSENT_CI("setitimer",       setitimer,      3),
625         /* 159 */ SYSENT_CI("lwp_create",      syslwp_create,  3),
626         /* 160 */ SYSENT_CI("lwp_exit", (int (*)())syslwp_exit, 0),
627         /* 161 */ SYSENT_CI("lwp_suspend",     syslwp_suspend, 1),
628         /* 162 */ SYSENT_CI("lwp_continue",    syslwp_continue, 1),
629         /* 163 */ SYSENT_CI("lwp_kill",        lwp_kill,       2),
630         /* 164 */ SYSENT_CI("lwp_self",        lwp_self,       0),
631         /* 165 */ SYSENT_2CI("lwp_sigmask",    lwp_sigmask,    5),
632         /* 166 */ IF_x86(
633                     SYSENT_CI("lwp_private", syslwp_private, 3),
634                     SYSENT_NOSYS()),
635         /* 167 */ SYSENT_CI("lwp_wait",        lwp_wait,       2),
636         /* 168 */ SYSENT_CI("lwp_mutex_wakeup", lwp_mutex_wakeup,      2),
637         /* 169 */ SYSENT_LOADABLE(),               /* (was lwp_mutex_lock) */
638         /* 170 */ SYSENT_CI("lwp_cond_wait",   lwp_cond_wait,          4),
639         /* 171 */ SYSENT_CI("lwp_cond_signal", lwp_cond_signal,        1),
640         /* 172 */ SYSENT_CI("lwp_cond_broadcast", lwp_cond_broadcast,  1),
641         /* 173 */ SYSENT_CL("pread",           pread,          4),
642         /* 174 */ SYSENT_CL("pwrite ",         pwrite,         4),
643         /*
644          * The 64-bit C library maps llseek() to lseek(), so this
645          * is needed as a native syscall only on the 32-bit kernel.
646          */
647         /* 175 */ IF_LP64(
648                     SYSENT_NOSYS(),
649                     SYSENT_C("llseek",      llseek32,       4)),
650         /* 176 */ SYSENT_LOADABLE(),           /* inst_sync */
651         /* 177 */ SYSENT_CI("brandsys",        brandsys,       6),
```

```
652         /* 178 */ SYSENT_LOADABLE(),            /* kaio */
653         /* 179 */ SYSENT_LOADABLE(),            /* cpc */
654         /* 180 */ SYSENT_CI("lgrpsys",          lgrpsys,        3),
655         /* 181 */ SYSENT_CI("rusagesys",        rusagesys,      5),
656         /* 182 */ SYSENT_LOADABLE(),            /* portfs */
657         /* 183 */ SYSENT_CI("pollsys",          pollsys,        4),
658         /* 184 */ SYSENT_CI("labelsys",         labelsys,       5),
659         /* 185 */ SYSENT_CI("acl",              acl,            4),
660         /* 186 */ SYSENT_AP("auditsys",         auditsys,       6),
661         /* 187 */ SYSENT_CI("processor_bind",   processor_bind, 4),
662         /* 188 */ SYSENT_CI("processor_info",   processor_info, 2),
663         /* 189 */ SYSENT_CI("p_online",         p_online,       2),
664         /* 190 */ SYSENT_CI("sigqueue",         sigqueue,       5),
665         /* 191 */ SYSENT_CI("clock_gettime",    clock_gettime,  2),
666         /* 192 */ SYSENT_CI("clock_settime",    clock_settime,  2),
667         /* 193 */ SYSENT_CI("clock_getres",     clock_getres,   2),
668         /* 194 */ SYSENT_CI("timer_create",     timer_create,   3),
669         /* 195 */ SYSENT_CI("timer_delete",     timer_delete,   1),
670         /* 196 */ SYSENT_CI("timer_settime",    timer_settime,  4),
671         /* 197 */ SYSENT_CI("timer_gettime",    timer_gettime,  2),
672         /* 198 */ SYSENT_CI("timer_getoverrun", timer_getoverrun, 1),
673         /* 199 */ SYSENT_CI("nanosleep",        nanosleep,      2),
674         /* 200 */ SYSENT_CI("facl",             facl,           4),
675         /* 201 */ SYSENT_LOADABLE(),            /* door */
676         /* 202 */ SYSENT_CI("setreuid",         setreuid,       2),
677         /* 203 */ SYSENT_CI("setregid",         setregid,       2),
678         /* 204 */ SYSENT_CI("install_utrap",    install_utrap,  3),
679         /* 205 */ SYSENT_CI("signotify",        signotify,      3),
680         /* 206 */ SYSENT_CL("schedctl",         schedctl,       0),
681         /* 207 */ SYSENT_LOADABLE(),            /* pset */
682         /* 208 */ IF_sparc(
683                    SYSENT_CI("sparc_utrap_install", sparc_utrap_install, 5),
684                    SYSENT_NOSYS()),
685         /* 209 */ SYSENT_CI("resolvepath",      resolvepath,    3),
686         /* 210 */ SYSENT_CI("lwp_mutex_timedlock", lwp_mutex_timedlock, 3),
687         /* 211 */ SYSENT_CI("lwp_sema_timedwait", lwp_sema_timedwait,   3),
688         /* 212 */ SYSENT_CI("lwp_rwlock_sys",   lwp_rwlock_sys, 3),
689         /*
690          * Syscalls 213-225: 32-bit system call support for large files.
691          *
692          * (The 64-bit C library transparently maps these system calls
693          * back to their native versions, so almost all of them are only
694          * needed as native syscalls on the 32-bit kernel).
695          */
696         /* 213 */ IF_LP64(
697                       SYSENT_NOSYS(),
698                       SYSENT_CI("getdents64", getdents64,     3)),
699         /* 214 */ IF_LP64(
700                       SYSENT_NOSYS(),
701                       SYSENT_AP("smmaplf32",  smmaplf32,      7)),
702         /* 215 */ IF_LP64(
703                       SYSENT_NOSYS(),
704                       SYSENT_CI("stat64",     stat64,         2)),
705         /* 216 */ IF_LP64(
706                       SYSENT_NOSYS(),
707                       SYSENT_CI("lstat64",    lstat64,        2)),
708         /* 217 */ IF_LP64(
709                       SYSENT_NOSYS(),
710                       SYSENT_CI("fstat64",    fstat64,        2)),
711         /* 218 */ IF_LP64(
712                       SYSENT_NOSYS(),
713                       SYSENT_CI("statvfs64",  statvfs64,      2)),
714         /* 219 */ IF_LP64(
715                       SYSENT_NOSYS(),
716                       SYSENT_CI("fstatvfs64", fstatvfs64,     2)),
717         /* 220 */ IF_LP64(
```



```
718                       SYSENT_NOSYS(),
719                       SYSENT_CI("setrlimit64", setrlimit64,   2)),
720         /* 221 */ IF_LP64(
721                       SYSENT_NOSYS(),
722                       SYSENT_CI("getrlimit64", getrlimit64,   2)),
723         /* 222 */ IF_LP64(
724                       SYSENT_NOSYS(),
725                       SYSENT_CI("pread64",    pread64,        5)),
726         /* 223 */ IF_LP64(
727                       SYSENT_NOSYS(),
728                       SYSENT_CI("pwrite64",   pwrite64,       5)),
729         /* 224 */ SYSENT_LOADABLE(),                   /* (was creat64) */
730         /* 225 */ IF_LP64(
731                       SYSENT_NOSYS(),
732                       SYSENT_CI("open64",     open64,         3)),
733         /* 226 */ SYSENT_LOADABLE(),            /* rpcsys */
734         /* 227 */ SYSENT_CL("zone",             zone,           5),
735         /* 228 */ SYSENT_LOADABLE(),            /* autofssys */
736         /* 229 */ SYSENT_CI("getcwd",           getcwd,         2),
737         /* 230 */ SYSENT_CI("so_socket",        so_socket,      5),
738         /* 231 */ SYSENT_CI("so_socketpair",    so_socketpair,  1),
739         /* 232 */ SYSENT_CI("bind",             bind,           4),
740         /* 233 */ SYSENT_CI("listen",           listen,         3),
741         /* 234 */ SYSENT_CI("accept",           accept,         5),
742         /* 235 */ SYSENT_CI("connect",          connect,        4),
743         /* 236 */ SYSENT_CI("shutdown",         shutdown,       3),
744         /* 237 */ SYSENT_CL("recv",             recv,           4),
745         /* 238 */ SYSENT_CL("recvfrom",         recvfrom,       6),
746         /* 239 */ SYSENT_CL("recvmsg",          recvmsg,        3),
747         /* 240 */ SYSENT_CL("send",             send,           4),
748         /* 241 */ SYSENT_CL("sendmsg",          sendmsg,        3),
749         /* 242 */ SYSENT_CL("sendto",           sendto,         6),
750         /* 243 */ SYSENT_CI("getpeername",      getpeername,    4),
751         /* 244 */ SYSENT_CI("getsockname",      getsockname,    4),
752         /* 245 */ SYSENT_CI("getsockopt",       getsockopt,     6),
753         /* 246 */ SYSENT_CI("setsockopt",       setsockopt,     6),
754         /* 247 */ SYSENT_CI("sockconfig",       sockconfig,     5),
755         /* 248 */ SYSENT_CI("ntp_gettime",      ntp_gettime,    1),
756         /* 249 */ SYSENT_CI("ntp_adjtime",      ntp_adjtime,    1),
757         /* 250 */ SYSENT_CI("lwp_mutex_unlock", lwp_mutex_unlock,       1),
758         /* 251 */ SYSENT_CI("lwp_mutex_trylock", lwp_mutex_trylock,     2),
759         /* 252 */ SYSENT_CI("lwp_mutex_register", lwp_mutex_register,   2),
760         /* 253 */ SYSENT_CI("cladm",            cladm,          3),
761         /* 254 */ SYSENT_CI("uucopy",           uucopy,         3),
762         /* 255 */ SYSENT_CI("umount2",          umount2,        2)
763 };


766 #ifdef _SYSCALL32_IMPL

768 extern int ulimit32(int, int);
769 extern ssize_t read32(int32_t, caddr32_t, size32_t);
770 extern ssize_t write32(int32_t, caddr32_t, size32_t);
771 extern ssize_t pread32(int32_t, caddr32_t, size32_t, off32_t);
772 extern ssize_t pwrite32(int32_t, caddr32_t, size32_t, off32_t);
773 extern ssize_t readv32(int32_t, caddr32_t, int32_t);
774 extern ssize_t writev32(int32_t, caddr32_t, int32_t);
775 extern ssize_t readlink32(caddr32_t, caddr32_t, size32_t);
776 extern ssize_t readlinkat32(int, caddr32_t, caddr32_t, size32_t);
777 extern int open32(char *, int, int);
778 extern int openat32(int, char *, int, int);
779 extern int stat32(char *, struct stat32 *);
780 extern int fstatat32(int, char *, struct stat32 *, int);
781 extern int lstat32(char *, struct stat32 *);
782 extern int fstat32(int, struct stat32 *);
783 extern int fstatat64_32(int, char *, struct stat64_32 *, int);
```

```
784 extern int stat64_32(char *, struct stat64_32 *);
785 extern int lstat64_32(char *, struct stat64_32 *);
786 extern int fstat64_32(int, struct stat64_32 *);
787 extern int getmsg32(int, struct strbuf32 *, struct strbuf32 *, int32_t *);
788 extern int putmsg32(int, struct strbuf32 *, struct strbuf32 *, int32_t *);
789 extern int getpmsg32(int, struct strbuf32 *, struct strbuf32 *, int32_t *,
790     int32_t *);
791 extern int putpmsg32(int, struct strbuf32 *, struct strbuf32 *, int32_t,
792     int32_t);
793 extern int getsetcontext32(int, void *);
794 extern int statvfs32(char *, struct statvfs32 *);
795 extern int fstatvfs32(int, struct statvfs32 *);
796 extern int statvfs64_32(char *, struct statvfs64_32 *);
797 extern int fstatvfs64_32(int, struct statvfs64_32 *);
798 extern int sigaction32(int, struct sigaction32 *, struct sigaction32 *);
799 extern clock32_t times32(struct tms32 *);
800 extern int stime32(time32_t);
801 extern int getpagesizes32(int, size32_t *, int);
802 extern int sigaltstack32(struct sigaltstack32 *, struct sigaltstack32 *);
803 extern int sigqueue32(pid_t, int, caddr32_t, int, int);
804 extern offset_t llseek32(int32_t, uint32_t, uint32_t, int);
805 extern int waitsys32(idtype_t, id_t, siginfo_t *, int);

807 extern ssize_t recv32(int32_t, caddr32_t, size32_t, int32_t);
808 extern ssize_t recvfrom32(int32_t, caddr32_t, size32_t, int32_t, caddr32_t,
809     caddr32_t);
810 extern ssize_t send32(int32_t, caddr32_t, size32_t, int32_t);
811 extern ssize_t sendto32(int32_t, caddr32_t, size32_t, int32_t, caddr32_t,
812     socklen_t);

814 extern int privsys32(int, priv_op_t, priv_ptype_t, caddr32_t, size32_t, int);
815 extern int ucredsys32(int, int, caddr32_t);

817 /*
818  * sysent table for ILP32 processes running on
819  * a LP64 kernel.
820  */
821 struct sysent sysent32[NSYSCALL] =
822 {
823         /*  0 */ SYSENT_C("indir",              indir,          1),
824         /*  1 */ SYSENT_CI("exit",      (int (*)())rexit,       1),
825         /*  2 */ SYSENT_CI("psecflags",         psecflags,      3),
496         /*  2 */ SYSENT_LOADABLE32(),                   /* (was forkall) */
826         /*  3 */ SYSENT_CI("read",              read32,         3),
827         /*  4 */ SYSENT_CI("write",             write32,        3),
828         /*  5 */ SYSENT_CI("open",              open32,         3),
829         /*  6 */ SYSENT_CI("close",             close,          1),
830         /*  7 */ SYSENT_CI("linkat",            linkat,         5),
831         /*  8 */ SYSENT_LOADABLE32(),                   /* (was creat32) */
832         /*  9 */ SYSENT_CI("link",              link,           2),
833         /* 10 */ SYSENT_CI("unlink",            unlink,         1),
834         /* 11 */ SYSENT_CI("symlinkat",         symlinkat,      3),
835         /* 12 */ SYSENT_CI("chdir",             chdir,          1),
836         /* 13 */ SYSENT_CI("time",              gtime,          0),
837         /* 14 */ SYSENT_CI("mknod",             mknod,          3),
838         /* 15 */ SYSENT_CI("chmod",             chmod,          2),
839         /* 16 */ SYSENT_CI("chown",             chown,          3),
840         /* 17 */ SYSENT_CI("brk",               brk,            1),
841         /* 18 */ SYSENT_CI("stat",              stat32,         2),
842         /* 19 */ SYSENT_CI("lseek",             lseek32,        3),
843         /* 20 */ SYSENT_2CI("getpid",           getpid,         0),
844         /* 21 */ SYSENT_AP("mount",             mount,          8),
845         /* 22 */ SYSENT_CI("readlinkat",        readlinkat32,   4),
846         /* 23 */ SYSENT_CI("setuid",            setuid,         1),
847         /* 24 */ SYSENT_2CI("getuid",           getuid,         0),
848         /* 25 */ SYSENT_CI("stime",             stime32,        1),
```



```
849         /* 26 */ SYSENT_CI("pcsample",          pcsample,       2),
850         /* 27 */ SYSENT_CI("alarm",             alarm,          1),
851         /* 28 */ SYSENT_CI("fstat",             fstat32,        2),
852         /* 29 */ SYSENT_CI("pause",             pause,          0),
853         /* 30 */ SYSENT_LOADABLE32(),                  /* (was utime) */
854         /* 31 */ SYSENT_CI("stty",              stty,           2),
855         /* 32 */ SYSENT_CI("gtty",              gtty,           2),
856         /* 33 */ SYSENT_CI("access",            access,         2),
857         /* 34 */ SYSENT_CI("nice",              nice,           1),
858         /* 35 */ SYSENT_CI("statfs",            statfs32,       4),
859         /* 36 */ SYSENT_CI("sync",              syssync,        0),
860         /* 37 */ SYSENT_CI("kill",              kill,           2),
861         /* 38 */ SYSENT_CI("fstatfs",           fstatfs32,      4),
862         /* 39 */ SYSENT_CI("setpgrp",           setpgrp,        3),
863         /* 40 */ SYSENT_CI("uucopystr",         uucopystr,      3),
864         /* 41 */ SYSENT_LOADABLE32(),                  /* (was dup) */
865         /* 42 */ SYSENT_LOADABLE32(),                  /* pipe */
866         /* 43 */ SYSENT_CI("times",             times32,        1),
867         /* 44 */ SYSENT_CI("prof",              profil,         4),
868         /* 45 */ SYSENT_CI("faccessat",         faccessat,      4),
869         /* 46 */ SYSENT_CI("setgid",            setgid,         1),
870         /* 47 */ SYSENT_2CI("getgid",           getgid,         0),
871         /* 48 */ SYSENT_CI("mknodat",           mknodat,        4),
872         /* 49 */ SYSENT_LOADABLE32(),                  /* msgsys */
873         /* 50 */ IF_386_ABI(
874                         SYSENT_CI("sysi86",     sysi86,         4),
875                         SYSENT_LOADABLE32()),           /* (was sys3b) */
876         /* 51 */ SYSENT_LOADABLE32(),                  /* sysacct */
877         /* 52 */ SYSENT_LOADABLE32(),                  /* shmsys */
878         /* 53 */ SYSENT_LOADABLE32(),                  /* semsys */
879         /* 54 */ SYSENT_CI("ioctl",             ioctl,          3),
880         /* 55 */ SYSENT_CI("uadmin",            uadmin,         3),
881         /* 56 */ SYSENT_CI("fchownat",          fchownat,       5),
882         /* 57 */ SYSENT_2CI("utssys",           utssys32,       4),
883         /* 58 */ SYSENT_CI("fdsync",            fdsync,         2),
884         /* 59 */ SYSENT_CI("exece",             exece,          3),
885         /* 60 */ SYSENT_CI("umask",             umask,          1),
886         /* 61 */ SYSENT_CI("chroot",            chroot,         1),
887         /* 62 */ SYSENT_CI("fcntl",             fcntl,          3),
888         /* 63 */ SYSENT_CI("ulimit",            ulimit32,       2),
889         /* 64 */ SYSENT_CI("renameat",          renameat,       4),
890         /* 65 */ SYSENT_CI("unlinkat",          unlinkat,       3),
891         /* 66 */ SYSENT_CI("fstatat",           fstatat32,      4),
892         /* 67 */ SYSENT_CI("fstatat64",         fstatat64_32,   4),
893         /* 68 */ SYSENT_CI("openat",            openat32,       4),
894         /* 69 */ SYSENT_CI("openat64",          openat64,       4),
895         /* 70 */ SYSENT_CI("tasksys",           tasksys,        5),
896         /* 71 */ SYSENT_LOADABLE32(),           /* acctctl */
897         /* 72 */ SYSENT_LOADABLE32(),           /* exacct */
898         /* 73 */ SYSENT_CI("getpagesizes",      getpagesizes32, 3),
899         /* 74 */ SYSENT_CI("rctlsys",           rctlsys,        6),
900         /* 75 */ SYSENT_2CI("sidsys",           sidsys,         4),
901         /* 76 */ SYSENT_LOADABLE32(),                  /* (was fsat) */
902         /* 77 */ SYSENT_CI("lwp_park",          syslwp_park,    3),
903         /* 78 */ SYSENT_CI("sendfilev",         sendfilev,      5),
904         /* 79 */ SYSENT_CI("rmdir",             rmdir,          1),
905         /* 80 */ SYSENT_CI("mkdir",             mkdir,          2),
906         /* 81 */ SYSENT_CI("getdents",          getdents32,     3),
907         /* 82 */ SYSENT_CI("privsys",           privsys32,      6),
908         /* 83 */ SYSENT_CI("ucredsys",          ucredsys32,     3),
909         /* 84 */ SYSENT_CI("sysfs",             sysfs,          3),
910         /* 85 */ SYSENT_CI("getmsg",            getmsg32,       4),
911         /* 86 */ SYSENT_CI("putmsg",            putmsg32,       4),
912         /* 87 */ SYSENT_LOADABLE32(),                  /* (was poll) */
913         /* 88 */ SYSENT_CI("lstat",             lstat32,        2),
914         /* 89 */ SYSENT_CI("symlink",           symlink,        2),
```

```
915         /* 90 */ SYSENT_CI("readlink",          readlink32,     3),
916         /* 91 */ SYSENT_CI("setgroups",         setgroups,      2),
917         /* 92 */ SYSENT_CI("getgroups",         getgroups,      2),
918         /* 93 */ SYSENT_CI("fchmod",            fchmod,         2),
919         /* 94 */ SYSENT_CI("fchown",            fchown,         3),
920         /* 95 */ SYSENT_CI("sigprocmask",       sigprocmask,    3),
921         /* 96 */ SYSENT_CI("sigsuspend",        sigsuspend,     1),
922         /* 97 */ SYSENT_CI("sigaltstack",       sigaltstack32,  2),
923         /* 98 */ SYSENT_CI("sigaction",         sigaction32,    3),
924         /* 99 */ SYSENT_CI("sigpending",        sigpending,     2),
925         /* 100 */ SYSENT_CI("getsetcontext",    getsetcontext32, 2),
926         /* 101 */ SYSENT_CI("fchmodat",         fchmodat,       4),
927         /* 102 */ SYSENT_CI("mkdirat",          mkdirat,        3),
928         /* 103 */ SYSENT_CI("statvfs",          statvfs32,      2),
929         /* 104 */ SYSENT_CI("fstatvfs",         fstatvfs32,     2),
930         /* 105 */ SYSENT_CI("getloadavg",       getloadavg,     2),
931         /* 106 */ SYSENT_LOADABLE32(),          /* nfssys */
932         /* 107 */ SYSENT_CI("waitsys",          waitsys32,      4),
933         /* 108 */ SYSENT_CI("sigsendset",       sigsendsys,     2),
934         /* 109 */ IF_x86(
935                       SYSENT_AP("hrtsys",       hrtsys,         5),
936                       SYSENT_LOADABLE32()),
937         /* 110 */ SYSENT_CI("utimesys",         utimesys,       5),
938         /* 111 */ SYSENT_CI("sigresend",        sigresend,      3),
939         /* 112 */ SYSENT_CI("priocntlsys",      priocntlsys,    5),
940         /* 113 */ SYSENT_CI("pathconf",         pathconf,       2),
941         /* 114 */ SYSENT_CI("mincore",          mincore,        3),
942         /* 115 */ SYSENT_CI("mmap",             smmap32,        6),
943         /* 116 */ SYSENT_CI("mprotect",         mprotect,       3),
944         /* 117 */ SYSENT_CI("munmap",           munmap,         2),
945         /* 118 */ SYSENT_CI("fpathconf",        fpathconf,      2),
946         /* 119 */ SYSENT_2CI("vfork",           vfork,          0),
947         /* 120 */ SYSENT_CI("fchdir",           fchdir,         1),
948         /* 121 */ SYSENT_CI("readv",            readv32,        3),
949         /* 122 */ SYSENT_CI("writev",           writev32,       3),
950         /* 123 */ SYSENT_CI("preadv",           preadv,         5),
951         /* 124 */ SYSENT_CI("pwritev",          pwritev,        5),
952         /* 125 */ SYSENT_LOADABLE32(),          /*      was fxstat32    */
953         /* 126 */ SYSENT_CI("getrandom",        getrandom,      3),
954         /* 127 */ SYSENT_CI("mmapobj",          mmapobjsys,     5),
955         /* 128 */ SYSENT_CI("setrlimit",        setrlimit32,    2),
956         /* 129 */ SYSENT_CI("getrlimit",        getrlimit32,    2),
957         /* 130 */ SYSENT_CI("lchown",           lchown,         3),
958         /* 131 */ SYSENT_CI("memcntl",          memcntl,        6),
959         /* 132 */ SYSENT_CI("getpmsg",          getpmsg32,      5),
960         /* 133 */ SYSENT_CI("putpmsg",          putpmsg32,      5),
961         /* 134 */ SYSENT_CI("rename",           rename,         2),
962         /* 135 */ SYSENT_CI("uname",            uname,          1),
963         /* 136 */ SYSENT_CI("setegid",          setegid,        1),
964         /* 137 */ SYSENT_CI("sysconfig",        sysconfig,      1),
965         /* 138 */ SYSENT_CI("adjtime",          adjtime,        2),
966         /* 139 */ SYSENT_CI("systeminfo",       systeminfo,     3),
967         /* 140 */ SYSENT_LOADABLE32(),          /* sharefs */
968         /* 141 */ SYSENT_CI("seteuid",          seteuid,        1),
969         /* 142 */ SYSENT_2CI("forksys",         forksys,        2),
970         /* 143 */ SYSENT_LOADABLE32(),                  /* (was fork1) */
971         /* 144 */ SYSENT_CI("sigtimedwait",     sigtimedwait,   3),
972         /* 145 */ SYSENT_CI("lwp_info",         lwp_info,       1),
973         /* 146 */ SYSENT_CI("yield",            yield,          0),
974         /* 147 */ SYSENT_LOADABLE32(),             /* (was lwp_sema_wait) */
975         /* 148 */ SYSENT_CI("lwp_sema_post",    lwp_sema_post,  1),
976         /* 149 */ SYSENT_CI("lwp_sema_trywait", lwp_sema_trywait, 1),
977         /* 150 */ SYSENT_CI("lwp_detach",       lwp_detach,     1),
978         /* 151 */ SYSENT_CI("corectl",          corectl,        4),
979         /* 152 */ SYSENT_CI("modctl",           modctl,         6),
980         /* 153 */ SYSENT_CI("fchroot",          fchroot,        1),
```



```
981         /* 154 */ SYSENT_LOADABLE32(),                  /* (was utimes) */
982         /* 155 */ SYSENT_CI("vhangup",          vhangup,        0),
983         /* 156 */ SYSENT_CI("gettimeofday",     gettimeofday,   1),
984         /* 157 */ SYSENT_CI("getitimer",        getitimer,      2),
985         /* 158 */ SYSENT_CI("setitimer",        setitimer,      3),
986         /* 159 */ SYSENT_CI("lwp_create",       syslwp_create,  3),
987         /* 160 */ SYSENT_CI("lwp_exit", (int (*)())syslwp_exit, 0),
988         /* 161 */ SYSENT_CI("lwp_suspend",      syslwp_suspend, 1),
989         /* 162 */ SYSENT_CI("lwp_continue",     syslwp_continue, 1),
990         /* 163 */ SYSENT_CI("lwp_kill",         lwp_kill,       2),
991         /* 164 */ SYSENT_CI("lwp_self",         lwp_self,       0),
992         /* 165 */ SYSENT_2CI("lwp_sigmask",     lwp_sigmask,    5),
993         /* 166 */ IF_x86(
994                       SYSENT_CI("lwp_private", syslwp_private, 3),
995                       SYSENT_NOSYS()),
996         /* 167 */ SYSENT_CI("lwp_wait",         lwp_wait,       2),
997         /* 168 */ SYSENT_CI("lwp_mutex_wakeup", lwp_mutex_wakeup,       2),
998         /* 169 */ SYSENT_LOADABLE32(),                  /* (was lwp_mutex_lock) */
999         /* 170 */ SYSENT_CI("lwp_cond_wait",    lwp_cond_wait,          4),
1000        /* 171 */ SYSENT_CI("lwp_cond_signal",  lwp_cond_signal,        1),
1001        /* 172 */ SYSENT_CI("lwp_cond_broadcast", lwp_cond_broadcast,   1),
1002        /* 173 */ SYSENT_CI("pread",            pread32,                4),
1003        /* 174 */ SYSENT_CI("pwrite",           pwrite32,               4),
1004        /* 175 */ SYSENT_C("llseek",            llseek32,       4),
1005        /* 176 */ SYSENT_LOADABLE32(),          /* inst_sync */
1006        /* 177 */ SYSENT_CI("brandsys",         brandsys,       6),
1007        /* 178 */ SYSENT_LOADABLE32(),          /* kaio */
1008        /* 179 */ SYSENT_LOADABLE32(),          /* cpc */
1009        /* 180 */ SYSENT_CI("lgrpsys",          lgrpsys,        3),
1010        /* 181 */ SYSENT_CI("rusagesys",        rusagesys,      5),
1011        /* 182 */ SYSENT_LOADABLE32(),          /* portfs */
1012        /* 183 */ SYSENT_CI("pollsys",          pollsys,        4),
1013        /* 184 */ SYSENT_CI("labelsys",         labelsys,       5),
1014        /* 185 */ SYSENT_CI("acl",              acl,            4),
1015        /* 186 */ SYSENT_AP("auditsys",         auditsys,       6),
1016        /* 187 */ SYSENT_CI("processor_bind",   processor_bind, 4),
1017        /* 188 */ SYSENT_CI("processor_info",   processor_info, 2),
1018        /* 189 */ SYSENT_CI("p_online",         p_online,       2),
1019        /* 190 */ SYSENT_CI("sigqueue",         sigqueue32,     5),
1020        /* 191 */ SYSENT_CI("clock_gettime",    clock_gettime,  2),
1021        /* 192 */ SYSENT_CI("clock_settime",    clock_settime,  2),
1022        /* 193 */ SYSENT_CI("clock_getres",     clock_getres,   2),
1023        /* 194 */ SYSENT_CI("timer_create",     timer_create,   3),
1024        /* 195 */ SYSENT_CI("timer_delete",     timer_delete,   1),
1025        /* 196 */ SYSENT_CI("timer_settime",    timer_settime,  4),
1026        /* 197 */ SYSENT_CI("timer_gettime",    timer_gettime,  2),
1027        /* 198 */ SYSENT_CI("timer_getoverrun", timer_getoverrun, 1),
1028        /* 199 */ SYSENT_CI("nanosleep",        nanosleep,      2),
1029        /* 200 */ SYSENT_CI("facl",             facl,           4),
1030        /* 201 */ SYSENT_LOADABLE32(),          /* door */
1031        /* 202 */ SYSENT_CI("setreuid",         setreuid,       2),
1032        /* 203 */ SYSENT_CI("setregid",         setregid,       2),
1033        /* 204 */ SYSENT_CI("install_utrap",    install_utrap,  3),
1034        /* 205 */ SYSENT_CI("signotify",        signotify,      3),
1035        /* 206 */ SYSENT_CI("schedctl",         schedctl,       0),
1036        /* 207 */ SYSENT_LOADABLE32(),          /* pset */
1037        /* 208 */ SYSENT_LOADABLE32(),
1038        /* 209 */ SYSENT_CI("resolvepath",      resolvepath,    3),
1039        /* 210 */ SYSENT_CI("lwp_mutex_timedlock", lwp_mutex_timedlock, 3),
1040        /* 211 */ SYSENT_CI("lwp_sema_timedwait", lwp_sema_timedwait,   3),
1041        /* 212 */ SYSENT_CI("lwp_rwlock_sys",   lwp_rwlock_sys, 3),
1042        /*
1043         * Syscalls 213-225: 32-bit system call support for large files.
1044         */
1045        /* 213 */ SYSENT_CI("getdents64",       getdents64,     3),
1046        /* 214 */ SYSENT_AP("smmaplf32",        smmaplf32,      7),
```

```
1047         /* 215 */ SYSENT_CI("stat64",           stat64_32,      2),
1048         /* 216 */ SYSENT_CI("lstat64",          lstat64_32,     2),
1049         /* 217 */ SYSENT_CI("fstat64",          fstat64_32,     2),
1050         /* 218 */ SYSENT_CI("statvfs64",        statvfs64_32,   2),
1051         /* 219 */ SYSENT_CI("fstatvfs64",       fstatvfs64_32,  2),
1052         /* 220 */ SYSENT_CI("setrlimit64",      setrlimit64,    2),
1053         /* 221 */ SYSENT_CI("getrlimit64",      getrlimit64,    2),
1054         /* 222 */ SYSENT_CI("pread64",          pread64,        5),
1055         /* 223 */ SYSENT_CI("pwrite64",         pwrite64,       5),
1056         /* 224 */ SYSENT_LOADABLE32(),                 /* (was creat64) */
1057         /* 225 */ SYSENT_CI("open64",           open64,         3),
1058         /* 226 */ SYSENT_LOADABLE32(),          /* rpcsys */
1059         /* 227 */ SYSENT_CI("zone",             zone,           6),
1060         /* 228 */ SYSENT_LOADABLE32(),          /* autofssys */
1061         /* 229 */ SYSENT_CI("getcwd",           getcwd,         2),
1062         /* 230 */ SYSENT_CI("so_socket",        so_socket,      5),
1063         /* 231 */ SYSENT_CI("so_socketpair",    so_socketpair,  1),
1064         /* 232 */ SYSENT_CI("bind",             bind,           4),
1065         /* 233 */ SYSENT_CI("listen",           listen,         3),
1066         /* 234 */ SYSENT_CI("accept",           accept,         5),
1067         /* 235 */ SYSENT_CI("connect",          connect,        4),
1068         /* 236 */ SYSENT_CI("shutdown",         shutdown,       3),
1069         /* 237 */ SYSENT_CI("recv",             recv32,         4),
1070         /* 238 */ SYSENT_CI("recvfrom",         recvfrom32,     6),
1071         /* 239 */ SYSENT_CI("recvmsg",          recvmsg,        3),
1072         /* 240 */ SYSENT_CI("send",             send32,         4),
1073         /* 241 */ SYSENT_CI("sendmsg",          sendmsg,        3),
1074         /* 242 */ SYSENT_CI("sendto",           sendto32,       6),
1075         /* 243 */ SYSENT_CI("getpeername",      getpeername,    4),
1076         /* 244 */ SYSENT_CI("getsockname",      getsockname,    4),
1077         /* 245 */ SYSENT_CI("getsockopt",       getsockopt,     6),
1078         /* 246 */ SYSENT_CI("setsockopt",       setsockopt,     6),
1079         /* 247 */ SYSENT_CI("sockconfig",       sockconfig,     5),
1080         /* 248 */ SYSENT_CI("ntp_gettime",      ntp_gettime,    1),
1081         /* 249 */ SYSENT_CI("ntp_adjtime",      ntp_adjtime,    1),
1082         /* 250 */ SYSENT_CI("lwp_mutex_unlock", lwp_mutex_unlock,       1),
1083         /* 251 */ SYSENT_CI("lwp_mutex_trylock", lwp_mutex_trylock,     2),
1084         /* 252 */ SYSENT_CI("lwp_mutex_register", lwp_mutex_register,   2),
1085         /* 253 */ SYSENT_CI("cladm",            cladm,          3),
1086         /* 254 */ SYSENT_CI("uucopy",           uucopy,         3),
1087         /* 255 */ SYSENT_CI("umount2",          umount2,        2)
1088 };
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   22102 Wed May 27 19:49:32 2015
new/usr/src/uts/common/sys/Makefile
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
#
# Copyright (c) 1989, 2010, Oracle and/or its affiliates. All rights reserved.
# Copyright 2013, Joyent, Inc. All rights reserved.
# Copyright 2013 Garrett D'Amore <garrett@damore.org>
#

include $(SRC)/uts/Makefile.uts

FILEMODE=644

#
#       Note that the following headers are present in the kernel but
#       neither installed or shipped as part of the product:
#               cpuid_drv.h:            Private interface for cpuid consumers
#               unix_bb_info.h:         Private interface to kcov
#

i386_HDRS=                      \
        agp/agpamd64gart_io.h   \
        agp/agpdefs.h           \
        agp/agpgart_impl.h      \
        agp/agpmaster_io.h      \
        agp/agptarget_io.h      \
        agpgart.h               \
        asy.h                   \
        fd_debug.h              \
        fdc.h                   \
        fdmedia.h               \
        mouse.h                 \
        ucode.h

sparc_HDRS=                     \
        mouse.h                 \
        scsi/targets/ssddef.h   \
        $(MDESCHDRS)

# Generated headers
GENHDRS=                        \
```



```
        priv_const.h            \
        priv_names.h            \
        usb/usbdevs.h

CHKHDRS=                        \
        acpi_drv.h              \
        acct.h                  \
        acctctl.h               \
        acl.h                   \
        acl_impl.h              \
        aggr.h                  \
        aggr_impl.h             \
        aio.h                   \
        aio_impl.h              \
        aio_req.h               \
        aiocb.h                 \
        ascii.h                 \
        asynch.h                \
        atomic.h                \
        attr.h                  \
        audio.h                 \
        audioio.h               \
        autoconf.h              \
        auxv.h                  \
        auxv_386.h              \
        auxv_SPARC.h            \
        avl.h                   \
        avl_impl.h              \
        bitmap.h                \
        bitset.h                \
        bl.h                    \
        blkdev.h                \
        bofi.h                  \
        bofi_impl.h             \
        bpp_io.h                \
        bootstat.h              \
        brand.h                 \
        buf.h                   \
        bufmod.h                \
        bustypes.h              \
        byteorder.h             \
        callb.h                 \
        callo.h                 \
        cap_util.h              \
        cpucaps.h               \
        cpucaps_impl.h          \
        ccompile.h              \
        cdio.h                  \
        cladm.h                 \
        class.h                 \
        clconf.h                \
        clock_impl.h            \
        cmlb.h                  \
        cmn_err.h               \
        compress.h              \
        condvar.h               \
        condvar_impl.h          \
        conf.h                  \
        consdev.h               \
        console.h               \
        consplat.h              \
        vt.h                    \
        vtdaemon.h              \
        kd.h                    \
        contract.h              \
        contract_impl.h         \
```

```
125         copyops.h               \
126         core.h                  \
127         corectl.h               \
128         cpc_impl.h              \
129         cpc_pcbe.h              \
130         cpr.h                   \
131         cpupart.h               \
132         cpuvar.h                \
133         crc32.h                 \
134         cred.h                  \
135         cred_impl.h             \
136         crtctl.h                \
137         cryptmod.h              \
138         csiioctl.h              \
139         ctf.h                   \
140         ctfs.h                  \
141         ctfs_impl.h             \
142         ctf_api.h               \
143         ctype.h                 \
144         cyclic.h                \
145         cyclic_impl.h           \
146         dacf.h                  \
147         dacf_impl.h             \
148         damap.h                 \
149         damap_impl.h            \
150         dc_ki.h                 \
151         ddi.h                   \
152         ddifm.h                 \
153         ddifm_impl.h            \
154         ddi_hp.h                \
155         ddi_hp_impl.h           \
156         ddi_intr.h              \
157         ddi_intr_impl.h         \
158         ddi_impldefs.h          \
159         ddi_implfuncs.h         \
160         ddi_obsolete.h          \
161         ddi_periodic.h          \
162         ddidevmap.h             \
163         ddidmareq.h             \
164         ddimapreq.h             \
165         ddipropdefs.h           \
166         dditypes.h              \
167         debug.h                 \
168         des.h                   \
169         devctl.h                \
170         devcache.h              \
171         devcache_impl.h         \
172         devfm.h                 \
173         devid_cache.h           \
174         devinfo_impl.h          \
175         devops.h                \
176         devpolicy.h             \
177         devpoll.h               \
178         dirent.h                \
179         disp.h                  \
180         dkbad.h                 \
181         dkio.h                  \
182         dklabel.h               \
183         dl.h                    \
184         dlpi.h                  \
185         dld.h                   \
186         dld_impl.h              \
187         dld_ioc.h               \
188         dls.h                   \
189         dls_mgmt.h              \
190         dls_impl.h              \
```



```
191         dma_i8237A.h            \
192         dnlc.h                  \
193         door.h                  \
194         door_data.h             \
195         door_impl.h             \
196         dtrace.h                \
197         dtrace_impl.h           \
198         dumpadm.h               \
199         dumphdr.h               \
200         ecppsys.h               \
201         ecppio.h                \
202         ecppreg.h               \
203         ecppvar.h               \
204         efi_partition.h         \
205         elf.h                   \
206         elf_386.h               \
207         elf_SPARC.h             \
208         elf_notes.h             \
209         elf_amd64.h             \
210         elftypes.h              \
211         emul64.h                \
212         emul64cmd.h             \
213         emul64var.h             \
214         epm.h                   \
215         errno.h                 \
216         errorq.h                \
217         errorq_impl.h           \
218         esunddi.h               \
219         ethernet.h              \
220         euc.h                   \
221         eucioctl.h              \
222         exacct.h                \
223         exacct_catalog.h        \
224         exacct_impl.h           \
225         exec.h                  \
226         exechdr.h               \
227         extdirent.h             \
228         fault.h                 \
229         fasttrap.h              \
230         fasttrap_impl.h         \
231         fbio.h                  \
232         fbuf.h                  \
233         fcntl.h                 \
234         fct.h                   \
235         fct_defines.h           \
236         fctio.h                 \
237         fdbuffer.h              \
238         fdio.h                  \
239         feature_tests.h         \
240         fem.h                   \
241         file.h                  \
242         filio.h                 \
243         flock.h                 \
244         flock_impl.h            \
245         fork.h                  \
246         fss.h                   \
247         fsspriocntl.h           \
248         fsid.h                  \
249         fssnap.h                \
250         fssnap_if.h             \
251         fstyp.h                 \
252         ftrace.h                \
253         fx.h                    \
254         fxpriocntl.h            \
255         gfs.h                   \
256         gld.h                   \
```

```
257         gldpriv.h                \
258         group.h                  \
259         hdio.h                   \
260         hook.h                   \
261         hook_event.h             \
262         hook_impl.h              \
263         hwconf.h                 \
264         ia.h                     \
265         iapriocntl.h             \
266         ibpart.h                 \
267         id32.h                   \
268         idmap.h                  \
269         ieeefp.h                 \
270         id_space.h               \
271         instance.h               \
272         int_const.h              \
273         int_fmtio.h              \
274         int_limits.h             \
275         int_types.h              \
276         inttypes.h               \
277         ioccom.h                 \
278         ioctl.h                  \
279         ipc.h                    \
280         ipc_impl.h               \
281         ipc_rctl.h               \
282         ipd.h                    \
283         ipmi.h                   \
284         isa_defs.h               \
285         iscsi_authclient.h       \
286         iscsi_authclientglue.h   \
287         iscsi_protocol.h         \
288         jioctl.h                 \
289         kbd.h                    \
290         kbdreg.h                 \
291         kbio.h                   \
292         kcpc.h                   \
293         kdi.h                    \
294         kdi_impl.h               \
295         kiconv.h                 \
296         kiconv_big5_utf8.h       \
297         kiconv_cck_common.h      \
298         kiconv_cp950hkscs_utf8.h         \
299         kiconv_emea1.h           \
300         kiconv_emea2.h           \
301         kiconv_euckr_utf8.h      \
302         kiconv_euctw_utf8.h      \
303         kiconv_gb18030_utf8.h    \
304         kiconv_gb2312_utf8.h     \
305         kiconv_hkscs_utf8.h      \
306         kiconv_ja.h              \
307         kiconv_ja_jis_to_unicode.h       \
308         kiconv_ja_unicode_to_jis.h       \
309         kiconv_ko.h              \
310         kiconv_latin1.h          \
311         kiconv_sc.h              \
312         kiconv_tc.h              \
313         kiconv_uhc_utf8.h        \
314         kiconv_utf8_big5.h       \
315         kiconv_utf8_cp950hkscs.h         \
316         kiconv_utf8_euckr.h      \
317         kiconv_utf8_euctw.h      \
318         kiconv_utf8_gb18030.h    \
319         kiconv_utf8_gb2312.h     \
320         kiconv_utf8_hkscs.h      \
321         kiconv_utf8_uhc.h        \
322         kidmap.h                 \
```



```
323         klpd.h                   \
324         klwp.h                   \
325         kmdb.h                   \
326         kmem.h                   \
327         kmem_impl.h              \
328         kobj.h                   \
329         kobj_impl.h              \
330         ksocket.h                \
331         kstat.h                  \
332         kstr.h                   \
333         ksyms.h                  \
334         ksynch.h                 \
335         ldterm.h                 \
336         lgrp.h                   \
337         lgrp_user.h              \
338         libc_kernel.h            \
339         link.h                   \
340         list.h                   \
341         list_impl.h              \
342         llc1.h                   \
343         loadavg.h                \
344         lock.h                   \
345         lockfs.h                 \
346         lockstat.h               \
347         lofi.h                   \
348         log.h                    \
349         logindmux.h              \
350         logindmux_impl.h         \
351         lwp.h                    \
352         lwp_timer_impl.h         \
353         lwp_upimutex_impl.h      \
354         lpif.h                   \
355         mac.h                    \
356         mac_client.h             \
357         mac_client_impl.h        \
358         mac_ether.h              \
359         mac_flow.h               \
360         mac_flow_impl.h          \
361         mac_impl.h               \
362         mac_provider.h           \
363         mac_soft_ring.h          \
364         mac_stat.h               \
365         machelf.h                \
366         map.h                    \
367         md4.h                    \
368         md5.h                    \
369         md5_consts.h             \
370         mdi_impldefs.h           \
371         mem.h                    \
372         mem_config.h             \
373         memlist.h                \
374         mkdev.h                  \
375         mhd.h                    \
376         mii.h                    \
377         miiregs.h                \
378         mixer.h                  \
379         mman.h                   \
380         mmapobj.h                \
381         mntent.h                 \
382         mntio.h                  \
383         mnttab.h                 \
384         modctl.h                 \
385         mode.h                   \
386         model.h                  \
387         modhash.h                \
388         modhash_impl.h           \
```

```
389         mount.h                 \
390         mouse.h                 \
391         msacct.h                \
392         msg.h                   \
393         msg_impl.h              \
394         msio.h                  \
395         msreg.h                 \
396         mtio.h                  \
397         multidata.h             \
398         multidata_impl.h        \
399         mutex.h                 \
400         nbmlock.h               \
401         ndifm.h                 \
402         ndi_impldefs.h          \
403         net80211.h              \
404         net80211_crypto.h       \
405         net80211_ht.h           \
406         net80211_proto.h        \
407         netconfig.h             \
408         neti.h                  \
409         netstack.h              \
410         nexusdefs.h             \
411         note.h                  \
412         nvpair.h                \
413         nvpair_impl.h           \
414         objfs.h                 \
415         objfs_impl.h            \
416         ontrap.h                \
417         open.h                  \
418         openpromio.h            \
419         panic.h                 \
420         param.h                 \
421         pathconf.h              \
422         pathname.h              \
423         pattr.h                 \
424         queue.h                 \
425         serializer.h            \
426         pbio.h                  \
427         pccard.h                \
428         pci.h                   \
429         pcie.h                  \
430         pci_impl.h              \
431         pci_tools.h             \
432         pcmcia.h                \
433         pctypes.h               \
434         pfmod.h                 \
435         pg.h                    \
436         pghw.h                  \
437         physmem.h               \
438         pkp_hash.h              \
439         pm.h                    \
440         policy.h                \
441         poll.h                  \
442         poll_impl.h             \
443         pool.h                  \
444         pool_impl.h             \
445         pool_pset.h             \
446         port.h                  \
447         port_impl.h             \
448         port_kernel.h           \
449         portif.h                \
450         ppmio.h                 \
451         pppt_ic_if.h            \
452         pppt_ioctl.h            \
453         priocntl.h              \
454         priv.h                  \
```



```
455         priv_impl.h             \
456         prnio.h                 \
457         proc.h                  \
458         processor.h             \
459         procfs.h                \
460         procset.h               \
461         project.h               \
462         protosw.h               \
463         prsystm.h               \
464         pset.h                  \
465         pshot.h                 \
466         ptem.h                  \
467         ptms.h                  \
468         ptyvar.h                \
469         raidioctl.h             \
470         ramdisk.h               \
471         random.h                \
472         rctl.h                  \
473         rctl_impl.h             \
474         rds.h                   \
475         reboot.h                \
476         refstr.h                \
477         refstr_impl.h           \
478         resource.h              \
479         rlioctl.h               \
480         rt.h                    \
481         rtpriocntl.h            \
482         rwlock.h                \
483         rwlock_impl.h           \
484         rwstlock.h              \
485         sad.h                   \
486         schedctl.h              \
487         sdt.h                   \
488         secflags.h              \
489 #endif /* ! codereview */
490         select.h                \
491         sem.h                   \
492         sem_impl.h              \
493         sema_impl.h             \
494         semaphore.h             \
495         sendfile.h              \
496         ser_sync.h              \
497         session.h               \
498         sha1.h                  \
499         sha1_consts.h           \
500         sha2.h                  \
501         sha2_consts.h           \
502         share.h                 \
503         shm.h                   \
504         shm_impl.h              \
505         sid.h                   \
506         siginfo.h               \
507         signal.h                \
508         sleepq.h                \
509         smbios.h                \
510         smbios_impl.h           \
511         sobject.h               \
512         socket.h                \
513         socket_impl.h           \
514         socket_proto.h          \
515         socketvar.h             \
516         sockfilter.h            \
517         sockio.h                \
518         soundcard.h             \
519         squeue.h                \
520         squeue_impl.h           \
```

```
521         srn.h                   \
522         sservice.h              \
523         stat.h                  \
524         statfs.h                \
525         statvfs.h               \
526         stdbool.h               \
527         stdint.h                \
528         stermio.h               \
529         stmf.h                  \
530         stmf_defines.h          \
531         stmf_ioctl.h            \
532         stmf_sbd_ioctl.h        \
533         stream.h                \
534         strft.h                 \
535         strlog.h                \
536         strmdep.h               \
537         stropts.h               \
538         strredir.h              \
539         strstat.h               \
540         strsubr.h               \
541         strsun.h                \
542         strtty.h                \
543         sunddi.h                \
544         sunldi.h                \
545         sunldi_impl.h           \
546         sunmdi.h                \
547         sunndi.h                \
548         sunos_dhcp_class.h      \
549         sunpm.h                 \
550         suntpi.h                \
551         suntty.h                \
552         swap.h                  \
553         synch.h                 \
554         sysdc.h                 \
555         sysdc_impl.h            \
556         syscall.h               \
557         sysconf.h               \
558         sysconfig.h             \
559         sysevent.h              \
560         sysevent_impl.h         \
561         sysinfo.h               \
562         syslog.h                \
563         sysmacros.h             \
564         sysmsg_impl.h           \
565         systeminfo.h            \
566         systm.h                 \
567         task.h                  \
568         taskq.h                 \
569         taskq_impl.h            \
570         t_kuser.h               \
571         t_lock.h                \
572         telioctl.h              \
573         termio.h                \
574         termios.h               \
575         termiox.h               \
576         thread.h                \
577         ticlts.h                \
578         ticots.h                \
579         ticotsord.h             \
580         tihdr.h                 \
581         time.h                  \
582         time_impl.h             \
583         time_std_impl.h         \
584         timeb.h                 \
585         timer.h                 \
586         times.h                 \
```



```
587         timex.h                 \
588         timod.h                 \
589         tirdwr.h                \
590         tiuser.h                \
591         tl.h                    \
592         tnf.h                   \
593         tnf_com.h               \
594         tnf_probe.h             \
595         tnf_writer.h            \
596         todio.h                 \
597         tpicommon.h             \
598         ts.h                    \
599         tspriocntl.h            \
600         ttcompat.h              \
601         ttold.h                 \
602         tty.h                   \
603         ttychars.h              \
604         ttydev.h                \
605         tuneable.h              \
606         turnstile.h             \
607         types.h                 \
608         types32.h               \
609         tzfile.h                \
610         u8_textprep.h           \
611         u8_textprep_data.h      \
612         uadmin.h                \
613         ucred.h                 \
614         uio.h                   \
615         ulimit.h                \
616         un.h                    \
617         unistd.h                \
618         user.h                  \
619         ustat.h                 \
620         utime.h                 \
621         utsname.h               \
622         utssys.h                \
623         uuid.h                  \
624         va_impl.h               \
625         va_list.h               \
626         var.h                   \
627         varargs.h               \
628         vfs.h                   \
629         vfs_opreg.h             \
630         vfstab.h                \
631         vgareg.h                \
632         videodev2.h             \
633         visual_io.h             \
634         vlan.h                  \
635         vm.h                    \
636         vm_usage.h              \
637         vmem.h                  \
638         vmem_impl.h             \
639         vmsystm.h               \
640         vnic.h                  \
641         vnic_impl.h             \
642         vnode.h                 \
643         vscan.h                 \
644         vtoc.h                  \
645         vtrace.h                \
646         vuid_event.h            \
647         vuid_wheel.h            \
648         vuid_queue.h            \
649         vuid_state.h            \
650         vuid_store.h            \
651         wait.h                  \
652         waitq.h                 \
```

```
653         wanboot_impl.h          \
654         watchpoint.h            \
655         winlockio.h             \
656         zcons.h                 \
657         zone.h                  \
658         xti_inet.h              \
659         xti_osi.h               \
660         xti_xtiopt.h            \
661         zmod.h

663 HDRS=                           \
664         $(GENHDRS)              \
665         $(CHKHDRS)

667 AUDIOHDRS=                      \
668         ac97.h                  \
669         audio_common.h          \
670         audio_driver.h          \
671         audio_oss.h             \
672         g711.h

674 AVHDRS=                        \
675         iec61883.h

677 BSCHDRS=                       \
678         bscbus.h                \
679         bscv_impl.h             \
680         lom_ebuscodes.h         \
681         lom_io.h                \
682         lom_priv.h              \
683         lombus.h

685 MDESCHDRS=                     \
686         mdesc.h                 \
687         mdesc_impl.h

689 CPUDRVHDRS=                    \
690         cpudrv.h

692 CRYPTOHDRS=                    \
693         elfsign.h               \
694         ioctl.h                 \
695         ioctladmin.h            \
696         common.h                \
697         impl.h                  \
698         spi.h                   \
699         api.h                   \
700         ops_impl.h              \
701         sched_impl.h

703 DCAMHDRS=                      \
704         dcam1394_io.h

706 IBHDRS=                  \
707         ib_types.h      \
708         ib_pkt_hdrs.h

710 IBTLHDRS=                \
711         ibtl_types.h    \
712         ibtl_status.h   \
713         ibti.h          \
714         ibti_cm.h       \
715         ibci.h          \
716         ibti_common.h   \
717         ibvti.h         \
718         ibtl_ci_types.h
```



```
720 IBTLIMPLHDRS=           \
721         ibtl_util.h

723 IBNEXHDRS=              \
724         ibnex_devctl.h

726 IBMFHDRS=               \
727         ibmf.h          \
728         ibmf_msg.h      \
729         ibmf_saa.h      \
730         ibmf_utils.h

732 IBMGTHDRS=              \
733         ib_dm_attr.h    \
734         ib_mad.h        \
735         sm_attr.h       \
736         sa_recs.h

738 IBDHDRS=                \
739         ibd.h

741 OFHDRS=                 \
742         ofa_solaris.h   \
743         ofed_kernel.h

745 RDMAHDRS=               \
746         ib_addr.h       \
747         ib_user_mad.h   \
748         ib_user_sa.h    \
749         ib_user_verbs.h \
750         ib_verbs.h      \
751         rdma_cm.h       \
752         rdma_user_cm.h

754 SOL_UVERBSHDRS=          \
755         sol_uverbs.h    \
756         sol_uverbs2ucma.h       \
757         sol_uverbs_comp.h       \
758         sol_uverbs_hca.h        \
759         sol_uverbs_qp.h         \
760         sol_uverbs_event.h

762 SOL_UMADHDRS=            \
763         sol_umad.h

765 SOL_UCMAHDRS=            \
766         sol_ucma.h      \
767         sol_rdma_user_cm.h

769 SOL_OFSHDRS=            \
770         sol_cma.h       \
771         sol_ib_cma.h    \
772         sol_ofs_common.h        \
773         sol_kverb_impl.h

775 TAVORHDRS=              \
776         tavor_ioctl.h

778 HERMONHDRS=             \
779         hermon_ioctl.h

781 MLNXHDRS=              \
782         mlnx_umap.h

784 IDMHDRS=                \
```

```
785         idm.h           \
786         idm_impl.h      \
787         idm_so.h        \
788         idm_text.h      \
789         idm_transport.h \
790         idm_conn_sm.h

792 ISCSITHDRS=            \
793         radius_packet.h         \
794         radius_protocol.h       \
795         chap.h                  \
796         isns_protocol.h         \
797         iscsi_if.h              \
798         iscsit_common.h

800 ISOHDRS=                       \
801         signal_iso.h

803 DERIVED_LVMHDRS=       \
804         md_mdiox.h              \
805         md_basic.h              \
806         mdmed.h                 \
807         md_mhdx.h               \
808         mdmn_commd.h

810 LVMHDRS=                       \
811         md_convert.h            \
812         md_crc.h                \
813         md_hotspares.h          \
814         md_mddb.h               \
815         md_mirror.h             \
816         md_mirror_shared.h      \
817         md_names.h              \
818         md_notify.h             \
819         md_raid.h               \
820         md_rename.h             \
821         md_sp.h                 \
822         md_stripe.h             \
823         md_trans.h              \
824         mdio.h                  \
825         mdvar.h

827 ALL_LVMHDRS=                    \
828         $(LVMHDRS)              \
829         $(DERIVED_LVMHDRS)

831 FMHDRS=                        \
832         protocol.h              \
833         util.h

835 FMFSHDRS=                      \
836         zfs.h

838 FMIOHDRS=                      \
839         ddi.h                   \
840         disk.h                  \
841         pci.h                   \
842         scsi.h                  \
843         sun4upci.h              \
844         opl_mc_fm.h

846 FSHDRS=                        \
847         autofs.h                \
848         cachefs_dir.h           \
849         cachefs_dlog.h          \
850         cachefs_filegrp.h       \
```



```
851         cachefs_fs.h            \
852         cachefs_fscache.h       \
853         cachefs_ioctl.h         \
854         cachefs_log.h           \
855         decomp.h                \
856         dv_node.h               \
857         sdev_impl.h             \
858         fifonode.h              \
859         hsfs_isospec.h          \
860         hsfs_node.h             \
861         hsfs_rrip.h             \
862         hsfs_spec.h             \
863         hsfs_susp.h             \
864         lofs_info.h             \
865         lofs_node.h             \
866         mntdata.h               \
867         namenode.h              \
868         pc_dir.h                \
869         pc_fs.h                 \
870         pc_label.h              \
871         pc_node.h               \
872         pxfs_ki.h               \
873         snode.h                 \
874         swapnode.h              \
875         tmp.h                   \
876         tmpnode.h               \
877         udf_inode.h             \
878         udf_volume.h            \
879         ufs_acl.h               \
880         ufs_bio.h               \
881         ufs_filio.h             \
882         ufs_fs.h                \
883         ufs_fsdir.h             \
884         ufs_inode.h             \
885         ufs_lockfs.h            \
886         ufs_log.h               \
887         ufs_mount.h             \
888         ufs_panic.h             \
889         ufs_prot.h              \
890         ufs_quota.h             \
891         ufs_snap.h              \
892         ufs_trans.h             \
893         zfs.h                   \
894         zut.h

896 SCSIHDRS=              \
897         scsi.h          \
898         scsi_address.h  \
899         scsi_ctl.h      \
900         scsi_fm.h       \
901         scsi_params.h   \
902         scsi_pkt.h      \
903         scsi_resource.h \
904         scsi_types.h    \
905         scsi_watch.h

907 SCSICONFHDRS=           \
908         autoconf.h      \
909         device.h

911 SCSIGENHDRS=            \
912         commands.h      \
913         dad_mode.h      \
914         inquiry.h       \
915         message.h       \
916         mode.h          \
```

```
917         persist.h       \
918         sense.h         \
919         sff_frames.h    \
920         smp_frames.h    \
921         status.h

923 SCSIIMPLHDRS=            \
924         commands.h      \
925         inquiry.h       \
926         mode.h          \
927         scsi_reset_notify.h \
928         scsi_sas.h      \
929         sense.h         \
930         services.h      \
931         smp_transport.h \
932         spc3_types.h    \
933         status.h        \
934         transport.h     \
935         types.h         \
936         uscsi.h         \
937         usmp.h

939 SCSITARGETSHDRS=         \
940         ses.h           \
941         sesio.h         \
942         sgendef.h       \
943         stdef.h         \
944         sddef.h         \
945         smp.h

947 SCSIADHDRS=

949 SCSICADHDRS=

951 SCSIISCSIHDRS=          \
952         iscsi_door.h    \
953         iscsi_if.h

955 SCSIVHCIHDRS=           \
956         scsi_vhci.h     \
957         mpapi_impl.h    \
958         mpapi_scsi_vhci.h

960 SDCARDHDRS=             \
961         sda.h           \
962         sda_impl.h      \
963         sda_ioctl.h

965 FC4HDRS=                       \
966         fc_transport.h  \
967         linkapp.h       \
968         fc.h            \
969         fcp.h           \
970         fcal_transport.h \
971         fcal.h          \
972         fcal_linkapp.h  \
973         fcio.h

975 FCHDRS=                 \
976         fc.h            \
977         fcio.h          \
978         fc_types.h      \
979         fc_appif.h

981 FCIMPLHDRS=             \
982         fc_error.h      \
```



```
983         fcph.h

985 FCULPHDRS=              \
986         fcp_util.h      \
987         fcsm.h

989 SATAGENHDRS=            \
990         sata_hba.h      \
991         sata_defs.h     \
992         sata_cfgadm.h

994 SYSEVENTHDRS=           \
995         ap_driver.h     \
996         dev.h           \
997         domain.h        \
998         dr.h            \
999         env.h           \
1000        eventdefs.h     \
1001        ipmp.h          \
1002        pwrctl.h        \
1003        svm.h           \
1004        vrrp.h

1006 CONTRACTHDRS=          \
1007        process.h       \
1008        process_impl.h  \
1009        device.h        \
1010        device_impl.h

1012 USBHDRS=        \
1013        usba.h  \
1014        usbai.h

1016 USBAUDHDRS=           \
1017        usb_audio.h

1019 USBHUBDHDRS=          \
1020        hub.h           \
1021        hubd_impl.h

1023 USBHIDHDRS=           \
1024        hid.h

1026 USBMSHDRS=            \
1027        usb_bulkonly.h  \
1028        usb_cbi.h

1030 USBPRNHDRS=          \
1031        usb_printer.h

1033 USBCDCHDRS=           \
1034        usb_cdc.h

1036 USBVIDHDRS=          \
1037        usbvc.h

1039 USBWCMHDRS=           \
1040        usbwcm.h

1042 UGENHDRS=          \
1043        usb_ugen.h

1045 HOTPLUGHDRS=    \
1046        hpcsvc.h        \
1047        hpctrl.h
```

```
1049 HOTPLUGPCIHDRS= \
1050         pcicfg.h        \
1051         pcihp.h

1053 RSMHDRS= \
1054         rsm.h   \
1055         rsm_common.h    \
1056         rsmapi_common.h \
1057         rsmpi.h \
1058         rsmpi_driver.h  \
1059         rsmka_path_int.h

1061 TSOLHDRS=               \
1062         label.h         \
1063         label_macro.h   \
1064         priv.h          \
1065         tndb.h          \
1066         tsyscall.h

1068 I1394HDRS=              \
1069         cmd1394.h       \
1070         id1394.h        \
1071         ieee1212.h      \
1072         ieee1394.h      \
1073         ixl1394.h       \
1074         s1394_impl.h    \
1075         t1394.h

1077 # "cmdk" headers used on sparc
1078 SDKTPHDRS=              \
1079         dadkio.h        \
1080         fdisk.h

1082 # "cmdk" headers used on i386
1083 DKTPHDRS=               \
1084         altsctr.h       \
1085         bbh.h           \
1086         cm.h            \
1087         cmdev.h         \
1088         cmdk.h          \
1089         cmpkt.h         \
1090         controller.h    \
1091         dadev.h         \
1092         dadk.h          \
1093         dadkio.h        \
1094         fctypes.h       \
1095         fdisk.h         \
1096         flowctrl.h      \
1097         gda.h           \
1098         quetypes.h      \
1099         queue.h         \
1100         tgcom.h         \
1101         tgdk.h

1103 # "pc" header files used on i386
1104 PCHDRS=                 \
1105         avintr.h        \
1106         dma_engine.h    \
1107         i8272A.h        \
1108         pcic_reg.h      \
1109         pcic_var.h      \
1110         pic.h           \
1111         pit.h           \
1112         rtc.h

1114 NXGEHDRS=                       \
```



```
1115         nxge.h                  \
1116         nxge_common.h           \
1117         nxge_common_impl.h      \
1118         nxge_defs.h             \
1119         nxge_hw.h               \
1120         nxge_impl.h             \
1121         nxge_ipp.h              \
1122         nxge_ipp_hw.h           \
1123         nxge_mac.h              \
1124         nxge_mac_hw.h           \
1125         nxge_fflp.h             \
1126         nxge_fflp_hw.h          \
1127         nxge_mii.h              \
1128         nxge_rxdma.h            \
1129         nxge_rxdma_hw.h         \
1130         nxge_txc.h              \
1131         nxge_txc_hw.h           \
1132         nxge_txdma.h            \
1133         nxge_txdma_hw.h         \
1134         nxge_virtual.h          \
1135         nxge_espc.h

1137 include Makefile.syshdrs

1139 dcam/%.check:   dcam/%.h
1140         $(DOT_H_CHECK)

1142 CHECKHDRS=                                                      \
1143         $($(MACH)_HDRS:%.h=%.check)                             \
1144         $(AUDIOHDRS:%.h=audio/%.check)                          \
1145         $(AVHDRS:%.h=av/%.check)                                \
1146         $(BSCHDRS:%.h=%.check)                                  \
1147         $(CHKHDRS:%.h=%.check)                                  \
1148         $(CPUDRVHDRS:%.h=%.check)                               \
1149         $(CRYPTOHDRS:%.h=crypto/%.check)                        \
1150         $(DCAMHDRS:%.h=dcam/%.check)                            \
1151         $(FC4HDRS:%.h=fc4/%.check)                              \
1152         $(FCHDRS:%.h=fibre-channel/%.check)                     \
1153         $(FCIMPLHDRS:%.h=fibre-channel/impl/%.check)            \
1154         $(FCULPHDRS:%.h=fibre-channel/ulp/%.check)              \
1155         $(IBHDRS:%.h=ib/%.check)                                \
1156         $(IBDHDRS:%.h=ib/clients/ibd/%.check)                   \
1157         $(IBTLHDRS:%.h=ib/ibtl/%.check)                         \
1158         $(IBTLIMPLHDRS:%.h=ib/ibtl/impl/%.check)                \
1159         $(IBNEXHDRS:%.h=ib/ibnex/%.check)               \
1160         $(IBMGTHDRS:%.h=ib/mgt/%.check)                         \
1161         $(IBMFHDRS:%.h=ib/mgt/ibmf/%.check)                     \
1162         $(OFHDRS:%.h=ib/clients/of/%.check)                             \
1163         $(RDMAHDRS:%.h=ib/clients/of/rdma/%.check)                      \
1164         $(SOL_UVERBSHDRS:%.h=ib/clients/of/sol_uverbs/%.check)  \
1165         $(SOL_UCMAHDRS:%.h=ib/clients/of/sol_ucma/%.check)      \
1166         $(SOL_OFSHDRS:%.h=ib/clients/of/sol_ofs/%.check)        \
1167         $(TAVORHDRS:%.h=ib/adapters/tavor/%.check)      \
1168         $(HERMONHDRS:%.h=ib/adapters/hermon/%.check)    \
1169         $(MLNXHDRS:%.h=ib/adapters/%.check)                     \
1170         $(IDMHDRS:%.h=idm/%.check)                              \
1171         $(ISCSIHDRS:%.h=iscsi/%.check)                          \
1172         $(ISCSITHDRS:%.h=iscsit/%.check)                                \
1173         $(ISOHDRS:%.h=iso/%.check)                              \
1174         $(FMHDRS:%.h=fm/%.check)                                \
1175         $(FMFSHDRS:%.h=fm/fs/%.check)                           \
1176         $(FMIOHDRS:%.h=fm/io/%.check)                           \
1177         $(FSHDRS:%.h=fs/%.check)                                \
1178         $(LVMHDRS:%.h=lvm/%.check)                              \
1179         $(SCSIHDRS:%.h=scsi/%.check)                            \
1180         $(SCSIADHDRS:%.h=scsi/adapters/%.check)                 \
```

```
1181         $(SCSICONFHDRS:%.h=scsi/conf/%.check)           \
1182         $(SCSIIMPLHDRS:%.h=scsi/impl/%.check)           \
1183         $(SCSIISCSIHDRS:%.h=scsi/adapters/%.check)      \
1184         $(SCSIGENHDRS:%.h=scsi/generic/%.check)         \
1185         $(SCSITARGETSHDRS:%.h=scsi/targets/%.check)     \
1186         $(SCSIVHCIHDRS:%.h=scsi/adapters/%.check)       \
1187         $(SATAGENHDRS:%.h=sata/%.check)                 \
1188         $(SDCARDHDRS:%.h=sdcard/%.check)                \
1189         $(SYSEVENTHDRS:%.h=sysevent/%.check)            \
1190         $(CONTRACTHDRS:%.h=contract/%.check)            \
1191         $(USBAUDHDRS:%.h=usb/clients/audio/%.check)   \
1192         $(USBHUBDHDRS:%.h=usb/hubd/%.check)             \
1193         $(USBHIDHDRS:%.h=usb/clients/hid/%.check)       \
1194         $(USBMSHDRS:%.h=usb/clients/mass_storage/%.check)       \
1195         $(USBPRNHDRS:%.h=usb/clients/printer/%.check)   \
1196         $(USBCDCHDRS:%.h=usb/clients/usbcdc/%.check)    \
1197         $(USBVIDHDRS:%.h=usb/clients/video/usbvc/%.check)   \
1198         $(USBWCMHDRS:%.h=usb/clients/usbinput/usbwcm/%.check)   \
1199         $(UGENHDRS:%.h=usb/clients/ugen/%.check)   \
1200         $(USBHDRS:%.h=usb/%.check)                      \
1201         $(I1394HDRS:%.h=1394/%.check)                   \
1202         $(RSMHDRS:%.h=rsm/%.check)                      \
1203         $(TSOLHDRS:%.h=tsol/%.check)                    \
1204         $(NXGEHDRS:%.h=nxge/%.check)


1207 .KEEP_STATE:

1209 .PARALLEL:                      \
1210         $(CHECKHDRS)            \
1211         $(ROOTHDRS)             \
1212         $(ROOTAUDHDRS)          \
1213         $(ROOTAVHDRS)           \
1214         $(ROOTCRYPTOHDRS)       \
1215         $(ROOTDCAMHDRS)         \
1216         $(ROOTISOHDRS)          \
1217         $(ROOTIDMHDRS)          \
1218         $(ROOTISCSIHDRS)        \
1219         $(ROOTISCSITHDRS)       \
1220         $(ROOTFC4HDRS)          \
1221         $(ROOTFCHDRS)           \
1222         $(ROOTFCIMPLHDRS)       \
1223         $(ROOTFCULPHDRS)        \
1224         $(ROOTFMHDRS)           \
1225         $(ROOTFMIOHDRS)         \
1226         $(ROOTFMFSHDRS)         \
1227         $(ROOTFSHDRS)           \
1228         $(ROOTIBDHDRS)          \
1229         $(ROOTIBHDRS)           \
1230         $(ROOTIBTLHDRS)         \
1231         $(ROOTIBTLIMPLHDRS)     \
1232         $(ROOTIBNEXHDRS)        \
1233         $(ROOTIBMGTHDRS)        \
1234         $(ROOTIBMFHDRS)         \
1235         $(ROOTOFHDRS)           \
1236         $(ROOTRDMAHDRS)         \
1237         $(ROOTSOL_OFSHDRS)      \
1238         $(ROOTSOL_UMADHDRS)     \
1239         $(ROOTSOL_UVERBSHDRS)   \
1240         $(ROOTSOL_UCMAHDRS)     \
1241         $(ROOTTAVORHDRS)        \
1242         $(ROOTHERMONHDRS)       \
1243         $(ROOTMLNXHDRS)         \
1244         $(ROOTLVMHDRS)          \
1245         $(ROOTSCSIHDRS)         \
1246         $(ROOTSCSIADHDRS)       \
```



```
1247         $(ROOTSCSICONFHDRS)     \
1248         $(ROOTSCSIISCSIHDRS)    \
1249         $(ROOTSCSIGENHDRS)      \
1250         $(ROOTSCSIIMPLHDRS)     \
1251         $(ROOTSCSIVHCIHDRS)     \
1252         $(ROOTSDCARDHDRS)       \
1253         $(ROOTSYSEVENTHDRS)     \
1254         $(ROOTCONTRACTHDRS)     \
1255         $(ROOTUSBHDRS)          \
1256         $(ROOTUWBHDRS)          \
1257         $(ROOTUWBAHDRS)         \
1258         $(ROOTUSBAUDHDRS)       \
1259         $(ROOTUSBHUBDHDRS)      \
1260         $(ROOTUSBHIDHDRS)       \
1261         $(ROOTUSBHRCHDRS)       \
1262         $(ROOTUSBMSHDRS)        \
1263         $(ROOTUSBPRNHDRS)       \
1264         $(ROOTUSBCDCHDRS)       \
1265         $(ROOTUSBVIDHDRS)       \
1266         $(ROOTUSBWCMHDRS)       \
1267         $(ROOTUGENHDRS)         \
1268         $(ROOT1394HDRS)         \
1269         $(ROOTHOTPLUGHDRS)      \
1270         $(ROOTHOTPLUGPCIHDRS)   \
1271         $(ROOTRSMHDRS)          \
1272         $(ROOTTSOLHDRS)         \
1273         $($(MACH)_ROOTHDRS)


1276 install_h:                      \
1277         $(ROOTDIRS)             \
1278         LVMDERIVED_H            \
1279         .WAIT                   \
1280         $(ROOTHDRS)             \
1281         $(ROOTAUDHDRS)          \
1282         $(ROOTAVHDRS)           \
1283         $(ROOTCRYPTOHDRS)       \
1284         $(ROOTDCAMHDRS)         \
1285         $(ROOTISOHDRS)          \
1286         $(ROOTIDMHDRS)          \
1287         $(ROOTISCSIHDRS)        \
1288         $(ROOTISCSITHDRS)       \
1289         $(ROOTFC4HDRS)          \
1290         $(ROOTFCHDRS)           \
1291         $(ROOTFCIMPLHDRS)       \
1292         $(ROOTFCULPHDRS)        \
1293         $(ROOTFMHDRS)           \
1294         $(ROOTFMFSHDRS)         \
1295         $(ROOTFMIOHDRS)         \
1296         $(ROOTFSHDRS)           \
1297         $(ROOTIBDHDRS)          \
1298         $(ROOTIBHDRS)           \
1299         $(ROOTIBTLHDRS)         \
1300         $(ROOTIBTLIMPLHDRS)     \
1301         $(ROOTIBNEXHDRS)        \
1302         $(ROOTIBMGTHDRS)        \
1303         $(ROOTIBMFHDRS)         \
1304         $(ROOTOFHDRS)           \
1305         $(ROOTRDMAHDRS)         \
1306         $(ROOTSOL_OFSHDRS)      \
1307         $(ROOTSOL_UMADHDRS)     \
1308         $(ROOTSOL_UVERBSHDRS)   \
1309         $(ROOTSOL_UCMAHDRS)     \
1310         $(ROOTTAVORHDRS)        \
1311         $(ROOTHERMONHDRS)       \
1312         $(ROOTMLNXHDRS)         \
```

```
1313         $(ROOTLVMHDRS)          \
1314         $(ROOTSCSIHDRS)         \
1315         $(ROOTSCSIADHDRS)       \
1316         $(ROOTSCSIISCSIHDRS)    \
1317         $(ROOTSCSICONFHDRS)     \
1318         $(ROOTSCSIGENHDRS)      \
1319         $(ROOTSCSIIMPLHDRS)     \
1320         $(ROOTSCSIVHCIHDRS)     \
1321         $(ROOTSDCARDHDRS)       \
1322         $(ROOTSYSEVENTHDRS)     \
1323         $(ROOTCONTRACTHDRS)     \
1324         $(ROOTUWBHDRS)          \
1325         $(ROOTUWBAHDRS)         \
1326         $(ROOTUSBHDRS)          \
1327         $(ROOTUSBAUDHDRS)       \
1328         $(ROOTUSBHUBDHDRS)      \
1329         $(ROOTUSBHIDHDRS)       \
1330         $(ROOTUSBHRCHDRS)       \
1331         $(ROOTUSBMSHDRS)        \
1332         $(ROOTUSBPRNHDRS)       \
1333         $(ROOTUSBCDCHDRS)       \
1334         $(ROOTUSBVIDHDRS)       \
1335         $(ROOTUSBWCMHDRS)       \
1336         $(ROOTUGENHDRS)         \
1337         $(ROOT1394HDRS)         \
1338         $(ROOTHOTPLUGHDRS)      \
1339         $(ROOTHOTPLUGPCIHDRS)   \
1340         $(ROOTRSMHDRS)          \
1341         $(ROOTTSOLHDRS)         \
1342         $($(MACH)_ROOTHDRS)

1344 all_h: $(GENHDRS)

1346 priv_const.h: $(PRIVS_AWK) $(PRIVS_DEF)
1347         $(NAWK) -f $(PRIVS_AWK) < $(PRIVS_DEF) -v privhfile=$@

1349 priv_names.h: $(PRIVS_AWK) $(PRIVS_DEF)
1350         $(NAWK) -f $(PRIVS_AWK) < $(PRIVS_DEF) -v pubhfile=$@

1352 usb/usbdevs.h: $(USBDEVS_AWK) $(USBDEVS_DATA)
1353         $(NAWK) -f $(USBDEVS_AWK) $(USBDEVS_DATA) -H > $@

1355 LVMDERIVED_H:
1356         cd $(SRC)/uts/common/sys/lvm; pwd; $(MAKE) all_h

1358 clean:
1359         $(RM) $(GENHDRS)

1361 clobber: clean
1362         cd $(SRC)/uts/common/sys/lvm; pwd; $(MAKE) clobber

1364 check:  $(CHECKHDRS)

1366 FRC:
```

****
6245 Wed May 27 19:49:32 2015
new/usr/src/uts/common/sys/auxv.h
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the stack and the heap. Introduce a service, svc:/system/process-security, and a tool psecflags(1) to control and observe it
****
_____unchanged_portion_omitted_

```
 66 #endif  /* _SYSCALL32 */

 68 #endif /* _ASM */

 70 #define AT_NULL         0
 71 #define AT_IGNORE       1
 72 #define AT_EXECFD       2
 73 #define AT_PHDR         3       /* &phdr[0] */
 74 #define AT_PHENT        4       /* sizeof(phdr[0]) */
 75 #define AT_PHNUM        5       /* # phdr entries */
 76 #define AT_PAGESZ       6       /* getpagesize(2) */
 77 #define AT_BASE         7       /* ld.so base addr */
 78 #define AT_FLAGS        8       /* processor flags */
 79 #define AT_ENTRY        9       /* a.out entry point */

 81 /*
 82  * These relate to the original PPC ABI document; Linux reused
 83  * the values for other things (see below), so disambiguation of
 84  * these values may require additional context in PPC binaries.
 85  *
 86  * AT_DCACHEBSIZE       10      smallest data cache block size
 87  * AT_ICACHEBSIZE       11      smallest instruction cache block size
 88  * AT_UCACHEBSIZE       12      smallest unified cache block size
 89  *
 90  * These are the values from LSB 1.3, the first five are also described
 91  * in the draft amd64 ABI.
 92  *
 93  * At the time of writing, Solaris doesn't place any of these values into
 94  * the aux vector, except AT_CLKTCK which is placed on the aux vector for
 95  * lx branded processes; also, we do similar things via AT_SUN_ values.
 96  *
 97  * AT_NOTELF            10      program is not ELF?
 98  * AT_UID               11      real user id
 99  * AT_EUID              12      effective user id
100  * AT_GID               13      real group id
101  * AT_EGID              14      effective group id
102  *
103  * AT_PLATFORM          15
104  * AT_HWCAP             16
105  * AT_CLKTCK            17      c.f. _SC_CLK_TCK
106  * AT_FPUCW             18
107  *
108  * AT_DCACHEBSIZE       19      (moved from 10)
109  * AT_ICACHEBSIZE       20      (moved from 11)
110  * AT_UCACHEBSIZE       21      (moved from 12)
111  *
112  * AT_IGNOREPPC         22
113  */

115 /*
116  * Sun extensions begin here
117  */
118 #define AT_SUN_UID      2000    /* effective user id */
119 #define AT_SUN_RUID     2001    /* real user id */
120 #define AT_SUN_GID      2002    /* effective group id */
121 #define AT_SUN_RGID     2003    /* real group id */
```



```
123 /*
124  * The following attributes are specific to the
125  * kernel implementation of the linker/loader.
126  */
127 #define AT_SUN_LDELF    2004    /* dynamic linker's ELF header */
128 #define AT_SUN_LDSHDR   2005    /* dynamic linker's section headers */
129 #define AT_SUN_LDNAME   2006    /* name of dynamic linker */
130 #define AT_SUN_LPAGESZ  2007    /* large pagesize */
131 /*
132  * The following aux vector provides a null-terminated platform
133  * identification string. This information is the same as provided
134  * by sysinfo(2) when invoked with the command SI_PLATFORM.
135  */
136 #define AT_SUN_PLATFORM 2008    /* platform name */

138 /*
139  * These attributes communicate performance -hints- about processor
140  * hardware capabilities that might be useful to library implementations.
141  */
142 #define AT_SUN_HWCAP    2009
143 #define AT_SUN_HWCAP2   2023

145 #if defined(_KERNEL)
146 /*
147  * User info regarding machine attributes, respectively reported to native and
148  * non-native user apps.
149  */
150 extern uint_t auxv_hwcap;
151 extern uint_t auxv_hwcap_2;
152 #if defined(_SYSCALL32)
153 extern uint_t auxv_hwcap32;
154 extern uint_t auxv_hwcap32_2;
155 #endif /* _SYSCALL32 */
156 #else
157 extern uint_t getisax(uint32_t *, uint_t);
158 #endif  /* _KERNEL */

160 #define AT_SUN_IFLUSH   2010    /* flush icache? */
161 #define AT_SUN_CPU      2011    /* cpu name */

163 /*
164  * The following aux vector provides a pointer to a null-terminated
165  * path name, a copy of the path name passed to the exec() system
166  * call but that has had all symlinks resolved (see resolvepath(2)).
167  */
168 #define AT_SUN_EXECNAME 2014    /* exec() path name */

170 #define AT_SUN_MMU      2015    /* mmu module name */
171 #define AT_SUN_LDDATA   2016    /* dynamic linkers data segment */

173 #define AT_SUN_AUXFLAGS 2017    /* AF_SUN_ flags passed from the kernel */

175 /*
176  * Used to indicate to the runtime linker the name of the emulation binary,
177  * if one is being used. For brands, this is the name of the brand library.
178  */
179 #define AT_SUN_EMULATOR         2018

181 #define AT_SUN_BRANDNAME        2019

183 /*
184  * Aux vectors available for brand modules.
185  */
186 #define AT_SUN_BRAND_AUX1       2020
187 #define AT_SUN_BRAND_AUX2       2021
```

```
 188 #define AT_SUN_BRAND_AUX3       2022

 190 /*
 191  * Note that 2023 is reserved for the AT_SUN_HWCAP2 word defined above.
 192  */

 194 #define AT_SUN_SECFLAGS         2024

 196 #endif /* ! codereview */
 197 /*
 198  * The kernel is in a better position to determine whether a process needs to
 199  * ignore dangerous LD environment variables.  If set, this flags tells
 200  * ld.so.1 to run "secure" and ignore the the environment.
 201  */
 202 #define AF_SUN_SETUGID          0x00000001

 204 /*
 205  * If set, this flag indicates that hardware capabilites can be verified
 206  * against the AT_SUN_HWCAP value.
 207  */
 208 #define AF_SUN_HWCAPVERIFY      0x00000002

 210 /*
 211  * If set, this flag indicates that the the linker should not initialize
 212  * any of its link maps as primary link wrt the unified libc threading
 213  * interfaces.
 214  */
 215 #define AF_SUN_NOPLM            0x00000004

 217 #ifdef  __cplusplus
 218 }
 219 #endif

 221 #if defined(_AUXV_TARGET_ALL) || defined(_AUXV_TARGET_SPARC) || defined(__sparc)
 222 #include <sys/auxv_SPARC.h>
 223 #endif

 225 #if defined(_AUXV_TARGET_ALL) || defined(_AUXV_TARGET_386) || defined(__x86)
 226 #include <sys/auxv_386.h>
 227 #endif

 229 #endif  /* _SYS_AUXV_H */
```

```
**********************************************************
   23385 Wed May 27 19:49:33 2015
new/usr/src/uts/common/sys/link.h
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
______unchanged_portion_omitted_
  69 #endif  /* defined(_LP64) || defined(_LONGLONG_TYPE) */
  70 #endif  /* _ASM */

  72 /*
  73  * Tag values
  74  */
  75 #define DT_NULL         0       /* last entry in list */
  76 #define DT_NEEDED       1       /* a needed object */
  77 #define DT_PLTRELSZ     2       /* size of relocations for the PLT */
  78 #define DT_PLTGOT       3       /* addresses used by procedure linkage table */
  79 #define DT_HASH         4       /* hash table */
  80 #define DT_STRTAB       5       /* string table */
  81 #define DT_SYMTAB       6       /* symbol table */
  82 #define DT_RELA         7       /* addr of relocation entries */
  83 #define DT_RELASZ       8       /* size of relocation table */
  84 #define DT_RELAENT      9       /* base size of relocation entry */
  85 #define DT_STRSZ        10      /* size of string table */
  86 #define DT_SYMENT       11      /* size of symbol table entry */
  87 #define DT_INIT         12      /* _init addr */
  88 #define DT_FINI         13      /* _fini addr */
  89 #define DT_SONAME       14      /* name of this shared object */
  90 #define DT_RPATH        15      /* run-time search path */
  91 #define DT_SYMBOLIC     16      /* shared object linked -Bsymbolic */
  92 #define DT_REL          17      /* addr of relocation entries */
  93 #define DT_RELSZ        18      /* size of relocation table */
  94 #define DT_RELENT       19      /* base size of relocation entry */
  95 #define DT_PLTREL       20      /* relocation type for PLT entry */
  96 #define DT_DEBUG        21      /* pointer to r_debug structure */
  97 #define DT_TEXTREL      22      /* text relocations remain for this object */
  98 #define DT_JMPREL       23      /* pointer to the PLT relocation entries */
  99 #define DT_BIND_NOW     24      /* perform all relocations at load of object */
 100 #define DT_INIT_ARRAY   25      /* pointer to .init_array */
 101 #define DT_FINI_ARRAY   26      /* pointer to .fini_array */
 102 #define DT_INIT_ARRAYSZ 27      /* size of .init_array */
 103 #define DT_FINI_ARRAYSZ 28      /* size of .fini_array */
 104 #define DT_RUNPATH      29      /* run-time search path */
 105 #define DT_FLAGS        30      /* state flags - see DF_* */

 107 /*
 108  * DT_* encoding rules: The value of each dynamic tag determines the
 109  * interpretation of the d_un union. This convention provides for simpler
 110  * interpretation of dynamic tags by external tools. A tag whose value
 111  * is an even number indicates a dynamic section entry that uses d_ptr.
 112  * A tag whose value is an odd number indicates a dynamic section entry
 113  * that uses d_val, or that uses neither d_ptr nor d_val.
 114  *
 115  * There are exceptions to the above rule:
 116  *      - Tags with values that are less than DT_ENCODING.
 117  *      - Tags with values that fall between DT_LOOS and DT_SUNW_ENCODING
 118  *      - Tags with values that fall between DT_HIOS and DT_LOPROC
 119  *
 120  * Third party tools must handle these exception ranges explicitly
 121  * on an item by item basis.
 122  */
 123 #define DT_ENCODING             32      /* positive tag DT_* encoding rules */
 124                                         /*      start after this */
 125 #define DT_PREINIT_ARRAY        32      /* pointer to .preinit_array */
```



```
 126 #define DT_PREINIT_ARRAYSZ      33      /* size of .preinit_array */

 128 #define DT_MAXPOSTAGS           34      /* number of positive tags */

 130 /*
 131  * DT_* encoding rules do not apply between DT_LOOS and DT_SUNW_ENCODING
 132  */
 133 #define DT_LOOS                 0x6000000d      /* OS specific range */
 134 #define DT_SUNW_AUXILIARY       0x6000000d      /* symbol auxiliary name */
 135 #define DT_SUNW_RTLDINF         0x6000000e      /* ld.so.1 info (private) */
 136 #define DT_SUNW_FILTER          0x6000000f      /* symbol filter name */
 137 #define DT_SUNW_CAP             0x60000010      /* hardware/software */
 138                                                 /*      capabilities */
 139 #define DT_SUNW_SYMTAB          0x60000011      /* symtab with local fcn */
 140                                                 /*      symbols immediately */
 141                                                 /*      preceding DT_SYMTAB */
 142 #define DT_SUNW_SYMSZ           0x60000012      /* Size of SUNW_SYMTAB table */

 144 /*
 145  * DT_* encoding rules apply between DT_SUNW_ENCODING and DT_HIOS
 146  */
 147 #define DT_SUNW_ENCODING        0x60000013      /* DT_* encoding rules resume */
 148                                                 /*      after this */
 149 #define DT_SUNW_SORTENT         0x60000013      /* sizeof [SYM|TLS]SORT entry */
 150 #define DT_SUNW_SYMSORT         0x60000014      /* sym indices sorted by addr */
 151 #define DT_SUNW_SYMSORTSZ       0x60000015      /* size of SUNW_SYMSORT */
 152 #define DT_SUNW_TLSSORT         0x60000016      /* tls sym ndx sort by offset */
 153 #define DT_SUNW_TLSSORTSZ       0x60000017      /* size of SUNW_TLSSORT */
 154 #define DT_SUNW_CAPINFO         0x60000018      /* capabilities symbols */
 155 #define DT_SUNW_STRPAD          0x60000019      /* # of unused bytes at the */
 156                                                 /*      end of dynstr */
 157 #define DT_SUNW_CAPCHAIN        0x6000001a      /* capabilities chain info */
 158 #define DT_SUNW_LDMACH          0x6000001b      /* EM_ machine code of linker */
 159                                                 /*      that produced object */
 160 #define DT_SUNW_CAPCHAINENT     0x6000001d      /* capabilities chain entry */
 161 #define DT_SUNW_CAPCHAINSZ      0x6000001f      /* capabilities chain size */
 162 /* 0x60000021 would be DT_SUNW_PARENT */
 163 #define DT_SUNW_ASLR            0x60000023      /* executable ASLR desire */
 164 #endif /* ! codereview */

 166 /*
 167  * DT_* encoding rules do not apply between DT_HIOS and DT_LOPROC
 168  */
 169 #define DT_HIOS                 0x6ffff000

 171 /*
 172  * The following values have been deprecated and remain here to allow
 173  * compatibility with older binaries.
 174  */
 175 #define DT_DEPRECATED_SPARC_REGISTER    0x7000001

 177 /*
 178  * DT_* entries which fall between DT_VALRNGHI & DT_VALRNGLO use the
 179  * Dyn.d_un.d_val field of the Elf*_Dyn structure.
 180  */
 181 #define DT_VALRNGLO     0x6ffffd00

 183 #define DT_GNU_PRELINKED 0x6ffffdf5     /* prelinking timestamp (unused) */
 184 #define DT_GNU_CONFLICTSZ 0x6ffffdf6    /* size of conflict section (unused) */
 185 #define DT_GNU_LIBLISTSZ 0x6ffffdf7     /* size of library list (unused) */
 186 #define DT_CHECKSUM     0x6ffffdf8      /* elf checksum */
 187 #define DT_PLTPADSZ     0x6ffffdf9      /* pltpadding size */
 188 #define DT_MOVEENT      0x6ffffdfa      /* move table entry size */
 189 #define DT_MOVESZ       0x6ffffdfb      /* move table size */
 190 #define DT_FEATURE_1    0x6ffffdfc      /* feature holder (unused) */
 191 #define DT_POSFLAG_1    0x6ffffdfd      /* flags for DT_* entries, effecting */
```

```
192                                     /*      the following DT_* entry. */
193                                     /*      See DF_P1_* definitions */
194 #define DT_SYMINSZ      0x6ffffdfe      /* syminfo table size (in bytes) */
195 #define DT_SYMINENT     0x6ffffdff      /* syminfo entry size (in bytes) */
196 #define DT_VALRNGHI     0x6ffffdff

198 /*
199  * DT_* entries which fall between DT_ADDRRNGHI & DT_ADDRRNGLO use the
200  * Dyn.d_un.d_ptr field of the Elf*_Dyn structure.
201  *
202  * If any adjustment is made to the ELF object after it has been
203  * built, these entries will need to be adjusted.
204  */
205 #define DT_ADDRRNGLO    0x6ffffe00

207 #define DT_GNU_HASH     0x6ffffef5      /* GNU-style hash table (unused) */
208 #define DT_TLSDESC_PLT  0x6ffffef6      /* GNU (unused) */
209 #define DT_TLSDESC_GOT  0x6ffffef7      /* GNU (unused) */
210 #define DT_GNU_CONFLICT 0x6ffffef8      /* start of conflict section (unused) */
211 #define DT_GNU_LIBLIST  0x6ffffef9      /* Library list (unused) */

213 #define DT_CONFIG       0x6ffffefa      /* configuration information */
214 #define DT_DEPAUDIT     0x6ffffefb      /* dependency auditing */
215 #define DT_AUDIT        0x6ffffefc      /* object auditing */
216 #define DT_PLTPAD       0x6ffffefd      /* pltpadding (sparcv9) */
217 #define DT_MOVETAB      0x6ffffefe      /* move table */
218 #define DT_SYMINFO      0x6ffffeff      /* syminfo table */
219 #define DT_ADDRRNGHI    0x6ffffeff

221 /*
222  * The following DT_* entries should have been assigned within one of the
223  * DT_* ranges, but existed before such ranges had been established.
224  */
225 #define DT_VERSYM       0x6ffffff0      /* version symbol table - unused by */
226                                         /*      Solaris (see libld/update.c) */

228 #define DT_RELACOUNT    0x6ffffff9      /* number of RELATIVE relocations */
229 #define DT_RELCOUNT     0x6ffffffa      /* number of RELATIVE relocations */
230 #define DT_FLAGS_1      0x6ffffffb      /* state flags - see DF_1_* defs */
231 #define DT_VERDEF       0x6ffffffc      /* version definition table and */
232 #define DT_VERDEFNUM    0x6ffffffd      /*      associated no. of entries */
233 #define DT_VERNEED      0x6ffffffe      /* version needed table and */
234 #define DT_VERNEEDNUM   0x6fffffff      /*      associated no. of entries */

236 /*
237  * DT_* entries between DT_HIPROC and DT_LOPROC are reserved for processor
238  * specific semantics.
239  *
240  * DT_* encoding rules apply to all tag values larger than DT_LOPROC.
241  */
242 #define DT_LOPROC       0x70000000      /* processor specific range */
243 #define DT_AUXILIARY    0x7ffffffd      /* shared library auxiliary name */
244 #define DT_USED         0x7ffffffe      /* ignored - same as needed */
245 #define DT_FILTER       0x7fffffff      /* shared library filter name */
246 #define DT_HIPROC       0x7fffffff


249 /*
250  * Values for DT_FLAGS
251  */
252 #define DF_ORIGIN       0x00000001      /* ORIGIN processing required */
253 #define DF_SYMBOLIC     0x00000002      /* symbolic bindings in effect */
254 #define DF_TEXTREL      0x00000004      /* text relocations remain */
255 #define DF_BIND_NOW     0x00000008      /* process all relocations */
256 #define DF_STATIC_TLS   0x00000010      /* obj. contains static TLS refs */
```



```
258 /*
259  * Values for the DT_POSFLAG_1 .dynamic entry.
260  * These values only affect the following DT_* entry.
261  */
262 #define DF_P1_LAZYLOAD  0x00000001      /* following object is to be */
263                                         /*      lazy loaded */
264 #define DF_P1_GROUPPERM 0x00000002      /* following object's symbols are */
265                                         /*      not available for general */
266                                         /*      symbol bindings. */
267 #define DF_P1_DEFERRED  0x00000004      /* following object is deferred */

269 /*
270  * Values for the DT_FLAGS_1 .dynamic entry.
271  */
272 #define DF_1_NOW        0x00000001      /* set RTLD_NOW for this object */
273 #define DF_1_GLOBAL     0x00000002      /* set RTLD_GLOBAL for this object */
274 #define DF_1_GROUP      0x00000004      /* set RTLD_GROUP for this object */
275 #define DF_1_NODELETE   0x00000008      /* set RTLD_NODELETE for this object */
276 #define DF_1_LOADFLTR   0x00000010      /* trigger filtee loading at runtime */
277 #define DF_1_INITFIRST  0x00000020      /* set RTLD_INITFIRST for this object */
278 #define DF_1_NOOPEN     0x00000040      /* set RTLD_NOOPEN for this object */
279 #define DF_1_ORIGIN     0x00000080      /* ORIGIN processing required */
280 #define DF_1_DIRECT     0x00000100      /* direct binding enabled */
281 #define DF_1_TRANS      0x00000200      /* unused obsolete name */
282 #define DF_1_INTERPOSE  0x00000400      /* object is an interposer */
283 #define DF_1_NODEFLIB   0x00000800      /* ignore default library search path */
284 #define DF_1_NODUMP     0x00001000      /* object can't be dldump(3x)'ed */
285 #define DF_1_CONFALT    0x00002000      /* configuration alternative created */
286 #define DF_1_ENDFILTEE  0x00004000      /* filtee terminates filters search */
287 #define DF_1_DISPRELDNE 0x00008000      /* disp reloc applied at build time */
288 #define DF_1_DISPRELPND 0x00010000      /* disp reloc applied at run-time */
289 #define DF_1_NODIRECT   0x00020000      /* object contains symbols that */
290                                         /*      cannot be directly bound to */
291 #define DF_1_IGNMULDEF  0x00040000      /* internal: krtld ignore muldefs */
292 #define DF_1_NOKSYMS    0x00080000      /* internal: don't export object's */
293                                         /*      symbols via /dev/ksyms */
294 #define DF_1_NOHDR      0x00100000      /* mapfile: 1st segment mapping */
295                                         /*      omits ELF & program headers */
296 #define DF_1_EDITED     0x00200000      /* object has been modified since */
297                                         /*      being built by 'ld' */
298 #define DF_1_NORELOC    0x00400000      /* internal: unrelocated object */
299 #define DF_1_SYMINTPOSE 0x00800000      /* individual symbol interposers */
300                                         /*      exist */
301 #define DF_1_GLOBAUDIT  0x01000000      /* establish global auditing */
302 #define DF_1_SINGLETON  0x02000000      /* singleton symbols exist */

304 /*
305  * Values set to DT_FEATURE_1 tag's d_val (unused obsolete tag)
306  */
307 #define DTF_1_PARINIT   0x00000001      /* partially initialization feature */
308 #define DTF_1_CONFEXP   0x00000002      /* configuration file expected */


311 /*
312  * Version structures.  There are three types of version structure:
313  *
314  *  o   A definition of the versions within the image itself.
315  *      Each version definition is assigned a unique index (starting from
316  *      VER_NDX_BGNDEF) which is used to cross-reference symbols associated to
317  *      the version.  Each version can have one or more dependencies on other
318  *      version definitions within the image.  The version name, and any
319  *      dependency names, are specified in the version definition auxiliary
320  *      array.  Version definition entries require a version symbol index table.
321  *
322  *  o   A version requirement on a needed dependency.  Each needed entry
323  *      specifies the shared object dependency (as specified in DT_NEEDED).
```

```
 *      One or more versions required from this dependency are specified in the
 *      version needed auxiliary array.
 *
 *  o   A version symbol index table.  Each symbol indexes into this array
 *      to determine its version index.  Index values of VER_NDX_BGNDEF or
 *      greater indicate the version definition to which a symbol is associated.
 *      (the size of a symbol index entry is recorded in the sh_info field).
 */
#ifndef _ASM

typedef struct {                        /* Version Definition Structure. */
        Elf32_Half      vd_version;     /* this structures version revision */
        Elf32_Half      vd_flags;       /* version information */
        Elf32_Half      vd_ndx;         /* version index */
        Elf32_Half      vd_cnt;         /* no. of associated aux entries */
        Elf32_Word      vd_hash;        /* version name hash value */
        Elf32_Word      vd_aux;         /* no. of bytes from start of this */
                                        /*      verdef to verdaux array */
        Elf32_Word      vd_next;        /* no. of bytes from start of this */
} Elf32_Verdef;                         /*      verdef to next verdef entry */

typedef struct {                        /* Verdef Auxiliary Structure. */
        Elf32_Word      vda_name;       /* first element defines the version */
                                        /*      name. Additional entries */
                                        /*      define dependency names. */
        Elf32_Word      vda_next;       /* no. of bytes from start of this */
} Elf32_Verdaux;                        /*      verdaux to next verdaux entry */


typedef struct {                        /* Version Requirement Structure. */
        Elf32_Half      vn_version;     /* this structures version revision */
        Elf32_Half      vn_cnt;         /* no. of associated aux entries */
        Elf32_Word      vn_file;        /* name of needed dependency (file) */
        Elf32_Word      vn_aux;         /* no. of bytes from start of this */
                                        /*      verneed to vernaux array */
        Elf32_Word      vn_next;        /* no. of bytes from start of this */
} Elf32_Verneed;                        /*      verneed to next verneed entry */

typedef struct {                        /* Verneed Auxiliary Structure. */
        Elf32_Word      vna_hash;       /* version name hash value */
        Elf32_Half      vna_flags;      /* version information */
        Elf32_Half      vna_other;
        Elf32_Word      vna_name;       /* version name */
        Elf32_Word      vna_next;       /* no. of bytes from start of this */
} Elf32_Vernaux;                        /*      vernaux to next vernaux entry */

typedef Elf32_Half      Elf32_Versym;   /* Version symbol index array */

typedef struct {
        Elf32_Half      si_boundto;     /* direct bindings - symbol bound to */
        Elf32_Half      si_flags;       /* per symbol flags */
} Elf32_Syminfo;


#if defined(_LP64) || defined(_LONGLONG_TYPE)
typedef struct {
        Elf64_Half      vd_version;     /* this structures version revision */
        Elf64_Half      vd_flags;       /* version information */
        Elf64_Half      vd_ndx;         /* version index */
        Elf64_Half      vd_cnt;         /* no. of associated aux entries */
        Elf64_Word      vd_hash;        /* version name hash value */
        Elf64_Word      vd_aux;         /* no. of bytes from start of this */
                                        /*      verdef to verdaux array */
        Elf64_Word      vd_next;        /* no. of bytes from start of this */
} Elf64_Verdef;                         /*      verdef to next verdef entry */
```



```
typedef struct {
        Elf64_Word      vda_name;       /* first element defines the version */
                                        /*      name. Additional entries */
                                        /*      define dependency names. */
        Elf64_Word      vda_next;       /* no. of bytes from start of this */
} Elf64_Verdaux;                        /*      verdaux to next verdaux entry */

typedef struct {
        Elf64_Half      vn_version;     /* this structures version revision */
        Elf64_Half      vn_cnt;         /* no. of associated aux entries */
        Elf64_Word      vn_file;        /* name of needed dependency (file) */
        Elf64_Word      vn_aux;         /* no. of bytes from start of this */
                                        /*      verneed to vernaux array */
        Elf64_Word      vn_next;        /* no. of bytes from start of this */
} Elf64_Verneed;                        /*      verneed to next verneed entry */

typedef struct {
        Elf64_Word      vna_hash;       /* version name hash value */
        Elf64_Half      vna_flags;      /* version information */
        Elf64_Half      vna_other;
        Elf64_Word      vna_name;       /* version name */
        Elf64_Word      vna_next;       /* no. of bytes from start of this */
} Elf64_Vernaux;                        /*      vernaux to next vernaux entry */

typedef Elf64_Half      Elf64_Versym;

typedef struct {
        Elf64_Half      si_boundto;     /* direct bindings - symbol bound to */
        Elf64_Half      si_flags;       /* per symbol flags */
} Elf64_Syminfo;
#endif  /* defined(_LP64) || defined(_LONGLONG_TYPE) */

#endif  /* _ASM */

/*
 * Versym symbol index values.  Values greater than VER_NDX_GLOBAL
 * and less then VER_NDX_LORESERVE associate symbols with user
 * specified version descriptors.
 */
#define VER_NDX_LOCAL           0       /* symbol is local */
#define VER_NDX_GLOBAL          1       /* symbol is global and assigned to */
                                        /*      the base version */
#define VER_NDX_LORESERVE       0xff00  /* beginning of RESERVED entries */
#define VER_NDX_ELIMINATE       0xff01  /* symbol is to be eliminated */

/*
 * Verdef (vd_flags) and Vernaux (vna_flags) flags values.
 */
#define VER_FLG_BASE            0x1     /* version definition of file itself */
                                        /*      (Verdef only) */
#define VER_FLG_WEAK            0x2     /* weak version identifier */
#define VER_FLG_INFO            0x4     /* version is recorded in object for */
                                        /*      informational purposes */
                                        /*      (Versym reference) only. No */
                                        /*      runtime verification is */
                                        /*      required. (Vernaux only) */

/*
 * Verdef version values.
 */
#define VER_DEF_NONE            0       /* Ver_def version */
#define VER_DEF_CURRENT         1
#define VER_DEF_NUM             2

/*
 * Verneed version values.
```

```
 */
#define VER_NEED_NONE           0       /* Ver_need version */
#define VER_NEED_CURRENT        1
#define VER_NEED_NUM            2


/*
 * Syminfo flag values
 */
#define SYMINFO_FLG_DIRECT      0x0001  /* symbol ref has direct association */
                                        /*      to object containing defn. */
#define SYMINFO_FLG_FILTER      0x0002  /* symbol ref is associated to a */
                                        /*      standard filter */
#define SYMINFO_FLG_PASSTHRU    SYMINFO_FLG_FILTER /* unused obsolete name */
#define SYMINFO_FLG_COPY        0x0004  /* symbol is a copy-reloc */
#define SYMINFO_FLG_LAZYLOAD    0x0008  /* object containing defn. should be */
                                        /*      lazily-loaded */
#define SYMINFO_FLG_DIRECTBIND  0x0010  /* ref should be bound directly to */
                                        /*      object containing defn. */
#define SYMINFO_FLG_NOEXTDIRECT 0x0020  /* don't let an external reference */
                                        /*      directly bind to this symbol */
#define SYMINFO_FLG_AUXILIARY   0x0040  /* symbol ref is associated to a */
                                        /*      auxiliary filter */
#define SYMINFO_FLG_INTERPOSE   0x0080  /* symbol defines an interposer */
#define SYMINFO_FLG_CAP         0x0100  /* symbol is capabilities specific */
#define SYMINFO_FLG_DEFERRED    0x0200  /* symbol should not be included in */
                                        /*      BIND_NOW relocations */

/*
 * Syminfo.si_boundto values.
 */
#define SYMINFO_BT_SELF         0xffff  /* symbol bound to self */
#define SYMINFO_BT_PARENT       0xfffe  /* symbol bound to parent */
#define SYMINFO_BT_NONE         0xfffd  /* no special symbol binding */
#define SYMINFO_BT_EXTERN       0xfffc  /* symbol defined as external */
#define SYMINFO_BT_LOWRESERVE   0xff00  /* beginning of reserved entries */

/*
 * Syminfo version values.
 */
#define SYMINFO_NONE            0       /* Syminfo version */
#define SYMINFO_CURRENT         1
#define SYMINFO_NUM             2


/*
 * Public structure defined and maintained within the runtime linker
 */
#ifndef _ASM

typedef struct link_map Link_map;

struct link_map {
        unsigned long   l_addr;         /* address at which object is mapped */
        char            *l_name;        /* full name of loaded object */
#ifdef _LP64
        Elf64_Dyn       *l_ld;          /* dynamic structure of object */
#else
        Elf32_Dyn       *l_ld;          /* dynamic structure of object */
#endif
        Link_map        *l_next;        /* next link object */
        Link_map        *l_prev;        /* previous link object */
        char            *l_refname;     /* filters reference name */
};

#ifdef _SYSCALL32
```



```
typedef struct link_map32 Link_map32;

struct link_map32 {
        Elf32_Word      l_addr;
        Elf32_Addr      l_name;
        Elf32_Addr      l_ld;
        Elf32_Addr      l_next;
        Elf32_Addr      l_prev;
        Elf32_Addr      l_refname;
};
#endif

typedef enum {
        RT_CONSISTENT,
        RT_ADD,
        RT_DELETE
} r_state_e;

typedef enum {
        RD_FL_NONE = 0,         /* no flags */
        RD_FL_ODBG = (1<<0),    /* old style debugger present */
        RD_FL_DBG = (1<<1)      /* debugging enabled */
} rd_flags_e;



/*
 * Debugging events enabled inside of the runtime linker.  To
 * access these events see the librtld_db interface.
 */
typedef enum {
        RD_NONE = 0,            /* no event */
        RD_PREINIT,             /* the Initial rendezvous before .init */
        RD_POSTINIT,            /* the Second rendezvous after .init */
        RD_DLACTIVITY           /* a dlopen or dlclose has happened */
} rd_event_e;

struct r_debug {
        int             r_version;      /* debugging info version no. */
        Link_map        *r_map;         /* address of link_map */
        unsigned long   r_brk;          /* address of update routine */
        r_state_e       r_state;
        unsigned long   r_ldbase;       /* base addr of ld.so */
        Link_map        *r_ldsomap;     /* address of ld.so.1's link map */
        rd_event_e      r_rdevent;      /* debug event */
        rd_flags_e      r_flags;        /* misc flags. */
};

#ifdef _SYSCALL32
struct r_debug32 {
        Elf32_Word      r_version;      /* debugging info version no. */
        Elf32_Addr      r_map;          /* address of link_map */
        Elf32_Word      r_brk;          /* address of update routine */
        r_state_e       r_state;
        Elf32_Word      r_ldbase;       /* base addr of ld.so */
        Elf32_Addr      r_ldsomap;      /* address of ld.so.1's link map */
        rd_event_e      r_rdevent;      /* debug event */
        rd_flags_e      r_flags;        /* misc flags. */
};
#endif


#define R_DEBUG_VERSION 2               /* current r_debug version */
#endif  /* _ASM */

/*
```

```
 * Attribute/value structures used to bootstrap ELF-based dynamic linker.
 */
#ifndef _ASM
typedef struct {
        Elf32_Sword eb_tag;             /* what this one is */
        union {                         /* possible values */
                Elf32_Word eb_val;
                Elf32_Addr eb_ptr;
                Elf32_Off  eb_off;
        } eb_un;
} Elf32_Boot;

#if defined(_LP64) || defined(_LONGLONG_TYPE)
typedef struct {
        Elf64_Xword eb_tag;             /* what this one is */
        union {                         /* possible values */
                Elf64_Xword eb_val;
                Elf64_Addr eb_ptr;
                Elf64_Off eb_off;
        } eb_un;
} Elf64_Boot;
#endif  /* defined(_LP64) || defined(_LONGLONG_TYPE) */
#endif  /* _ASM */

/*
 * Attributes
 */
#define EB_NULL         0               /* (void) last entry */
#define EB_DYNAMIC      1               /* (*) dynamic structure of subject */
#define EB_LDSO_BASE    2               /* (caddr_t) base address of ld.so */
#define EB_ARGV         3               /* (caddr_t) argument vector */
#define EB_ENVP         4               /* (char **) environment strings */
#define EB_AUXV         5               /* (auxv_t *) auxiliary vector */
#define EB_DEVZERO      6               /* (int) fd for /dev/zero */
#define EB_PAGESIZE     7               /* (int) page size */
#define EB_MAX          8               /* number of "EBs" */
#define EB_MAX_SIZE32   64              /* size in bytes, _ILP32 */
#define EB_MAX_SIZE64   128             /* size in bytes, _LP64 */


#ifndef _ASM

/*
 * Concurrency communication structure for libc callbacks.
 */
extern void     _ld_libc(void *);

#pragma unknown_control_flow(_ld_libc)
#endif /* _ASM */

#ifdef  __cplusplus
}
#endif

#endif  /* _SYS_LINK_H */
```

```
**********************************************************
   15166 Wed May 27 19:49:33 2015
new/usr/src/uts/common/sys/mman.h
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
```

```
   1 /*
   2  * CDDL HEADER START
   3  *
   4  * The contents of this file are subject to the terms of the
   5  * Common Development and Distribution License (the "License").
   6  * You may not use this file except in compliance with the License.
   7  *
   8  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
   9  * or http://www.opensolaris.org/os/licensing.
  10  * See the License for the specific language governing permissions
  11  * and limitations under the License.
  12  *
  13  * When distributing Covered Code, include this CDDL HEADER in each
  14  * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  15  * If applicable, add the following below this CDDL HEADER, with the
  16  * fields enclosed by brackets "[]" replaced with your own identifying
  17  * information: Portions Copyright [yyyy] [name of copyright owner]
  18  *
  19  * CDDL HEADER END
  20  */

  22 /* Copyright 2013 OmniTI Computer Consulting, Inc. All rights reserved. */
  23 /*
  24  * Copyright 2014 Garrett D'Amore <garrett@damore.org>
  25  *
  26  * Copyright 2008 Sun Microsystems, Inc.  All rights reserved.
  27  * Use is subject to license terms.
  28  * Copyright 2015 Joyent, Inc.  All rights reserved.
  29  */

  31 /*      Copyright (c) 1983, 1984, 1985, 1986, 1987, 1988, 1989 AT&T     */
  32 /*        All Rights Reserved   */

  34 /*
  35  * University Copyright- Copyright (c) 1982, 1986, 1988
  36  * The Regents of the University of California
  37  * All Rights Reserved
  38  *
  39  * University Acknowledgment- Portions of this document are derived from
  40  * software developed by the University of California, Berkeley, and its
  41  * contributors.
  42  */

  44 #ifndef _SYS_MMAN_H
  45 #define _SYS_MMAN_H

  47 #include <sys/feature_tests.h>

  49 #ifdef  __cplusplus
  50 extern "C" {
  51 #endif

  53 #if     !defined(_ASM) && !defined(_KERNEL)
  54 #include <sys/types.h>
  55 #endif  /* !_ASM && !_KERNEL */

  57 /*
  58  * Protections are chosen from these bits, or-ed together.
```



```
  59  * Note - not all implementations literally provide all possible
  60  * combinations.  PROT_WRITE is often implemented as (PROT_READ |
  61  * PROT_WRITE) and (PROT_EXECUTE as PROT_READ | PROT_EXECUTE).
  62  * However, no implementation will permit a write to succeed
  63  * where PROT_WRITE has not been set.  Also, no implementation will
  64  * allow any access to succeed where prot is specified as PROT_NONE.
  65  */
  66 #define PROT_READ       0x1             /* pages can be read */
  67 #define PROT_WRITE      0x2             /* pages can be written */
  68 #define PROT_EXEC       0x4             /* pages can be executed */

  70 #ifdef  _KERNEL
  71 #define PROT_USER       0x8             /* pages are user accessable */
  72 #define PROT_ZFOD       (PROT_READ | PROT_WRITE | PROT_EXEC | PROT_USER)
  73 #define PROT_ALL        (PROT_READ | PROT_WRITE | PROT_EXEC | PROT_USER)
  74 #endif  /* _KERNEL */

  76 #define PROT_NONE       0x0             /* pages cannot be accessed */

  78 /* sharing types:  must choose either SHARED or PRIVATE */
  79 #define MAP_SHARED      1               /* share changes */
  80 #define MAP_PRIVATE     2               /* changes are private */
  81 #define MAP_TYPE        0xf             /* mask for share type */

  83 /* other flags to mmap (or-ed in to MAP_SHARED or MAP_PRIVATE) */
  84 #define MAP_FIXED       0x10            /* user assigns address */
  85 /* Not implemented */
  86 #define MAP_RENAME      0x20            /* rename private pages to file */
  87 #endif /* ! codereview */
  88 #define MAP_NORESERVE   0x40            /* don't reserve needed swap area */
  89 /* Note that 0x80 is _MAP_LOW32, defined below */
  90 #endif /* ! codereview */
  91 #define MAP_ANON        0x100           /* map anonymous pages directly */
  92 #define MAP_ANONYMOUS   MAP_ANON        /* (source compatibility) */
  93 #define MAP_ALIGN       0x200           /* addr specifies alignment */
  94 #define MAP_TEXT        0x400           /* map code segment */
  95 #define MAP_INITDATA    0x800           /* map data segment */

  97 #ifdef _KERNEL
  98 #define _MAP_TEXTREPL   0x1000
  99 #define _MAP_RANDOMIZE  0x2000
 100 #endif /* ! codereview */
 101 #endif /* _KERNEL */

  85 /* these flags not yet implemented */
  86 #define MAP_RENAME      0x20            /* rename private pages to file */

 103 #if     (_POSIX_C_SOURCE <= 2) && !defined(_XPG4_2)
 104 /* these flags are used by memcntl */
 105 #define PROC_TEXT       (PROT_EXEC | PROT_READ)
 106 #define PROC_DATA       (PROT_READ | PROT_WRITE | PROT_EXEC)
 107 #define SHARED          0x10
 108 #define PRIVATE         0x20
 109 #define VALID_ATTR  (PROT_READ|PROT_WRITE|PROT_EXEC|SHARED|PRIVATE)
 110 #endif  /* (_POSIX_C_SOURCE <= 2) && !defined(_XPG4_2) */

 112 #if     (_POSIX_C_SOURCE <= 2) || defined(_XPG4_2)
 113 #ifdef  _KERNEL
 114 #define PROT_EXCL       0x20
 115 #endif  /* _KERNEL */

 117 #define _MAP_LOW32      0x80    /* force mapping in lower 4G of address space */
 118 #define MAP_32BIT       _MAP_LOW32

 120 /*
 121  * For the sake of backward object compatibility, we use the _MAP_NEW flag.
```

```
 122  * This flag will be automatically or'ed in by the C library for all
 123  * new mmap calls.  Previous binaries with old mmap calls will continue
 124  * to get 0 or -1 for return values.  New mmap calls will get the mapped
 125  * address as the return value if successful and -1 on errors.  By default,
 126  * new mmap calls automatically have the kernel assign the map address
 127  * unless the MAP_FIXED flag is given.
 128  */
 129 #define _MAP_NEW        0x80000000      /* users should not need to use this */
 130 #endif  /* (_POSIX_C_SOURCE <= 2) */


 133 #if !defined(__XOPEN_OR_POSIX) || defined(__EXTENSIONS__)
 134 /* External flags for mmapobj syscall (Exclusive of MAP_* flags above) */
 135 #define MMOBJ_PADDING           0x10000
 136 #define MMOBJ_INTERPRET         0x20000

 138 #define MMOBJ_ALL_FLAGS         (MMOBJ_PADDING | MMOBJ_INTERPRET)

 140 /*
 141  * Values for mr_flags field of mmapobj_result_t below.
 142  * The bottom 16 bits are mutually exclusive and thus only one
 143  * of them can be set at a time.  Use MR_GET_TYPE below to check this value.
 144  * The top 16 bits are used for flags which are not mutually exclusive and
 145  * thus more than one of these flags can be set for a given mmapobj_result_t.
 146  *
 147  * MR_PADDING being set indicates that this memory range represents the user
 148  * requested padding.
 149  *
 150  * MR_HDR_ELF being set indicates that the ELF header of the mapped object
 151  * is mapped at mr_addr + mr_offset.
 152  *
 153  * MR_HDR_AOUT being set indicates that the AOUT (4.x) header of the mapped
 154  * object is mapped at mr_addr + mr_offset.
 155  */

 157 /*
 158  * External flags for mr_flags field below.
 159  */
 160 #define MR_PADDING      0x1
 161 #define MR_HDR_ELF      0x2
 162 #define MR_HDR_AOUT     0x3

 164 /*
 165  * Internal flags for mr_flags field below.
 166  */
 167 #ifdef  _KERNEL
 168 #define MR_RESV 0x80000000      /* overmapped /dev/null */
 169 #endif  /* _KERNEL */

 171 #define MR_TYPE_MASK 0x0000ffff
 172 #define MR_GET_TYPE(val)        ((val) & MR_TYPE_MASK)

 174 #if     !defined(_ASM)
 175 typedef struct mmapobj_result {
 176         caddr_t         mr_addr;        /* mapping address */
 177         size_t          mr_msize;       /* mapping size */
 178         size_t          mr_fsize;       /* file size */
 179         size_t          mr_offset;      /* offset into file */
 180         uint_t          mr_prot;        /* the protections provided */
 181         uint_t          mr_flags;       /* info on the mapping */
 182 } mmapobj_result_t;
```

_____***unchanged_portion_omitted_***

```
**********************************************************
    9094 Wed May 27 19:49:33 2015
new/usr/src/uts/common/sys/policy.h
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
```

```
   1 /*
   2  * CDDL HEADER START
   3  *
   4  * The contents of this file are subject to the terms of the
   5  * Common Development and Distribution License (the "License").
   6  * You may not use this file except in compliance with the License.
   7  *
   8  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
   9  * or http://www.opensolaris.org/os/licensing.
  10  * See the License for the specific language governing permissions
  11  * and limitations under the License.
  12  *
  13  * When distributing Covered Code, include this CDDL HEADER in each
  14  * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  15  * If applicable, add the following below this CDDL HEADER, with the
  16  * fields enclosed by brackets "[]" replaced with your own identifying
  17  * information: Portions Copyright [yyyy] [name of copyright owner]
  18  *
  19  * CDDL HEADER END
  20  */
  21 /*
  22  * Copyright (c) 2003, 2010, Oracle and/or its affiliates. All rights reserved.
  23  * Copyright 2013, Joyent, Inc. All rights reserved.
  24  */

  26 #ifndef _SYS_POLICY_H
  27 #define _SYS_POLICY_H

  29 #include <sys/types.h>
  30 #include <sys/cred.h>
  31 #include <sys/vnode.h>
  32 #include <sys/fs/snode.h>

  34 #ifdef __cplusplus
  35 extern "C" {
  36 #endif

  38 #ifdef _KERNEL

  40 #ifndef _IN_PORT_T
  41 #define _IN_PORT_T
  42 typedef uint16_t in_port_t;
  43 #endif

  45 /*
  46  * Policy routines; in case we check privileges in-line.
  47  *
  48  * priv_policy
  49  *              privilege debugging
  50  *              audits success & failure
  51  *              returns 0 on success, error on failure
  52  *
  53  * priv_policy_choice
  54  *              determines extend of operation
  55  *              audit on success
  56  *              returns a boolean_t indicating success (B_TRUE) or failure.
  57  *
  58  * priv_policy_only
```



```
  59  *              when auditing is in appropriate (interrupt context)
  60  *              to determine context of operation
  61  *              returns a boolean_t indicating success (B_TRUE) or failure.
  62  *
  63  */
  64 int priv_policy(const cred_t *, int, boolean_t, int, const char *);
  65 boolean_t priv_policy_only(const cred_t *, int, boolean_t);
  66 boolean_t priv_policy_choice(const cred_t *, int, boolean_t);

  68 struct kipc_perm;
  69 struct vfs;
  70 struct proc;
  71 struct priv_set;

  73 int secpolicy_acct(const cred_t *);
  74 int secpolicy_require_privs(const cred_t *, const struct priv_set *);
  75 int secpolicy_allow_setid(const cred_t *, uid_t, boolean_t);
  76 int secpolicy_audit_config(const cred_t *);
  77 int secpolicy_audit_getattr(const cred_t *, boolean_t);
  78 int secpolicy_audit_modify(const cred_t *);
  79 int secpolicy_blacklist(const cred_t *);
  80 int secpolicy_chroot(const cred_t *);
  81 int secpolicy_clock_highres(const cred_t *);
  82 int secpolicy_console(const cred_t *);
  83 int secpolicy_contract_identity(const cred_t *);
  84 int secpolicy_contract_observer(const cred_t *, struct contract *);
  85 boolean_t secpolicy_contract_observer_choice(const cred_t *);
  86 int secpolicy_contract_event(const cred_t *);
  87 boolean_t secpolicy_contract_event_choice(const cred_t *);
  88 int secpolicy_coreadm(const cred_t *);
  89 int secpolicy_cpc_cpu(const cred_t *);
  90 int secpolicy_dispadm(const cred_t *);
  91 int secpolicy_error_inject(const cred_t *);
  92 int secpolicy_excl_open(const cred_t *);
  93 int secpolicy_fs_allowed_mount(const char *);
  94 int secpolicy_fs_config(const cred_t *, const struct vfs *);
  95 int secpolicy_fs_linkdir(const cred_t *, const struct vfs *);
  96 int secpolicy_fs_minfree(const cred_t *, const struct vfs *);
  97 int secpolicy_fs_mount(cred_t *, vnode_t *, struct vfs *);
  98 int secpolicy_fs_quota(const cred_t *, const struct vfs *);
  99 int secpolicy_fs_unmount(cred_t *, struct vfs *);
 100 int secpolicy_idmap(const cred_t *);
 101 int secpolicy_ip(const cred_t *, int, boolean_t);
 102 int secpolicy_ip_config(const cred_t *, boolean_t);
 103 int secpolicy_dl_config(const cred_t *);
 104 int secpolicy_iptun_config(const cred_t *);
 105 int secpolicy_ipc_access(const cred_t *, const struct kipc_perm *, mode_t);
 106 int secpolicy_ipc_config(const cred_t *);
 107 int secpolicy_ipc_owner(const cred_t *, const struct kipc_perm *);
 108 int secpolicy_kmdb(const cred_t *);
 109 int secpolicy_lock_memory(const cred_t *);
 110 int secpolicy_modctl(const cred_t *, int);
 111 int secpolicy_net(const cred_t *, int, boolean_t);
 112 int secpolicy_net_bindmlp(const cred_t *);
 113 int secpolicy_net_config(const cred_t *, boolean_t);
 114 int secpolicy_net_icmpaccess(const cred_t *);
 115 int secpolicy_net_mac_aware(const cred_t *);
 116 int secpolicy_net_mac_implicit(const cred_t *);
 117 int secpolicy_net_observability(const cred_t *);
 118 int secpolicy_net_privaddr(const cred_t *, in_port_t, int proto);
 119 int secpolicy_net_rawaccess(const cred_t *);
 120 boolean_t secpolicy_net_reply_equal(const cred_t *);
 121 int secpolicy_newproc(const cred_t *);
 122 int secpolicy_nfs(const cred_t *);
 123 int secpolicy_pbind(const cred_t *);
 124 int secpolicy_pcfs_modify_bootpartition(const cred_t *);
```

```
125 int secpolicy_pfexec_register(const cred_t *);
126 int secpolicy_ponline(const cred_t *);
127 int secpolicy_pool(const cred_t *);
128 int secpolicy_power_mgmt(const cred_t *);
129 int secpolicy_ppp_config(const cred_t *);
130 int secpolicy_proc_access(const cred_t *);
131 int secpolicy_proc_excl_open(const cred_t *);
132 int secpolicy_proc_owner(const cred_t *, const cred_t *, int);
133 int secpolicy_proc_zone(const cred_t *);
134 int secpolicy_psecflags(const cred_t *, struct proc *, struct proc *);
135 #endif /* ! codereview */
136 int secpolicy_pset(const cred_t *);
137 int secpolicy_rctlsys(const cred_t *, boolean_t);
138 int secpolicy_resource(const cred_t *);
139 int secpolicy_resource_anon_mem(const cred_t *);
140 int secpolicy_rpcmod_open(const cred_t *);
141 int secpolicy_rsm_access(const cred_t *, uid_t, mode_t);
142 int secpolicy_raisepriority(const cred_t *);
143 int secpolicy_setpriority(const cred_t *);
144 int secpolicy_settime(const cred_t *);
145 int secpolicy_smb(const cred_t *);
146 int secpolicy_smbfs_login(const cred_t *, uid_t);
147 int secpolicy_spec_open(const cred_t *, struct vnode *, int);
148 int secpolicy_sti(const cred_t *);
149 int secpolicy_swapctl(const cred_t *);
150 int secpolicy_sys_config(const cred_t *, boolean_t);
151 int secpolicy_zone_admin(const cred_t *, boolean_t);
152 int secpolicy_zone_config(const cred_t *);
153 int secpolicy_sys_devices(const cred_t *);
154 int secpolicy_systeminfo(const cred_t *);
155 int secpolicy_tasksys(const cred_t *);
156 int secpolicy_vnode_access(const cred_t *, vnode_t *, uid_t, mode_t);
157 int secpolicy_vnode_access2(const cred_t *, vnode_t *, uid_t, mode_t, mode_t);
158 int secpolicy_vnode_any_access(const cred_t *, vnode_t *, uid_t);
159 int secpolicy_vnode_chown(const cred_t *, uid_t);
160 int secpolicy_vnode_create_gid(const cred_t *);
161 int secpolicy_vnode_owner(const cred_t *, uid_t);
162 int secpolicy_vnode_remove(const cred_t *);
163 int secpolicy_vnode_setdac(const cred_t *, uid_t);
164 int secpolicy_vnode_setid_retain(const cred_t *, boolean_t);
165 int secpolicy_vnode_setids_setgids(const cred_t *, gid_t);
166 int secpolicy_vnode_stky_modify(const cred_t *);
167 int secpolicy_vscan(const cred_t *);
168 int secpolicy_zinject(const cred_t *);
169 int secpolicy_zfs(const cred_t *);
170 int secpolicy_ucode_update(const cred_t *);
171 int secpolicy_sadopen(const cred_t *);
172 void secpolicy_setid_clear(vattr_t *, cred_t *);
173 void secpolicy_fs_mount_clearopts(cred_t *, struct vfs *);
174 int secpolicy_setid_setsticky_clear(vnode_t *, vattr_t *,
175     const vattr_t *, cred_t *);
176 int secpolicy_xvattr(xvattr_t *, uid_t, cred_t *, vtype_t);
177 int secpolicy_xvm_control(const cred_t *);

179 int secpolicy_basic_exec(const cred_t *, vnode_t *);
180 int secpolicy_basic_fork(const cred_t *);
181 int secpolicy_basic_link(const cred_t *);
182 int secpolicy_basic_file_read(const cred_t *, vnode_t *, const char *);
183 int secpolicy_basic_file_write(const cred_t *, vnode_t *, const char *);
184 int secpolicy_basic_net_access(const cred_t *);
185 int secpolicy_basic_proc(const cred_t *);
186 int secpolicy_basic_procinfo(const cred_t *, struct proc *, struct proc *);

188 int secpolicy_gart_access(const cred_t *);
189 int secpolicy_gart_map(const cred_t *);
190 /*
```



```
191  * This function to be called from xxfs_setattr().
192  * Must be called with the node's attributes read-write locked.
193  *
194  *              cred_t *                - acting credentials
195  *              struct vnode *          - vnode we're operating on
196  *              struct vattr *va        - new attributes, va_mask may be
197  *                                        changed on return from a call
198  *              struct vattr *oldva     - old attributes, need include owner
199  *                                        and mode only
200  *              int flags               - setattr flags
201  *              int iaccess(void *node, int mode, cred_t *cr)
202  *                                      - non-locking internal access function
203  *                                              mode be checked
204  *                                              w/ VREAD|VWRITE|VEXEC, not fs
205  *                                              internal mode encoding.
206  *
207  *              void *node              - internal node (inode, tmpnode) to
208  *                                        pass as arg to iaccess
209  */
210 int secpolicy_vnode_setattr(cred_t *, struct vnode *, struct vattr *,
211     const struct vattr *, int, int (void *, int, cred_t *), void *);

213 /*
214  * Test privilege. Audit success or failure, allow privilege debugging.
215  * Returns 0 for success, err for failure.
216  */
217 #define PRIV_POLICY(cred, priv, all, err, reason) \
218                 priv_policy((cred), (priv), (all), (err), (reason))

220 /*
221  * Test privilege. Audit success only, no privilege debugging.
222  * Returns 1 for success, and 0 for failure.
223  */
224 #define PRIV_POLICY_CHOICE(cred, priv, all) \
225                 priv_policy_choice((cred), (priv), (all))

227 /*
228  * Test privilege. No priv_debugging, no auditing.
229  * Returns 1 for success, and 0 for failure.
230  */

232 #define PRIV_POLICY_ONLY(cred, priv, all) \
233                 priv_policy_only((cred), (priv), (all))


236 #endif

238 #ifdef  __cplusplus
239 }
240 #endif

242 #endif  /* _SYS_POLICY_H */
```

```
**********************************************************
   29861 Wed May 27 19:49:34 2015
new/usr/src/uts/common/sys/proc.h
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright (c) 1988, 2010, Oracle and/or its affiliates. All rights reserved.
 */

/*      Copyright (c) 1984, 1986, 1987, 1988, 1989 AT&T */
/*        All Rights Reserved   */

#ifndef _SYS_PROC_H
#define _SYS_PROC_H

#include <sys/time.h>
#include <sys/thread.h>
#include <sys/cred.h>
#include <sys/user.h>
#include <sys/watchpoint.h>
#include <sys/timer.h>
#if defined(__x86)
#include <sys/tss.h>
#include <sys/segments.h>
#endif
#include <sys/utrap.h>
#include <sys/model.h>
#include <sys/refstr.h>
#include <sys/avl.h>
#include <sys/rctl.h>
#include <sys/list.h>
#include <sys/avl.h>
#include <sys/door_impl.h>
#include <sys/secflags.h>
#endif /* ! codereview */

#ifdef  __cplusplus
extern "C" {
#endif

/*
 * Profile arguments.
```



```
 */
struct prof {
        void            *pr_base;       /* buffer base */
        uintptr_t       pr_off;         /* pc offset */
        size_t          pr_size;        /* buffer size */
        uint32_t        pr_scale;       /* pc scaling */
        long            pr_samples;     /* sample count */
};

/*
 * An lwp directory entry.
 * If le_thread != NULL, this is an active lwp.
 * If le_thread == NULL, this is an unreaped zombie lwp.
 */
typedef struct lwpent {
        kthread_t       *le_thread;     /* the active lwp, NULL if zombie */
        id_t            le_lwpid;       /* its lwpid (t->t_tid) */
        uint16_t        le_waiters;     /* total number of lwp_wait()ers */
        uint16_t        le_dwaiters;    /* number that are daemons */
        clock_t         le_start;       /* start time of this lwp */
        struct vnode    *le_trace;      /* pointer to /proc lwp vnode */
} lwpent_t;

typedef struct pctxop {
        void    (*save_op)(void *);     /* function to invoke to save ctx */
        void    (*restore_op)(void *);  /* function to invoke to restore ctx */
        void    (*fork_op)(void *, void *); /* invoke to fork context */
        void    (*exit_op)(void *);     /* invoked during process exit */
        void    (*free_op)(void *, int); /* function which frees the context */
        void    *arg;                   /* argument to above functions */
        struct pctxop *next;            /* next pcontext ops */
} pctxop_t;

/*
 * Elements of the lwp directory, p->p_lwpdir[].
 *
 * We allocate lwp directory entries separately from lwp directory
 * elements because the lwp directory must be allocated as an array.
 * The number of lwps can grow quite large and we want to keep the
 * size of the kmem_alloc()d directory as small as possible.
 *
 * If ld_entry == NULL, the entry is free and is on the free list,
 * p->p_lwpfree, linked through ld_next.  If ld_entry != NULL, the
 * entry is used and ld_next is the thread-id hash link pointer.
 */
typedef struct lwpdir {
        struct lwpdir   *ld_next;       /* hash chain or free list */
        struct lwpent   *ld_entry;      /* lwp directory entry */
} lwpdir_t;

/*
 * Element of the p_tidhash thread-id (lwpid) hash table.
 */
typedef struct tidhash {
        kmutex_t        th_lock;
        lwpdir_t        *th_list;
} tidhash_t;

/*
 * Retired tidhash hash tables.
 */
typedef struct ret_tidhash {
        struct ret_tidhash      *rth_next;
        tidhash_t               *rth_tidhash;
        uint_t                  rth_tidhash_sz;
} ret_tidhash_t;
```

```
126 struct pool;
127 struct task;
128 struct zone;
129 struct brand;
130 struct corectl_path;
131 struct corectl_content;

133 /*
134  * One structure allocated per active process.  It contains all
135  * data needed about the process while the process may be swapped
136  * out.  Other per-process data (user.h) is also inside the proc structure.
137  * Lightweight-process data (lwp.h) and the kernel stack may be swapped out.
138  */
139 typedef struct  proc {
140         /*
141          * Fields requiring no explicit locking
142          */
143         struct  vnode *p_exec;          /* pointer to a.out vnode */
144         struct  as *p_as;               /* process address space pointer */
145         struct  plock *p_lockp;         /* ptr to proc struct's mutex lock */
146         kmutex_t p_crlock;              /* lock for p_cred */
147         struct  cred    *p_cred;        /* process credentials */
148         /*
149          * Fields protected by pidlock
150          */
151         int     p_swapcnt;              /* number of swapped out lwps */
152         char    p_stat;                 /* status of process */
153         char    p_wcode;                /* current wait code */
154         ushort_t p_pidflag;             /* flags protected only by pidlock */
155         int     p_wdata;                /* current wait return value */
156         pid_t   p_ppid;                 /* process id of parent */
157         struct  proc    *p_link;        /* forward link */
158         struct  proc    *p_parent;      /* ptr to parent process */
159         struct  proc    *p_child;       /* ptr to first child process */
160         struct  proc    *p_sibling;     /* ptr to next sibling proc on chain */
161         struct  proc    *p_psibling;    /* ptr to prev sibling proc on chain */
162         struct  proc    *p_sibling_ns;  /* prt to siblings with new state */
163         struct  proc    *p_child_ns;    /* prt to children with new state */
164         struct  proc    *p_next;        /* active chain link next */
165         struct  proc    *p_prev;        /* active chain link prev */
166         struct  proc    *p_nextofkin;   /* gets accounting info at exit */
167         struct  proc    *p_orphan;
168         struct  proc    *p_nextorph;
169         struct  proc    *p_pglink;      /* process group hash chain link next */
170         struct  proc    *p_ppglink;     /* process group hash chain link prev */
171         struct  sess    *p_sessp;       /* session information */
172         struct  pid     *p_pidp;        /* process ID info */
173         struct  pid     *p_pgidp;       /* process group ID info */
174         /*
175          * Fields protected by p_lock
176          */
177         kcondvar_t p_cv;                /* proc struct's condition variable */
178         kcondvar_t p_flag_cv;
179         kcondvar_t p_lwpexit;           /* waiting for some lwp to exit */
180         kcondvar_t p_holdlwps;          /* process is waiting for its lwps */
181                                         /* to to be held.  */
182         uint_t  p_proc_flag;            /* /proc-related flags */
183         uint_t  p_flag;                 /* protected while set. */
184                                         /* flags defined below */
185         clock_t p_utime;                /* user time, this process */
186         clock_t p_stime;                /* system time, this process */
187         clock_t p_cutime;               /* sum of children's user time */
188         clock_t p_cstime;               /* sum of children's system time */
189         avl_tree_t *p_segacct;          /* System V shared segment list */
190         avl_tree_t *p_semacct;          /* System V semaphore undo list */
```



```
191         caddr_t p_bssbase;              /* base addr of last bss below heap */
192         caddr_t p_brkbase;              /* base addr of heap */
193         size_t  p_brksize;              /* heap size in bytes */
194         uint_t  p_brkpageszc;           /* preferred heap max page size code */
195         /*
196          * Per process signal stuff.
197          */
198         k_sigset_t p_sig;               /* signals pending to this process */
199         k_sigset_t p_extsig;            /* signals sent from another contract */
200         k_sigset_t p_ignore;            /* ignore when generated */
201         k_sigset_t p_siginfo;           /* gets signal info with signal */
202         struct sigqueue *p_sigqueue;    /* queued siginfo structures */
203         struct sigqhdr *p_sigqhdr;      /* hdr to sigqueue structure pool */
204         struct sigqhdr *p_signhdr;      /* hdr to signotify structure pool */
205         uchar_t p_stopsig;              /* jobcontrol stop signal */

207         /*
208          * Special per-process flag when set will fix misaligned memory
209          * references.
210          */
211         char    p_fixalignment;

213         /*
214          * Per process lwp and kernel thread stuff
215          */
216         id_t    p_lwpid;                /* most recently allocated lwpid */
217         int     p_lwpcnt;               /* number of lwps in this process */
218         int     p_lwprcnt;              /* number of not stopped lwps */
219         int     p_lwpdaemon;            /* number of TP_DAEMON lwps */
220         int     p_lwpwait;              /* number of lwps in lwp_wait() */
221         int     p_lwpdwait;             /* number of daemons in lwp_wait() */
222         int     p_zombcnt;              /* number of zombie lwps */
223         kthread_t *p_tlist;             /* circular list of threads */
224         lwpdir_t *p_lwpdir;             /* thread (lwp) directory */
225         lwpdir_t *p_lwpfree;            /* p_lwpdir free list */
226         tidhash_t *p_tidhash;           /* tid (lwpid) lookup hash table */
227         uint_t  p_lwpdir_sz;            /* number of p_lwpdir[] entries */
228         uint_t  p_tidhash_sz;           /* number of p_tidhash[] entries */
229         ret_tidhash_t *p_ret_tidhash;   /* retired tidhash hash tables */
230         uint64_t p_lgrpset;             /* unprotected hint of set of lgrps */
231                                         /* on which process has threads */
232         volatile lgrp_id_t  p_t1_lgrpid; /* main's thread lgroup id */
233         volatile lgrp_id_t  p_tr_lgrpid; /* text replica's lgroup id */
234 #if defined(_LP64)
235         uintptr_t  p_lgrpres2;          /* reserved for lgrp migration */
236 #endif
237         /*
238          * /proc (process filesystem) debugger interface stuff.
239          */
240         k_sigset_t p_sigmask;           /* mask of traced signals (/proc) */
241         k_fltset_t p_fltmask;           /* mask of traced faults (/proc) */
242         struct vnode *p_trace;          /* pointer to primary /proc vnode */
243         struct vnode *p_plist;          /* list of /proc vnodes for process */
244         kthread_t *p_agenttp;           /* thread ptr for /proc agent lwp */
245         avl_tree_t p_warea;             /* list of watched areas */
246         avl_tree_t p_wpage;             /* remembered watched pages (vfork) */
247         watched_page_t *p_wprot;        /* pages that need to have prot set */
248         int     p_mapcnt;               /* number of active pr_mappage()s */
249         kmutex_t p_maplock;             /* lock for pr_mappage() */
250         struct  proc  *p_rlink;         /* linked list for server */
251         kcondvar_t p_srwchan_cv;
252         size_t  p_stksize;              /* process stack size in bytes */
253         uint_t  p_stkpageszc;           /* preferred stack max page size code */

255         /*
256          * Microstate accounting, resource usage, and real-time profiling
```

```
257          */
258         hrtime_t p_mstart;              /* hi-res process start time */
259         hrtime_t p_mterm;               /* hi-res process termination time */
260         hrtime_t p_mlreal;              /* elapsed time sum over defunct lwps */
261         hrtime_t p_acct[NMSTATES];      /* microstate sum over defunct lwps */
262         hrtime_t p_cacct[NMSTATES];     /* microstate sum over child procs */
263         struct lrusage p_ru;            /* lrusage sum over defunct lwps */
264         struct lrusage p_cru;           /* lrusage sum over child procs */
265         struct itimerval p_rprof_timer; /* ITIMER_REALPROF interval timer */
266         uintptr_t p_rprof_cyclic;       /* ITIMER_REALPROF cyclic */
267         uint_t  p_defunct;              /* number of defunct lwps */
268         /*
269          * profiling. A lock is used in the event of multiple lwp's
270          * using the same profiling base/size.
271          */
272         kmutex_t p_pflock;              /* protects user profile arguments */
273         struct prof p_prof;             /* profile arguments */

275         /*
276          * Doors.
277          */
278         door_pool_t             p_server_threads; /* common thread pool */
279         struct door_node        *p_door_list;   /* active doors */
280         struct door_node        *p_unref_list;
281         kcondvar_t              p_unref_cv;
282         char                    p_unref_thread; /* unref thread created */

284         /*
285          * Kernel probes
286          */
287         uchar_t                 p_tnf_flags;

289         /*
290          * Solaris Audit
291          */
292         struct p_audit_data     *p_audit_data; /* per process audit structure */

294         pctxop_t        *p_pctx;

296 #if defined(__x86)
297         /*
298          * LDT support.
299          */
300         kmutex_t        p_ldtlock;      /* protects the following fields */
301         user_desc_t     *p_ldt;         /* Pointer to private LDT */
302         system_desc_t   p_ldt_desc;     /* segment descriptor for private LDT */
303         ushort_t        p_ldtlimit;     /* highest selector used */
304 #endif
305         size_t p_swrss;                 /* resident set size before last swap */
306         struct aio      *p_aio;         /* pointer to async I/O struct */
307         struct itimer   **p_itimer;     /* interval timers */
308         timeout_id_t    p_alarmid;      /* alarm's timeout id */
309         caddr_t         p_usrstack;     /* top of the process stack */
310         uint_t          p_stkprot;      /* stack memory protection */
311         uint_t          p_datprot;      /* data memory protection */
312         model_t         p_model;        /* data model determined at exec time */
313         struct lwpchan_data *p_lcp;     /* lwpchan cache */
314         kmutex_t        p_lcp_lock;     /* protects assignments to p_lcp */
315         utrap_handler_t *p_utraps;      /* pointer to user trap handlers */
316         struct corectl_path     *p_corefile;    /* pattern for core file */
317         struct task     *p_task;        /* our containing task */
318         struct proc     *p_taskprev;    /* ptr to previous process in task */
319         struct proc     *p_tasknext;    /* ptr to next process in task */
320         kmutex_t        p_sc_lock;      /* protects p_pagep */
321         struct sc_page_ctl *p_pagep;    /* list of process's shared pages */
322         struct rctl_set *p_rctls;       /* resource controls for this process */
```



```
323         rlim64_t        p_stk_ctl;      /* currently enforced stack size */
324         rlim64_t        p_fsz_ctl;      /* currently enforced file size */
325         rlim64_t        p_vmem_ctl;     /* currently enforced addr-space size */
326         rlim64_t        p_fno_ctl;      /* currently enforced file-desc limit */
327         pid_t           p_ancpid;       /* ancestor pid, used by exacct */
328         struct itimerval p_realitimer;  /* real interval timer */
329         timeout_id_t    p_itimerid;     /* real interval timer's timeout id */
330         struct corectl_content *p_content;      /* content of core file */

332         avl_tree_t      p_ct_held;      /* held contracts */
333         struct ct_equeue **p_ct_equeue; /* process-type event queues */

335         struct cont_process *p_ct_process; /* process contract */
336         list_node_t     p_ct_member;    /* process contract membership */
337         sigqueue_t      *p_killsqp;     /* sigqueue pointer for SIGKILL */

339         int             p_dtrace_probes; /* are there probes for this proc? */
340         uint64_t        p_dtrace_count; /* number of DTrace tracepoints */
341                                         /* (protected by P_PR_LOCK) */
342         void            *p_dtrace_helpers; /* DTrace helpers, if any */
343         struct pool     *p_pool;        /* pointer to containing pool */
344         kcondvar_t      p_poolcv;       /* synchronization with pools */
345         uint_t          p_poolcnt;      /* # threads inside pool barrier */
346         uint_t          p_poolflag;     /* pool-related flags (see below) */
347         uintptr_t       p_portcnt;      /* event ports counter */
348         struct zone     *p_zone;        /* zone in which process lives */
349         struct vnode    *p_execdir;     /* directory that p_exec came from */
350         struct brand    *p_brand;       /* process's brand  */
351         void            *p_brand_data;  /* per-process brand state */
352         psecflags_t     p_secflags;     /* per-process security flags */
353 #endif /* ! codereview */

355         /* additional lock to protect p_sessp (but not its contents) */
356         kmutex_t p_splock;
357         rctl_qty_t      p_locked_mem;   /* locked memory charged to proc */
358                                         /* protected by p_lock */
359         rctl_qty_t      p_crypto_mem;   /* /dev/crypto memory charged to proc */
360                                         /* protected by p_lock */
361         clock_t p_ttime;                /* buffered task time */

363         /*
364          * The user structure
365          */
366         struct user p_user;             /* (see sys/user.h) */
367 } proc_t;

369 #define PROC_T                          /* headers relying on proc_t are OK */

371 #ifdef _KERNEL

373 /* active process chain */

375 extern proc_t *practive;

377 /* Well known processes */

379 extern proc_t *proc_sched;              /* memory scheduler */
380 extern proc_t *proc_init;               /* init */
381 extern proc_t *proc_pageout;            /* pageout daemon */
382 extern proc_t *proc_fsflush;            /* filesystem sync-er */

384 #endif /* _KERNEL */

386 /*
387  * Stuff to keep track of the number of processes each uid has.
388  * It is tracked on a per-zone basis; that is, if users in different
```

```
 * zones have the same uid, they are tracked separately.
 *
 * A structure is allocated when a new <uid,zoneid> pair shows up
 * There is a hash to find each structure.
 */
struct  upcount {
        struct  upcount *up_next;
        uid_t           up_uid;
        zoneid_t        up_zoneid;
        uint_t          up_count;
};

/* process ID info */

struct pid {
        unsigned int pid_prinactive :1;
        unsigned int pid_pgorphaned :1;
        unsigned int pid_padding :6;    /* used to be pid_ref, now an int */
        unsigned int pid_prslot :24;
        pid_t pid_id;
        struct proc *pid_pglink;
        struct proc *pid_pgtail;
        struct pid *pid_link;
        uint_t pid_ref;
};

#define p_pgrp p_pgidp->pid_id
#define p_pid  p_pidp->pid_id
#define p_slot p_pidp->pid_prslot
#define p_detached p_pgidp->pid_pgorphaned

#define PID_HOLD(pidp)  ASSERT(MUTEX_HELD(&pidlock)); \
                        ++(pidp)->pid_ref;
#define PID_RELE(pidp)  ASSERT(MUTEX_HELD(&pidlock)); \
                        (pidp)->pid_ref > 1 ? \
                                --(pidp)->pid_ref : pid_rele(pidp);

/*
 * Structure containing persistent process lock.  The structure and
 * macro allow "mutex_enter(&p->p_lock)" to continue working.
 */
struct plock {
        kmutex_t pl_lock;
};
#define p_lock  p_lockp->pl_lock

#ifdef _KERNEL
extern proc_t p0;               /* process 0 */
extern struct plock p0lock;     /* p0's plock */
extern struct pid pid0;         /* p0's pid */

/* pid_allocate() flags */
#define PID_ALLOC_PROC  0x0001  /* assign a /proc slot as well */

#endif /* _KERNEL */

/* stat codes */

#define SSLEEP  1               /* awaiting an event */
#define SRUN    2               /* runnable */
#define SZOMB   3               /* process terminated but not waited for */
#define SSTOP   4               /* process stopped by debugger */
#define SIDL    5               /* intermediate state in process creation */
#define SONPROC 6               /* process is being run on a processor */
#define SWAIT   7               /* process is waiting to become runnable */
```



```
/* p_pidflag codes */
#define CLDPEND         0x0001  /* have yet to post a SIGCHLD to the parent */
#define CLDCONT         0x0002  /* child has notified parent of CLD_CONTINUED */
#define CLDNOSIGCHLD    0x0004  /* do not post SIGCHLD when child terminates */
#define CLDWAITPID      0x0008  /* only waitid(P_PID, pid) can reap the child */

/* p_proc_flag codes -- these flags are mostly private to /proc */
#define P_PR_TRACE      0x0001  /* signal, fault or syscall tracing via /proc */
#define P_PR_PTRACE     0x0002  /* ptrace() compatibility mode */
#define P_PR_FORK       0x0004  /* child inherits tracing flags */
#define P_PR_LOCK       0x0008  /* process locked by /proc */
#define P_PR_ASYNC      0x0010  /* asynchronous stopping via /proc */
#define P_PR_EXEC       0x0020  /* process is in exec() */
#define P_PR_BPTADJ     0x0040  /* adjust pc on breakpoint trap */
#define P_PR_RUNLCL     0x0080  /* set process running on last /proc close */
#define P_PR_KILLCL     0x0100  /* kill process on last /proc close */

/*
 * p_flag codes
 *
 * note that two of these flags, SMSACCT and SSYS, are exported to /proc's
 * psinfo_t.p_flag field.  Historically, all were, but since they are
 * implementation dependant, we only export the ones people have come to
 * rely upon.  Hence, the bit positions of SSYS and SMSACCT should not be
 * altered.
 */
#define SSYS       0x00000001   /* system (resident) process */
#define SEXITING   0x00000002   /* process is exiting */
#define SITBUSY    0x00000004   /* setitimer(ITIMER_REAL) in progress */
#define SFORKING   0x00000008   /* tells called functions that we're forking */
#define SWATCHOK   0x00000010   /* proc in acceptable state for watchpoints */
#define SKILLED    0x00000100   /* SIGKILL has been posted to the process */
#define SSCONT     0x00000200   /* SIGCONT has been posted to the process */
#define SZONETOP   0x00000400   /* process has no valid PPID in its zone */
#define SEXTKILLED 0x00000800   /* SKILLED is from another contract */
#define SUGID      0x00002000   /* process was result of set[ug]id exec */
#define SEXECED    0x00004000   /* this process has execed */
#define SJCTL      0x00010000   /* SIGCHLD sent when children stop/continue */
#define SNOWAIT    0x00020000   /* children never become zombies */
#define SVFORK     0x00040000   /* child of vfork that has not yet exec'd */
#define SVFWAIT    0x00080000   /* parent of vfork waiting for child to exec */
#define SEXITLWPS  0x00100000   /* have lwps exit within the process */
#define SHOLDFORK  0x00200000   /* hold lwps where they're cloneable */
#define SHOLDFORK1 0x00800000   /* hold lwps in place (not cloning) */
#define SCOREDUMP  0x01000000   /* process is dumping core */
#define SMSACCT    0x02000000   /* process is keeping micro-state accounting */
#define SLWPWRAP   0x04000000   /* process has wrapped its lwp ids */
#define SAUTOLPG   0x08000000   /* kernel controls page sizes */
#define SNOCD      0x10000000   /* new creds from VSxID, do not coredump */
#define SHOLDWATCH 0x20000000   /* hold lwps for watchpoint operation */
#define SMSFORK    0x40000000   /* child inherits micro-state accounting */
#define SDOCORE    0x80000000   /* process will attempt to dump core */

/*
 * p_poolflag codes
 *
 * These flags are used to synchronize with the pool subsystem to allow
 * re-binding of processes to new pools.
 */
#define PBWAIT          0x0001  /* process should wait outside fork/exec/exit */
#define PEXITED         0x0002  /* process exited and about to become zombie */

/* Macro to convert proc pointer to a user block pointer */
#define PTOU(p)         (&(p)->p_user)

#define tracing(p, sig) (sigismember(&(p)->p_sigmask, sig))
```

```
/* Macro to reduce unnecessary calls to issig() */

#define ISSIG(t, why)   ISSIG_FAST(t, ttolwp(t), ttoproc(t), why)

/*
 * Fast version of ISSIG.
 *      1. uses register pointers to lwp and proc instead of reloading them.
 *      2. uses bit-wise OR of tests, since the usual case is that none of them
 *         are true, this saves orcc's and branches.
 *      3. load the signal flags instead of using sigisempty() macro which does
 *         a branch to convert to boolean.
 */
#define ISSIG_FAST(t, lwp, p, why)              \
        (ISSIG_PENDING(t, lwp, p) && issig(why))

#define ISSIG_PENDING(t, lwp, p)                \
        ((lwp)->lwp_cursig |                    \
            sigcheck((p), (t)) |                \
            (p)->p_stopsig |                    \
            (t)->t_dtrace_stop |                \
            (t)->t_dtrace_sig |                 \
            ((t)->t_proc_flag & (TP_PRSTOP|TP_HOLDLWP|TP_CHKPT|TP_PAUSE)) | \
            ((p)->p_flag & (SEXITLWPS|SKILLED|SHOLDFORK1|SHOLDWATCH)))

#define ISSTOP(sig)      (u.u_signal[sig-1] == SIG_DFL && \
                                sigismember(&stopdefault, sig))

#define ISHOLD(p)       ((p)->p_flag & SHOLDFORK)

#define MUSTRETURN(p, t)        (ISHOLD(p) | (t)->t_activefd.a_stale)

/*
 * Determine if there are any watchpoints active in the process.
 */
#define pr_watch_active(p)      (avl_numnodes(&(p)->p_warea) != 0)

/* Reasons for calling issig() */

#define FORREAL         0       /* Usual side-effects */
#define JUSTLOOKING     1       /* Don't stop the process */

/* 'what' values for stop(PR_SUSPENDED, what) */
#define SUSPEND_NORMAL  0
#define SUSPEND_PAUSE   1

/* pseudo-flag to lwp_create() */
#define NOCLASS (-1)

/* unused scheduling class ID */
#define CLASS_UNUSED    (-2)

/* LWP stats updated via lwp_stats_update() */
typedef enum {
        LWP_STAT_INBLK,
        LWP_STAT_OUBLK,
        LWP_STAT_MSGRCV,
        LWP_STAT_MSGSND
} lwp_stat_id_t;

typedef struct prkillinfo {
        int32_t prk_error;              /* errno */
        int32_t prk_pad;                /* pad */
        siginfo_t prk_info;             /* siginfo of killing signal */
} prkillinfo_t;
```



```
#ifdef _KERNEL

/* user profiling functions */

extern void profil_tick(uintptr_t);

/* process management functions */

extern int newproc(void (*)(), caddr_t, id_t, int, struct contract **, pid_t);
extern void vfwait(pid_t);
extern void proc_detach(proc_t *);
extern void freeproc(proc_t *);
extern void setrun(kthread_t *);
extern void setrun_locked(kthread_t *);
extern void exit(int, int);
extern int proc_exit(int, int);
extern void proc_is_exiting(proc_t *);
extern void relvm(void);
extern void add_ns(proc_t *, proc_t *);
extern void delete_ns(proc_t *, proc_t *);
extern void upcount_inc(uid_t, zoneid_t);
extern void upcount_dec(uid_t, zoneid_t);
extern int upcount_get(uid_t, zoneid_t);
#if defined(__x86)
extern selector_t setup_thrptr(proc_t *, uintptr_t);
extern void deferred_singlestep_trap(caddr_t);
#endif

extern void sigcld(proc_t *, sigqueue_t *);
extern void sigcld_delete(k_siginfo_t *);
extern void sigcld_repost(void);
extern int fsig(k_sigset_t *, kthread_t *);
extern void psig(void);
extern void stop(int, int);
extern int stop_on_fault(uint_t, k_siginfo_t *);
extern int issig(int);
extern int jobstopped(proc_t *);
extern void psignal(proc_t *, int);
extern void tsignal(kthread_t *, int);
extern void sigtoproc(proc_t *, kthread_t *, int);
extern void trapsig(k_siginfo_t *, int);
extern void realsigprof(int, int, int);
extern int eat_signal(kthread_t *, int);
extern int signal_is_blocked(kthread_t *, int);
extern int sigcheck(proc_t *, kthread_t *);
extern void sigdefault(proc_t *);

extern void pid_setmin(void);
extern pid_t pid_allocate(proc_t *, pid_t, int);
extern int pid_rele(struct pid *);
extern void pid_exit(proc_t *, struct task *);
extern void proc_entry_free(struct pid *);
extern proc_t *prfind(pid_t);
extern proc_t *prfind_zone(pid_t, zoneid_t);
extern proc_t *pgfind(pid_t);
extern proc_t *pgfind_zone(pid_t, zoneid_t);
extern proc_t *sprlock(pid_t);
extern proc_t *sprlock_zone(pid_t, zoneid_t);
extern int sprtrylock_proc(proc_t *);
extern void sprwaitlock_proc(proc_t *);
extern void sprlock_proc(proc_t *);
extern void sprunlock(proc_t *);
extern void pid_init(void);
extern proc_t *pid_entry(int);
extern int pid_slot(proc_t *);
extern void signal(pid_t, int);
```

```
653 extern void prsignal(struct pid *, int);
654 extern int uread(proc_t *, void *, size_t, uintptr_t);
655 extern int uwrite(proc_t *, void *, size_t, uintptr_t);

657 extern void pgsignal(struct pid *, int);
658 extern void pgjoin(proc_t *, struct pid *);
659 extern void pgcreate(proc_t *);
660 extern void pgexit(proc_t *);
661 extern void pgdetach(proc_t *);
662 extern int pgmembers(pid_t);

664 extern  void    init_mstate(kthread_t *, int);
665 extern  int     new_mstate(kthread_t *, int);
666 extern  void    restore_mstate(kthread_t *);
667 extern  void    term_mstate(kthread_t *);
668 extern  void    estimate_msacct(kthread_t *, hrtime_t);
669 extern  void    disable_msacct(proc_t *);
670 extern  hrtime_t mstate_aggr_state(proc_t *, int);
671 extern  hrtime_t mstate_thread_onproc_time(kthread_t *);
672 extern  void    mstate_systhread_times(kthread_t *, hrtime_t *, hrtime_t *);
673 extern  void    syscall_mstate(int, int);

675 extern  uint_t  cpu_update_pct(kthread_t *, hrtime_t);

677 extern void     set_proc_pre_sys(proc_t *p);
678 extern void     set_proc_post_sys(proc_t *p);
679 extern void     set_proc_sys(proc_t *p);
680 extern void     set_proc_ast(proc_t *p);
681 extern void     set_all_proc_sys(void);
682 extern void     set_all_zone_usr_proc_sys(zoneid_t);

684 /* thread function prototypes */

686 extern  kthread_t       *thread_create(
687         caddr_t         stk,
688         size_t          stksize,
689         void            (*proc)(),
690         void            *arg,
691         size_t          len,
692         proc_t          *pp,
693         int             state,
694         pri_t           pri);
695 extern  void    thread_exit(void) __NORETURN;
696 extern  void    thread_free(kthread_t *);
697 extern  void    thread_rele(kthread_t *);
698 extern  void    thread_join(kt_did_t);
699 extern  int     reaper(void);
700 extern  void    installctx(kthread_t *, void *, void (*)(), void (*)(),
701     void (*)(), void (*)(), void (*)());
702 extern  int     removectx(kthread_t *, void *, void (*)(), void (*)(),
703     void (*)(), void (*)(), void (*)(), void (*)());
704 extern  void    savectx(kthread_t *);
705 extern  void    restorectx(kthread_t *);
706 extern  void    forkctx(kthread_t *, kthread_t *);
707 extern  void    lwp_createctx(kthread_t *, kthread_t *);
708 extern  void    exitctx(kthread_t *);
709 extern  void    freectx(kthread_t *, int);
710 extern  void    installpctx(proc_t *, void *, void (*)(), void (*)(),
711     void (*)(), void (*)(), void (*)());
712 extern  int     removepctx(proc_t *, void *, void (*)(), void (*)(),
713     void (*)(), void (*)(), void (*)());
714 extern  void    savepctx(proc_t *);
715 extern  void    restorepctx(proc_t *);
716 extern  void    forkpctx(proc_t *, proc_t *);
717 extern  void    exitpctx(proc_t *);
718 extern  void    freepctx(proc_t *, int);
```



```
719 extern  kthread_t *thread_unpin(void);
720 extern  void    thread_init(void);
721 extern  void    thread_load(kthread_t *, void (*)(), caddr_t, size_t);

723 extern  void    tsd_create(uint_t *, void (*)(void *));
724 extern  void    tsd_destroy(uint_t *);
725 extern  void    *tsd_getcreate(uint_t *, void (*)(void *), void *(*)(void));
726 extern  void    *tsd_get(uint_t);
727 extern  int     tsd_set(uint_t, void *);
728 extern  void    tsd_exit(void);
729 extern  void    *tsd_agent_get(kthread_t *, uint_t);
730 extern  int     tsd_agent_set(kthread_t *, uint_t, void *);

732 /* lwp function prototypes */

734 extern kthread_t *lwp_kernel_create(proc_t *, void (*)(), void *, int, pri_t);
735 extern  klwp_t          *lwp_create(
736         void            (*proc)(),
737         caddr_t         arg,
738         size_t          len,
739         proc_t          *p,
740         int             state,
741         int             pri,
742         const k_sigset_t *smask,
743         int             cid,
744         id_t            lwpid);
745 extern  kthread_t *idtot(proc_t *, id_t);
746 extern  void    lwp_hash_in(proc_t *, lwpent_t *, tidhash_t *, uint_t, int);
747 extern  void    lwp_hash_out(proc_t *, id_t);
748 extern  lwpdir_t *lwp_hash_lookup(proc_t *, id_t);
749 extern  lwpdir_t *lwp_hash_lookup_and_lock(proc_t *, id_t, kmutex_t **);
750 extern  void    lwp_create_done(kthread_t *);
751 extern  void    lwp_exit(void);
752 extern  void    lwp_pcb_exit(void);
753 extern  void    lwp_cleanup(void);
754 extern  int     lwp_suspend(kthread_t *);
755 extern  void    lwp_continue(kthread_t *);
756 extern  void    holdlwp(void);
757 extern  void    stoplwp(void);
758 extern  int     holdlwps(int);
759 extern  int     holdwatch(void);
760 extern  void    pokelwps(proc_t *);
761 extern  void    continuelwps(proc_t *);
762 extern  int     exitlwps(int);
763 extern  void    lwp_ctmpl_copy(klwp_t *, klwp_t *);
764 extern  void    lwp_ctmpl_clear(klwp_t *);
765 extern  klwp_t  *forklwp(klwp_t *, proc_t *, id_t);
766 extern  void    lwp_load(klwp_t *, gregset_t, uintptr_t);
767 extern  void    lwp_setrval(klwp_t *, int, int);
768 extern  void    lwp_forkregs(klwp_t *, klwp_t *);
769 extern  void    lwp_freeregs(klwp_t *, int);
770 extern  caddr_t lwp_stk_init(klwp_t *, caddr_t);
771 extern  void    lwp_stk_cache_init(void);
772 extern  void    lwp_stk_fini(klwp_t *);
773 extern  void    lwp_installctx(klwp_t *);
774 extern  void    lwp_rtt(void);
775 extern  void    lwp_rtt_initial(void);
776 extern  int     lwp_setprivate(klwp_t *, int, uintptr_t);
777 extern  void    lwp_stat_update(lwp_stat_id_t, long);
778 extern  void    lwp_attach_brand_hdlrs(klwp_t *);
779 extern  void    lwp_detach_brand_hdlrs(klwp_t *);

781 #if defined(__sparcv9)
782 extern  void    lwp_mmodel_newlwp(void);
783 extern  void    lwp_mmodel_shared_as(caddr_t, size_t);
784 #define LWP_MMODEL_NEWLWP()             lwp_mmodel_newlwp()
```

```
785 #define LWP_MMODEL_SHARED_AS(addr, sz)  lwp_mmodel_shared_as((addr), (sz))
786 #else
787 #define LWP_MMODEL_NEWLWP()
788 #define LWP_MMODEL_SHARED_AS(addr, sz)
789 #endif

791 /* Security flag manipulation */
792 extern boolean_t secflag_enabled(proc_t *, uint_t);
793 extern void secflag_set(proc_t *, uint_t);
794 extern void secflag_enable(proc_t *, uint_t);
795 extern void secflag_disable(proc_t *, uint_t);
796 extern void secflag_promote(proc_t *);

798 #endif /* ! codereview */
799 /*
800  * Signal queue function prototypes. Must be here due to header ordering
801  * dependencies.
802  */
803 extern void sigqfree(proc_t *);
804 extern void siginfofree(sigqueue_t *);
805 extern void sigdeq(proc_t *, kthread_t *, int, sigqueue_t **);
806 extern void sigdelq(proc_t *, kthread_t *, int);
807 extern void sigaddq(proc_t *, kthread_t *, k_siginfo_t *, int);
808 extern void sigaddqa(proc_t *, kthread_t *, sigqueue_t *);
809 extern void sigqsend(int, proc_t *, kthread_t *, sigqueue_t *);
810 extern void sigdupq(proc_t *, proc_t *);
811 extern int sigwillqueue(int, int);
812 extern sigqhdr_t *sigqhdralloc(size_t, uint_t);
813 extern sigqueue_t *sigqalloc(sigqhdr_t *);
814 extern void sigqhdrfree(sigqhdr_t *);
815 extern sigqueue_t *sigappend(k_sigset_t *, sigqueue_t *,
816         k_sigset_t *, sigqueue_t *);
817 extern sigqueue_t *sigprepend(k_sigset_t *, sigqueue_t *,
818         k_sigset_t *, sigqueue_t *);
819 extern void winfo(proc_t *, k_siginfo_t *, int);
820 extern int wstat(int, int);
821 extern int sendsig(int, k_siginfo_t *, void (*)());
822 #if defined(_SYSCALL32_IMPL)
823 extern int sendsig32(int, k_siginfo_t *, void (*)());
824 #endif

826 #endif  /* _KERNEL */

828 #ifdef  __cplusplus
829 }
830 #endif

832 #endif  /* _SYS_PROC_H */
```

```
**********************************************************
   35056 Wed May 27 19:49:34 2015
new/usr/src/uts/common/sys/procfs.h
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright 2006 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */
/*
 * Copyright 2012 DEY Storage Systems, Inc.  All rights reserved.
 */

#ifndef _SYS_PROCFS_H
#define _SYS_PROCFS_H

#ifdef  __cplusplus
extern "C" {
#endif

/*
 * This definition is temporary.  Structured proc is the preferred API,
 * and the older ioctl-based interface will be removed in a future version
 * of Solaris.  Until then, by default, including <sys/procfs.h> will
 * provide the older ioctl-based /proc definitions.  To get the structured
 * /proc definitions, either include <procfs.h> or define _STRUCTURED_PROC
 * to be 1 before including <sys/procfs.h>.
 */
#ifndef _STRUCTURED_PROC
#define _STRUCTURED_PROC        0
#endif

#if !defined(_KERNEL) && _STRUCTURED_PROC == 0

#include <sys/old_procfs.h>

#else   /* !defined(_KERNEL) && _STRUCTURED_PROC == 0 */

#include <sys/feature_tests.h>
#include <sys/types.h>
#include <sys/time_impl.h>
#include <sys/signal.h>
```



```
#include <sys/siginfo.h>
#include <sys/fault.h>
#include <sys/syscall.h>
#include <sys/pset.h>
#include <sys/procfs_isa.h>
#include <sys/priv.h>
#include <sys/stat.h>
#include <sys/param.h>
#include <sys/secflags.h>
#endif /* ! codereview */

/*
 * System call interfaces for /proc.
 */

/*
 * Control codes (long values) for messages written to ctl and lwpctl files.
 */
#define PCNULL   0L     /* null request, advance to next message */
#define PCSTOP   1L     /* direct process or lwp to stop and wait for stop */
#define PCDSTOP  2L     /* direct process or lwp to stop */
#define PCWSTOP  3L     /* wait for process or lwp to stop, no timeout */
#define PCTWSTOP 4L     /* wait for stop, with long millisecond timeout arg */
#define PCRUN    5L     /* make process/lwp runnable, w/ long flags argument */
#define PCCSIG   6L     /* clear current signal from lwp */
#define PCCFAULT 7L     /* clear current fault from lwp */
#define PCSSIG   8L     /* set current signal from siginfo_t argument */
#define PCKILL   9L     /* post a signal to process/lwp, long argument */
#define PCUNKILL 10L    /* delete a pending signal from process/lwp, long arg */
#define PCSHOLD  11L    /* set lwp signal mask from sigset_t argument */
#define PCSTRACE 12L    /* set traced signal set from sigset_t argument */
#define PCSFAULT 13L    /* set traced fault set from fltset_t argument */
#define PCSENTRY 14L    /* set traced syscall entry set from sysset_t arg */
#define PCSEXIT  15L    /* set traced syscall exit set from sysset_t arg */
#define PCSET    16L    /* set modes from long argument */
#define PCUNSET  17L    /* unset modes from long argument */
#define PCSREG   18L    /* set lwp general registers from prgregset_t arg */
#define PCSFPREG 19L    /* set lwp floating-point registers from prfpregset_t */
#define PCSXREG  20L    /* set lwp extra registers from prxregset_t arg */
#define PCNICE   21L    /* set nice priority from long argument */
#define PCSVADDR 22L    /* set %pc virtual address from long argument */
#define PCWATCH  23L    /* set/unset watched memory area from prwatch_t arg */
#define PCAGENT  24L    /* create agent lwp with regs from prgregset_t arg */
#define PCREAD   25L    /* read from the address space via priovec_t arg */
#define PCWRITE  26L    /* write to the address space via priovec_t arg */
#define PCSCRED  27L    /* set process credentials from prcred_t argument */
#define PCSASRS  28L    /* set ancillary state registers from asrset_t arg */
#define PCSPRIV  29L    /* set process privileges from prpriv_t argument */
#define PCSZONE  30L    /* set zoneid from zoneid_t argument */
#define PCSCREDX 31L    /* as PCSCRED but with supplemental groups */
/*
 * PCRUN long operand flags.
 */
#define PRCSIG          0x01    /* clear current signal, if any */
#define PRCFAULT        0x02    /* clear current fault, if any */
#define PRSTEP          0x04    /* direct the lwp to single-step */
#define PRSABORT        0x08    /* abort syscall, if in syscall */
#define PRSTOP          0x10    /* set directed stop request */

/*
 * lwp status file.  /proc/<pid>/lwp/<lwpid>/lwpstatus
 */
#define PRCLSZ          8       /* maximum size of scheduling class name */
#define PRSYSARGS       8       /* maximum number of syscall arguments */
typedef struct lwpstatus {
        int     pr_flags;       /* flags (see below) */
```

```
125         id_t    pr_lwpid;       /* specific lwp identifier */
126         short   pr_why;         /* reason for lwp stop, if stopped */
127         short   pr_what;        /* more detailed reason */
128         short   pr_cursig;      /* current signal, if any */
129         short   pr_pad1;
130         siginfo_t pr_info;      /* info associated with signal or fault */
131         sigset_t pr_lwppend;    /* set of signals pending to the lwp */
132         sigset_t pr_lwphold;    /* set of signals blocked by the lwp */
133         struct sigaction pr_action;     /* signal action for current signal */
134         stack_t pr_altstack;    /* alternate signal stack info */
135         uintptr_t pr_oldcontext;        /* address of previous ucontext */
136         short   pr_syscall;     /* system call number (if in syscall) */
137         short   pr_nsysarg;     /* number of arguments to this syscall */
138         int     pr_errno;       /* errno for failed syscall, 0 if successful */
139         long    pr_sysarg[PRSYSARGS];   /* arguments to this syscall */
140         long    pr_rval1;       /* primary syscall return value */
141         long    pr_rval2;       /* second syscall return value, if any */
142         char    pr_clname[PRCLSZ];      /* scheduling class name */
143         timestruc_t pr_tstamp;  /* real-time time stamp of stop */
144         timestruc_t pr_utime;   /* lwp user cpu time */
145         timestruc_t pr_stime;   /* lwp system cpu time */
146         int     pr_filler[11 - 2 * sizeof (timestruc_t) / sizeof (int)];
147         int     pr_errpriv;     /* missing privilege */
148         uintptr_t pr_ustack;    /* address of stack boundary data (stack_t) */
149         ulong_t pr_instr;       /* current instruction */
150         prgregset_t pr_reg;     /* general registers */
151         prfpregset_t pr_fpreg;  /* floating-point registers */
152 } lwpstatus_t;

154 /*
155  * process status file.  /proc/<pid>/status
156  */
157 typedef struct pstatus {
158         int     pr_flags;       /* flags (see below) */
159         int     pr_nlwp;        /* number of active lwps in the process */
160         pid_t   pr_pid;         /* process id */
161         pid_t   pr_ppid;        /* parent process id */
162         pid_t   pr_pgid;        /* process group id */
163         pid_t   pr_sid;         /* session id */
164         id_t    pr_aslwpid;     /* historical; now always zero */
165         id_t    pr_agentid;     /* lwp id of the /proc agent lwp, if any */
166         sigset_t pr_sigpend;    /* set of process pending signals */
167         uintptr_t pr_brkbase;   /* address of the process heap */
168         size_t  pr_brksize;     /* size of the process heap, in bytes */
169         uintptr_t pr_stkbase;   /* address of the process stack */
170         size_t  pr_stksize;     /* size of the process stack, in bytes */
171         timestruc_t pr_utime;   /* process user cpu time */
172         timestruc_t pr_stime;   /* process system cpu time */
173         timestruc_t pr_cutime;  /* sum of children's user times */
174         timestruc_t pr_cstime;  /* sum of children's system times */
175         sigset_t pr_sigtrace;   /* set of traced signals */
176         fltset_t pr_flttrace;   /* set of traced faults */
177         sysset_t pr_sysentry;   /* set of system calls traced on entry */
178         sysset_t pr_sysexit;    /* set of system calls traced on exit */
179         char    pr_dmodel;      /* data model of the process (see below) */
180         char    pr_pad[3];
181         taskid_t pr_taskid;     /* task id */
182         projid_t pr_projid;     /* project id */
183         int     pr_nzomb;       /* number of zombie lwps in the process */
184         zoneid_t pr_zoneid;     /* zone id */
185         psecflags_t pr_secflags; /* security flags */
186         int     pr_filler[13];  /* reserved for future use */
 67         int     pr_filler[15];  /* reserved for future use */
187         lwpstatus_t pr_lwp;     /* status of the representative lwp */
188 } pstatus_t;
```

_____unchanged_portion_omitted_


```
612 /*
613  * _ILP32 process status file.  /proc/<pid>/status
614  */
615 typedef struct pstatus32 {
616         int     pr_flags;       /* flags */
617         int     pr_nlwp;        /* number of active lwps in the process */
618         pid32_t pr_pid;         /* process id */
619         pid32_t pr_ppid;        /* parent process id */
620         pid32_t pr_pgid;        /* process group id */
621         pid32_t pr_sid;         /* session id */
622         id32_t  pr_aslwpid;     /* historical; now always zero */
623         id32_t  pr_agentid;     /* lwp id of the /proc agent lwp, if any */
624         sigset_t pr_sigpend;    /* set of process pending signals */
625         caddr32_t pr_brkbase;   /* address of the process heap */
626         size32_t pr_brksize;    /* size of the process heap, in bytes */
627         caddr32_t pr_stkbase;   /* address of the process stack */
628         size32_t pr_stksize;    /* size of the process stack, in bytes */
629         timestruc32_t pr_utime; /* process user cpu time */
630         timestruc32_t pr_stime; /* process system cpu time */
631         timestruc32_t pr_cutime;        /* sum of children's user times */
632         timestruc32_t pr_cstime;        /* sum of children's system times */
633         sigset_t pr_sigtrace;   /* set of traced signals */
634         fltset_t pr_flttrace;   /* set of traced faults */
635         sysset_t pr_sysentry;   /* set of system calls traced on entry */
636         sysset_t pr_sysexit;    /* set of system calls traced on exit */
637         char    pr_dmodel;      /* data model of the process */
638         char    pr_pad[3];
639         id32_t  pr_taskid;      /* task id */
640         id32_t  pr_projid;      /* project id */
641         int     pr_nzomb;       /* number of zombie lwps in the process */
642         id32_t  pr_zoneid;      /* zone id */
643         psecflags_t pr_secflags; /* security flags */
644         int     pr_filler[13];  /* reserved for future use */
524         int     pr_filler[15];  /* reserved for future use */
645         lwpstatus32_t pr_lwp;   /* status of the representative lwp */
646 } pstatus32_t;
```

_____unchanged_portion_omitted_

```
**********************************************************
    1040 Wed May 27 19:49:35 2015
new/usr/src/uts/common/sys/secflags.h
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
```

```
/*
 * This file and its contents are supplied under the terms of the
 * Common Development and Distribution License ("CDDL"), version 1.0.
 * You may only use this file in accordance with the terms of version
 * 1.0 of the CDDL.
 *
 * A full copy of the text of the CDDL should have accompanied this
 * source.  A copy of the CDDL is also available via the Internet at
 * http://www.illumos.org/license/CDDL.
 */

/* Copyright 2014, Richard Lowe */

#ifndef _SYS_SECFLAGS_H
#define _SYS_SECFLAGS_H

#ifdef __cplusplus
extern "C" {
#endif

#include <sys/types.h>

typedef struct psecflags {
        uint_t psf_effective;
        uint_t psf_inherit;
} psecflags_t;

/*
 * p_secflags codes
 *
 * These flags indicate the extra security-related features enabled for a
 * given process.
 */
#define PROC_SEC_ASLR   0x00000001
/* All valid bits */
#define PROC_SEC_MASK   (PROC_SEC_ASLR)

/* psecflags(2) commands */
typedef enum {
        PSECFLAGS_SET,
        PSECFLAGS_DISABLE,
        PSECFLAGS_ENABLE
} psecflags_cmd_t;

#ifdef __cplusplus
}
#endif

#endif /* _SYS_SECFLAGS_H */
#endif /* ! codereview */
```

```
**********************************************************
   13685 Wed May 27 19:49:35 2015
new/usr/src/uts/common/sys/syscall.h
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright 2014 Garrett D'Amore <garrett@damore.org>
 * Copyright (c) 1989, 2010, Oracle and/or its affiliates. All rights reserved.
 * Copyright (c) 2013 by Delphix. All rights reserved.
 * Copyright (c) 2015, Joyent, Inc.  All rights reserved.
 */

/*      Copyright (c) 1984, 1986, 1987, 1988, 1989 AT&T */
/*        All Rights Reserved   */

#ifndef _SYS_SYSCALL_H
#define _SYS_SYSCALL_H

#ifdef  __cplusplus
extern "C" {
#endif

/*
 *      system call numbers
 *              syscall(SYS_xxxx, ...)
 */

        /* syscall enumeration MUST begin with 1 */

        /*
         * SunOS/SPARC uses 0 for the indirect system call SYS_syscall
         * but this doesn't count because it is just another way
         * to specify the real system call number.
         */

#define SYS_syscall     0
#define SYS_exit        1
#define SYS_psecflags   2
#endif /* ! codereview */
#define SYS_read        3
#define SYS_write       4
#define SYS_open        5
```



```
#define SYS_close       6
#define SYS_linkat      7
#define SYS_link        9
#define SYS_unlink      10
#define SYS_symlinkat   11
#define SYS_chdir       12
#define SYS_time        13
#define SYS_mknod       14
#define SYS_chmod       15
#define SYS_chown       16
#define SYS_brk         17
#define SYS_stat        18
#define SYS_lseek       19
#define SYS_getpid      20
#define SYS_mount       21
#define SYS_readlinkat  22
#define SYS_setuid      23
#define SYS_getuid      24
#define SYS_stime       25
#define SYS_pcsample    26
#define SYS_alarm       27
#define SYS_fstat       28
#define SYS_pause       29
#define SYS_stty        31
#define SYS_gtty        32
#define SYS_access      33
#define SYS_nice        34
#define SYS_statfs      35
#define SYS_sync        36
#define SYS_kill        37
#define SYS_fstatfs     38
#define SYS_pgrpsys     39
        /*
         * subcodes:
         *      getpgrp()         :: syscall(39,0)
         *      setpgrp()         :: syscall(39,1)
         *      getsid(pid)       :: syscall(39,2,pid)
         *      setsid()          :: syscall(39,3)
         *      getpgid(pid)      :: syscall(39,4,pid)
         *      setpgid(pid,pgid) :: syscall(39,5,pid,pgid)
         */
#define SYS_uucopystr   40
#define SYS_pipe        42
#define SYS_times       43
#define SYS_profil      44
#define SYS_faccessat   45
#define SYS_setgid      46
#define SYS_getgid      47
#define SYS_mknodat     48
#define SYS_msgsys      49
        /*
         * subcodes:
         *      msgget(...)  :: msgsys(0, ...)
         *      msgctl(...)  :: msgsys(1, ...)
         *      msgrcv(...)  :: msgsys(2, ...)
         *      msgsnd(...)  :: msgsys(3, ...)
         *      msgids(...)  :: msgsys(4, ...)
         *      msgsnap(...) :: msgsys(5, ...)
         *      see <sys/msg.h>
         */
#define SYS_sysi86      50
        /*
         * subcodes:
         *      sysi86(code, ...)
         */
#define SYS_acct        51
```

```
#define SYS_shmsys      52
        /*
         * subcodes:
         *      shmat (...) :: shmsys(0, ...)
         *      shmctl(...) :: shmsys(1, ...)
         *      shmdt (...) :: shmsys(2, ...)
         *      shmget(...) :: shmsys(3, ...)
         *      shmids(...) :: shmsys(4, ...)
         *      see <sys/shm.h>
         */
#define SYS_semsys      53
        /*
         * subcodes:
         *      semctl(...) :: semsys(0, ...)
         *      semget(...) :: semsys(1, ...)
         *      semop (...) :: semsys(2, ...)
         *      semids(...) :: semsys(3, ...)
         *      semtimedop(...) :: semsys(4, ...)
         *      see <sys/sem.h>
         */
#define SYS_ioctl       54
#define SYS_uadmin      55
#define SYS_fchownat    56
#define SYS_utssys      57
        /*
         * subcodes (third argument):
         *      uname(obuf)  (obsolete)   :: syscall(57, obuf, ign, 0)
         *                                   subcode 1 unused
         *      ustat(dev, obuf)          :: syscall(57, obuf, dev, 2)
         *      fusers(path, flags, obuf) :: syscall(57, path, flags, 3, obuf)
         *      see <sys/utssys.h>
         */
#define SYS_fdsync      58
#define SYS_execve      59
#define SYS_umask       60
#define SYS_chroot      61
#define SYS_fcntl       62
#define SYS_ulimit      63
#define SYS_renameat    64
#define SYS_unlinkat    65
#define SYS_fstatat     66
#define SYS_fstatat64   67
#define SYS_openat      68
#define SYS_openat64    69
#define SYS_tasksys     70
        /*
         * subcodes:
         *      settaskid(...) :: tasksys(0, ...)
         *      gettaskid(...) :: tasksys(1, ...)
         *      getprojid(...) :: tasksys(2, ...)
         */
#define SYS_acctctl     71
#define SYS_exacctsys   72
        /*
         * subcodes:
         *      getacct(...) :: exacct(0, ...)
         *      putacct(...) :: exacct(1, ...)
         *      wracct(...) :: exacct(2, ...)
         */
#define SYS_getpagesizes        73
        /*
         * subcodes:
         *      getpagesizes2(...)      :: getpagesizes(0, ...)
         *      getpagesizes(...)       :: getpagesizes(1, ...) legacy
         */
#define SYS_rctlsys     74
```



```
        /*
         * subcodes:
         *      getrctl(...) :: rctlsys(0, ...)
         *      setrctl(...) :: rctlsys(1, ...)
         *      rctllist(...) :: rctlsys(2, ...)
         *      rctlctl(...) :: rctlsys(3, ...)
         */
#define SYS_sidsys      75
        /*
         * subcodes:
         *      allocids(...)           :: sidsys(0, ...)
         *      idmap_reg(...)          :: sidsys(1, ...)
         *      idmap_unreg(...)        :: sidsys(2, ...)
         */
#define SYS_lwp_park    77
        /*
         * subcodes:
         *      _lwp_park(timespec_t *, lwpid_t)        :: syslwp_park(0, ...)
         *      _lwp_unpark(lwpid_t, int)               :: syslwp_park(1, ...)
         *      _lwp_unpark_all(lwpid_t *, int)         :: syslwp_park(2, ...)
         *      _lwp_unpark_cancel(lwpid_t *, int)      :: syslwp_park(3, ...)
         *      _lwp_set_park(lwpid_t *, int)           :: syslwp_park(4, ...)
         */
#define SYS_sendfilev   78
        /*
         * subcodes :
         *      sendfilev()   :: sendfilev(0, ...)
         *      sendfilev64() :: sendfilev(1, ...)
         */
#define SYS_rmdir       79
#define SYS_mkdir       80
#define SYS_getdents    81
#define SYS_privsys     82
        /*
         * subcodes:
         *      setppriv(...)           :: privsys(0, ...)
         *      getppriv(...)           :: privsys(1, ...)
         *      getimplinfo(...)        :: privsys(2, ...)
         *      setpflags(...)          :: privsys(3, ...)
         *      getpflags(...)          :: privsys(4, ...)
         *      issetugid();            :: privsys(5)
         */
#define SYS_ucredsys    83
        /*
         * subcodes:
         *      ucred_get(...)          :: ucredsys(0, ...)
         *      getpeerucred(...)       :: ucredsys(1, ...)
         */
#define SYS_sysfs       84
        /*
         * subcodes:
         *      sysfs(code, ...)
         *      see <sys/fstyp.h>
         */
#define SYS_getmsg      85
#define SYS_putmsg      86
#define SYS_lstat       88
#define SYS_symlink     89
#define SYS_readlink    90
#define SYS_setgroups   91
#define SYS_getgroups   92
#define SYS_fchmod      93
#define SYS_fchown      94
#define SYS_sigprocmask 95
#define SYS_sigsuspend  96
#define SYS_sigaltstack 97
```

```
257 #define SYS_sigaction   98
258 #define SYS_sigpending  99
259         /*
260          * subcodes:
261          *                        subcode 0 unused
262          *      sigpending(...) :: syscall(99, 1, ...)
263          *      sigfillset(...) :: syscall(99, 2, ...)
264          */
265 #define SYS_context     100
266         /*
267          * subcodes:
268          *      getcontext(...) :: syscall(100, 0, ...)
269          *      setcontext(...) :: syscall(100, 1, ...)
270          */
271 #define SYS_fchmodat    101
272 #define SYS_mkdirat     102
273 #define SYS_statvfs     103
274 #define SYS_fstatvfs    104
275 #define SYS_getloadavg  105
276 #define SYS_nfssys      106
277 #define SYS_waitid      107
278 #define SYS_waitsys     SYS_waitid      /* historical */
279 #define SYS_sigsendsys  108
280 #define SYS_hrtsys      109
281 #define SYS_utimesys    110
282         /*
283          * subcodes:
284          *      futimens(...)   :: syscall(110, 0, ...)
285          *      utimensat(...)  :: syscall(110, 1, ...)
286          */
287 #define SYS_sigresend   111
288 #define SYS_priocntlsys 112
289 #define SYS_pathconf    113
290 #define SYS_mincore     114
291 #define SYS_mmap        115
292 #define SYS_mprotect    116
293 #define SYS_munmap      117
294 #define SYS_fpathconf   118
295 #define SYS_vfork       119
296 #define SYS_fchdir      120
297 #define SYS_readv       121
298 #define SYS_writev      122
299 #define SYS_preadv      123
300 #define SYS_pwritev     124
301 #define SYS_getrandom   126
302 #define SYS_mmapobj     127
303 #define SYS_setrlimit   128
304 #define SYS_getrlimit   129
305 #define SYS_lchown      130
306 #define SYS_memcntl     131
307 #define SYS_getpmsg     132
308 #define SYS_putpmsg     133
309 #define SYS_rename      134
310 #define SYS_uname       135
311 #define SYS_setegid     136
312 #define SYS_sysconfig   137
313 #define SYS_adjtime     138
314 #define SYS_systeminfo  139
315 #define SYS_sharefs     140
316 #define SYS_seteuid     141
317 #define SYS_forksys     142
318         /*
319          * subcodes:
320          *      forkx(flags)    :: forksys(0, flags)
321          *      forkallx(flags) :: forksys(1, flags)
322          *      vforkx(flags)   :: forksys(2, flags)
```



```
323          */
324 #define SYS_sigtimedwait        144
325 #define SYS_lwp_info    145
326 #define SYS_yield       146
327 #define SYS_lwp_sema_post       148
328 #define SYS_lwp_sema_trywait    149
329 #define SYS_lwp_detach  150
330 #define SYS_corectl     151
331 #define SYS_modctl      152
332 #define SYS_fchroot     153
333 #define SYS_vhangup     155
334 #define SYS_gettimeofday        156
335 #define SYS_getitimer           157
336 #define SYS_setitimer           158
337 #define SYS_lwp_create          159
338 #define SYS_lwp_exit            160
339 #define SYS_lwp_suspend         161
340 #define SYS_lwp_continue        162
341 #define SYS_lwp_kill            163
342 #define SYS_lwp_self            164
343 #define SYS_lwp_sigmask         165
344 #define SYS_lwp_private         166
345 #define SYS_lwp_wait            167
346 #define SYS_lwp_mutex_wakeup    168
347 #define SYS_lwp_cond_wait       170
348 #define SYS_lwp_cond_signal     171
349 #define SYS_lwp_cond_broadcast  172
350 #define SYS_pread               173
351 #define SYS_pwrite              174
352 #define SYS_llseek              175
353 #define SYS_inst_sync           176
354 #define SYS_brand               177
355 #define SYS_kaio                178
356         /*
357          * subcodes:
358          *      aioread(...)    :: kaio(AIOREAD, ...)
359          *      aiowrite(...)   :: kaio(AIOWRITE, ...)
360          *      aiowait(...)    :: kaio(AIOWAIT, ...)
361          *      aiocancel(...)  :: kaio(AIOCANCEL, ...)
362          *      aionotify()     :: kaio(AIONOTIFY)
363          *      aioinit()       :: kaio(AIOINIT)
364          *      aiostart()      :: kaio(AIOSTART)
365          *      see <sys/aio.h>
366          */
367 #define SYS_cpc                 179
368 #define SYS_lgrpsys             180
369 #define SYS_meminfosys          SYS_lgrpsys
370         /*
371          * subcodes:
372          *      meminfo(...) :: meminfosys(MISYS_MEMINFO, ...)
373          */
374 #define SYS_rusagesys           181
375         /*
376          * subcodes:
377          *      getrusage(...) :: rusagesys(RUSAGESYS_GETRUSAGE, ...)
378          *      getvmusage(...)    :: rusagesys(RUSAGESYS_GETVMUSAGE, ...)
379          */
380 #define SYS_port                182
381         /*
382          * subcodes:
383          *      port_create(...) :: portfs(PORT_CREATE, ...)
384          *      port_associate(...) :: portfs(PORT_ASSOCIATE, ...)
385          *      port_dissociate(...) :: portfs(PORT_DISSOCIATE, ...)
386          *      port_send(...) :: portfs(PORT_SEND, ...)
387          *      port_sendn(...) :: portfs(PORT_SENDN, ...)
388          *      port_get(...) :: portfs(PORT_GET, ...)
```

```
389          *      port_getn(...) :: portfs(PORT_GETN, ...)
390          *      port_alert(...) :: portfs(PORT_ALERT, ...)
391          *      port_dispatch(...) :: portfs(PORT_DISPATCH, ...)
392          */
393 #define SYS_pollsys             183
394 #define SYS_labelsys            184
395 #define SYS_acl                 185
396 #define SYS_auditsys            186
397 #define SYS_processor_bind      187
398 #define SYS_processor_info      188
399 #define SYS_p_online            189
400 #define SYS_sigqueue            190
401 #define SYS_clock_gettime       191
402 #define SYS_clock_settime       192
403 #define SYS_clock_getres        193
404 #define SYS_timer_create        194
405 #define SYS_timer_delete        195
406 #define SYS_timer_settime       196
407 #define SYS_timer_gettime       197
408 #define SYS_timer_getoverrun    198
409 #define SYS_nanosleep           199
410 #define SYS_facl                200
411 #define SYS_door                201
412         /*
413          * Door Subcodes:
414          *      0       door_create
415          *      1       door_revoke
416          *      2       door_info
417          *      3       door_call
418          *      4       door_return
419          */
420 #define SYS_setreuid            202
421 #define SYS_setregid            203
422 #define SYS_install_utrap       204
423 #define SYS_signotify           205
424 #define SYS_schedctl            206
425 #define SYS_pset                207
426 #define SYS_sparc_utrap_install 208
427 #define SYS_resolvepath         209
428 #define SYS_lwp_mutex_timedlock 210
429 #define SYS_lwp_sema_timedwait  211
430 #define SYS_lwp_rwlock_sys      212
431         /*
432          * subcodes:
433          *      lwp_rwlock_rdlock(...)    :: syscall(212, 0, ...)
434          *      lwp_rwlock_wrlock(...)    :: syscall(212, 1, ...)
435          *      lwp_rwlock_tryrdlock(...) :: syscall(212, 2, ...)
436          *      lwp_rwlock_trywrlock(...) :: syscall(212, 3, ...)
437          *      lwp_rwlock_unlock(...)    :: syscall(212, 4, ...)
438          */
439 /* system calls for large file ( > 2 gigabyte) support */
440 #define SYS_getdents64          213
441 #define SYS_mmap64              214
442 #define SYS_stat64              215
443 #define SYS_lstat64             216
444 #define SYS_fstat64             217
445 #define SYS_statvfs64           218
446 #define SYS_fstatvfs64          219
447 #define SYS_setrlimit64         220
448 #define SYS_getrlimit64         221
449 #define SYS_pread64             222
450 #define SYS_pwrite64            223
451 #define SYS_open64              225
452 #define SYS_rpcsys              226
453 #define SYS_zone                227
454         /*
```



```
455          * subcodes:
456          *      zone_create(...) :: zone(ZONE_CREATE, ...)
457          *      zone_destroy(...) :: zone(ZONE_DESTROY, ...)
458          *      zone_getattr(...) :: zone(ZONE_GETATTR, ...)
459          *      zone_enter(...) :: zone(ZONE_ENTER, ...)
460          *      zone_list(...) :: zone(ZONE_LIST, ...)
461          *      zone_shutdown(...) :: zone(ZONE_SHUTDOWN, ...)
462          *      zone_lookup(...) :: zone(ZONE_LOOKUP, ...)
463          *      zone_boot(...) :: zone(ZONE_BOOT, ...)
464          *      zone_version(...) :: zone(ZONE_VERSION, ...)
465          *      zone_setattr(...) :: zone(ZONE_SETATTR, ...)
466          *      zone_add_datalink(...) :: zone(ZONE_ADD_DATALINK, ...)
467          *      zone_remove_datalink(...) :: zone(ZONE_DEL_DATALINK, ...)
468          *      zone_check_datalink(...) :: zone(ZONE_CHECK_DATALINK, ...)
469          *      zone_list_datalink(...) :: zone(ZONE_LIST_DATALINK, ...)
470          */
471 #define SYS_autofssys           228
472 #define SYS_getcwd              229
473 #define SYS_so_socket           230
474 #define SYS_so_socketpair       231
475 #define SYS_bind                232
476 #define SYS_listen              233
477 #define SYS_accept              234
478 #define SYS_connect             235
479 #define SYS_shutdown            236
480 #define SYS_recv                237
481 #define SYS_recvfrom            238
482 #define SYS_recvmsg             239
483 #define SYS_send                240
484 #define SYS_sendmsg             241
485 #define SYS_sendto              242
486 #define SYS_getpeername         243
487 #define SYS_getsockname         244
488 #define SYS_getsockopt          245
489 #define SYS_setsockopt          246
490 #define SYS_sockconfig          247
491         /*
492          * NTP codes
493          */
494 #define SYS_ntp_gettime         248
495 #define SYS_ntp_adjtime         249
496 #define SYS_lwp_mutex_unlock    250
497 #define SYS_lwp_mutex_trylock   251
498 #define SYS_lwp_mutex_register  252
499 #define SYS_cladm               253
500 #define SYS_uucopy              254
501 #define SYS_umount2             255

503 #ifndef _ASM

505 typedef struct {                /* syscall set type */
506         unsigned int    word[16];
507 } sysset_t;

509 typedef struct {        /* return values from system call */
510         long    sys_rval1;      /* primary return value from system call */
511         long    sys_rval2;      /* second return value from system call */
512 } sysret_t;

514 #if !defined(_KERNEL)

516 extern int      syscall(int, ...);
517 extern int      __systemcall(sysret_t *, int, ...);
518 extern int      __set_errno(int);

520 #endif  /* _KERNEL */
```

```
522 #endif  /* _ASM */

524 #ifdef  __cplusplus
525 }
526 #endif

528 #endif  /* _SYS_SYSCALL_H */
```

```
**********************************************************
    1987 Wed May 27 19:49:36 2015
new/usr/src/uts/common/syscall/psecflags.c
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
```

```
/*
 * This file and its contents are supplied under the terms of the
 * Common Development and Distribution License ("CDDL"), version 1.0.
 * You may only use this file in accordance with the terms of version
 * 1.0 of the CDDL.
 *
 * A full copy of the text of the CDDL should have accompanied this
 * source.  A copy of the CDDL is also available via the Internet at
 * http://www.illumos.org/license/CDDL.
 */

/*
 * Copyright 2014 <contributor>.  All rights reserved.
 */

#include <sys/ddi.h>
#include <sys/errno.h>
#include <sys/policy.h>
#include <sys/proc.h>
#include <sys/procset.h>
#include <sys/systm.h>
#include <sys/types.h>

struct psecargs {
        psecflags_cmd_t cmd;
        uint_t val;
};

static int
psecdo(proc_t *p, struct psecargs *args)
{
        mutex_enter(&p->p_lock);

        if (secpolicy_psecflags(CRED(), p, curproc) != 0) {
                mutex_exit(&p->p_lock);
                return (EPERM);
        }

        switch (args->cmd) {
        case PSECFLAGS_SET:
                secflag_set(p, args->val);
                break;
        case PSECFLAGS_DISABLE:
                secflag_disable(p, args->val);
                break;
        case PSECFLAGS_ENABLE:
                secflag_enable(p, args->val);
                break;
        }
        mutex_exit(&p->p_lock);

        return (0);
}

int
psecflags(procset_t *psp, psecflags_cmd_t cmd, uint_t arg)
{
        procset_t procset;
```



```
        struct psecargs args = { 0 };
        int rv = 0;

        /*
         * We don't check the validity in 'arg' for the sake of newer
         * software on older systems not just falling down dead.
         *
         * XXX: I'm not particularly confident that's not dumb.
         */

        if (copyin(psp, &procset, sizeof (procset)) != 0)
                return (set_errno(EFAULT));

        /* secflags are per-process, procset must be in terms of processes */
        if ((procset.p_lidtype == P_LWPID) ||
            (procset.p_ridtype == P_LWPID))
                return (set_errno(EINVAL));

        switch (cmd) {
        case PSECFLAGS_SET:
        case PSECFLAGS_DISABLE:
        case PSECFLAGS_ENABLE:
                args.cmd = cmd;
                args.val = arg;
                rv = dotoprocs(&procset, psecdo,
                    (caddr_t)&args);
                break;
        default:
                return (set_errno(EINVAL));
        }

        return (rv ? set_errno(rv) : 0);
}
#endif /* ! codereview */
```

```
**********************************************************
   98025 Wed May 27 19:49:36 2015
new/usr/src/uts/i86pc/vm/vm_machdep.c
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
```

```
   1 /*
   2  * CDDL HEADER START
   3  *
   4  * The contents of this file are subject to the terms of the
   5  * Common Development and Distribution License (the "License").
   6  * You may not use this file except in compliance with the License.
   7  *
   8  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
   9  * or http://www.opensolaris.org/os/licensing.
  10  * See the License for the specific language governing permissions
  11  * and limitations under the License.
  12  *
  13  * When distributing Covered Code, include this CDDL HEADER in each
  14  * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  15  * If applicable, add the following below this CDDL HEADER, with the
  16  * fields enclosed by brackets "[]" replaced with your own identifying
  17  * information: Portions Copyright [yyyy] [name of copyright owner]
  18  *
  19  * CDDL HEADER END
  20  */
  21 /*
  22  * Copyright (c) 1992, 2010, Oracle and/or its affiliates. All rights reserved.
  23  */
  24 /*
  25  * Copyright (c) 2010, Intel Corporation.
  26  * All rights reserved.
  27  */

  29 /* Copyright (c) 1984, 1986, 1987, 1988, 1989 AT&T */
  30 /*       All Rights Reserved   */

  32 /*
  33  * Portions of this source code were derived from Berkeley 4.3 BSD
  34  * under license from the Regents of the University of California.
  35  */

  37 /*
  38  * UNIX machine dependent virtual memory support.
  39  */

  41 #include <sys/types.h>
  42 #include <sys/param.h>
  43 #include <sys/systm.h>
  44 #include <sys/user.h>
  45 #include <sys/proc.h>
  46 #include <sys/kmem.h>
  47 #include <sys/vmem.h>
  48 #include <sys/buf.h>
  49 #include <sys/cpuvar.h>
  50 #include <sys/lgrp.h>
  51 #include <sys/disp.h>
  52 #include <sys/vm.h>
  53 #include <sys/mman.h>
  54 #include <sys/vnode.h>
  55 #include <sys/cred.h>
  56 #include <sys/exec.h>
  57 #include <sys/exechdr.h>
  58 #include <sys/debug.h>
```



```
  59 #include <sys/vmsystm.h>
  60 #include <sys/swap.h>
  61 #include <sys/dumphdr.h>
  62 #include <sys/random.h>
  63 #endif /* ! codereview */

  65 #include <vm/hat.h>
  66 #include <vm/as.h>
  67 #include <vm/seg.h>
  68 #include <vm/seg_kp.h>
  69 #include <vm/seg_vn.h>
  70 #include <vm/page.h>
  71 #include <vm/seg_kmem.h>
  72 #include <vm/seg_kpm.h>
  73 #include <vm/vm_dep.h>

  75 #include <sys/cpu.h>
  76 #include <sys/vm_machparam.h>
  77 #include <sys/memlist.h>
  78 #include <sys/bootconf.h> /* XXX the memlist stuff belongs in memlist_plat.h */
  79 #include <vm/hat_i86.h>
  80 #include <sys/x86_archext.h>
  81 #include <sys/elf_386.h>
  82 #include <sys/cmn_err.h>
  83 #include <sys/archsystm.h>
  84 #include <sys/machsystm.h>

  86 #include <sys/vtrace.h>
  87 #include <sys/ddidmareq.h>
  88 #include <sys/promif.h>
  89 #include <sys/memnode.h>
  90 #include <sys/stack.h>
  91 #include <util/qsort.h>
  92 #include <sys/taskq.h>

  94 #ifdef __xpv

  96 #include <sys/hypervisor.h>
  97 #include <sys/xen_mmu.h>
  98 #include <sys/balloon_impl.h>

 100 /*
 101  * domain 0 pages usable for DMA are kept pre-allocated and kept in
 102  * distinct lists, ordered by increasing mfn.
 103  */
 104 static kmutex_t io_pool_lock;
 105 static kmutex_t contig_list_lock;
 106 static page_t *io_pool_4g;      /* pool for 32 bit dma limited devices */
 107 static page_t *io_pool_16m;     /* pool for 24 bit dma limited legacy devices */
 108 static long io_pool_cnt;
 109 static long io_pool_cnt_max = 0;
 110 #define DEFAULT_IO_POOL_MIN     128
 111 static long io_pool_cnt_min = DEFAULT_IO_POOL_MIN;
 112 static long io_pool_cnt_lowater = 0;
 113 static long io_pool_shrink_attempts; /* how many times did we try to shrink */
 114 static long io_pool_shrinks;    /* how many times did we really shrink */
 115 static long io_pool_grows;      /* how many times did we grow */
 116 static mfn_t start_mfn = 1;
 117 static caddr_t io_pool_kva;     /* use to alloc pages when needed */

 119 static int create_contig_pfnlist(uint_t);

 121 /*
 122  * percentage of phys mem to hold in the i/o pool
 123  */
 124 #define DEFAULT_IO_POOL_PCT     2
```

```
125 static long io_pool_physmem_pct = DEFAULT_IO_POOL_PCT;
126 static void page_io_pool_sub(page_t **, page_t *, page_t *);
127 int ioalloc_dbg = 0;

129 #endif /* __xpv */

131 uint_t vac_colors = 1;

133 int largepagesupport = 0;
134 extern uint_t page_create_new;
135 extern uint_t page_create_exists;
136 extern uint_t page_create_putbacks;
137 /*
138  * Allow users to disable the kernel's use of SSE.
139  */
140 extern int use_sse_pagecopy, use_sse_pagezero;

142 /*
143  * combined memory ranges from mnode and memranges[] to manage single
144  * mnode/mtype dimension in the page lists.
145  */
146 typedef struct {
147         pfn_t   mnr_pfnlo;
148         pfn_t   mnr_pfnhi;
149         int     mnr_mnode;
150         int     mnr_memrange;           /* index into memranges[] */
151         int     mnr_next;               /* next lower PA mnoderange */
152         int     mnr_exists;
153         /* maintain page list stats */
154         pgcnt_t mnr_mt_clpgcnt;         /* cache list cnt */
155         pgcnt_t mnr_mt_flpgcnt[MMU_PAGE_SIZES]; /* free list cnt per szc */
156         pgcnt_t mnr_mt_totcnt;          /* sum of cache and free lists */
157 #ifdef DEBUG
158         struct mnr_mts {                /* mnode/mtype szc stats */
159                 pgcnt_t mnr_mts_pgcnt;
160                 int     mnr_mts_colors;
161                 pgcnt_t *mnr_mtsc_pgcnt;
162         }       *mnr_mts;
163 #endif
164 } mnoderange_t;

166 #define MEMRANGEHI(mtype)                                               \
167         ((mtype > 0) ? memranges[mtype - 1] - 1: physmax)
168 #define MEMRANGELO(mtype)       (memranges[mtype])

170 #define MTYPE_FREEMEM(mt)       (mnoderanges[mt].mnr_mt_totcnt)

172 /*
173  * As the PC architecture evolved memory up was clumped into several
174  * ranges for various historical I/O devices to do DMA.
175  * < 16Meg - ISA bus
176  * < 2Gig - ???
177  * < 4Gig - PCI bus or drivers that don't understand PAE mode
178  *
179  * These are listed in reverse order, so that we can skip over unused
180  * ranges on machines with small memories.
181  *
182  * For now under the Hypervisor, we'll only ever have one memrange.
183  */
184 #define PFN_4GIG        0x100000
185 #define PFN_16MEG       0x1000
186 /* Indices into the memory range (arch_memranges) array. */
187 #define MRI_4G          0
188 #define MRI_2G          1
189 #define MRI_16M         2
190 #define MRI_0           3
```



```
191 static pfn_t arch_memranges[NUM_MEM_RANGES] = {
192     PFN_4GIG,   /* pfn range for 4G and above */
193     0x80000,    /* pfn range for 2G-4G */
194     PFN_16MEG,  /* pfn range for 16M-2G */
195     0x00000,    /* pfn range for 0-16M */
196 };
197 pfn_t *memranges = &arch_memranges[0];
198 int nranges = NUM_MEM_RANGES;

200 /*
201  * This combines mem_node_config and memranges into one data
202  * structure to be used for page list management.
203  */
204 mnoderange_t    *mnoderanges;
205 int             mnoderangecnt;
206 int             mtype4g;
207 int             mtype16m;
208 int             mtypetop;       /* index of highest pfn'ed mnoderange */

210 /*
211  * 4g memory management variables for systems with more than 4g of memory:
212  *
213  * physical memory below 4g is required for 32bit dma devices and, currently,
214  * for kmem memory. On systems with more than 4g of memory, the pool of memory
215  * below 4g can be depleted without any paging activity given that there is
216  * likely to be sufficient memory above 4g.
217  *
218  * physmax4g is set true if the largest pfn is over 4g. The rest of the
219  * 4g memory management code is enabled only when physmax4g is true.
220  *
221  * maxmem4g is the count of the maximum number of pages on the page lists
222  * with physical addresses below 4g. It can be a lot less then 4g given that
223  * BIOS may reserve large chunks of space below 4g for hot plug pci devices,
224  * agp aperture etc.
225  *
226  * freemem4g maintains the count of the number of available pages on the
227  * page lists with physical addresses below 4g.
228  *
229  * DESFREE4G specifies the desired amount of below 4g memory. It defaults to
230  * 6% (desfree4gshift = 4) of maxmem4g.
231  *
232  * RESTRICT4G_ALLOC returns true if freemem4g falls below DESFREE4G
233  * and the amount of physical memory above 4g is greater than freemem4g.
234  * In this case, page_get_* routines will restrict below 4g allocations
235  * for requests that don't specifically require it.
236  */

238 #define DESFREE4G       (maxmem4g >> desfree4gshift)

240 #define RESTRICT4G_ALLOC                                                \
241         (physmax4g && (freemem4g < DESFREE4G) && ((freemem4g << 1) < freemem))

243 static pgcnt_t  maxmem4g;
244 static pgcnt_t  freemem4g;
245 static int      physmax4g;
246 static int      desfree4gshift = 4;     /* maxmem4g shift to derive DESFREE4G */

248 /*
249  * 16m memory management:
250  *
251  * reserve some amount of physical memory below 16m for legacy devices.
252  *
253  * RESTRICT16M_ALLOC returns true if an there are sufficient free pages above
254  * 16m or if the 16m pool drops below DESFREE16M.
255  *
256  * In this case, general page allocations via page_get_{free,cache}list
```

```
257  * routines will be restricted from allocating from the 16m pool. Allocations
258  * that require specific pfn ranges (page_get_anylist) and PG_PANIC allocations
259  * are not restricted.
260  */

262 #define FREEMEM16M      MTYPE_FREEMEM(mtype16m)
263 #define DESFREE16M      desfree16m
264 #define RESTRICT16M_ALLOC(freemem, pgcnt, flags)                \
265         ((freemem != 0) && ((flags & PG_PANIC) == 0) &&         \
266             ((freemem >= (FREEMEM16M)) ||                       \
267             (FREEMEM16M  < (DESFREE16M + pgcnt))))

269 static pgcnt_t  desfree16m = 0x380;

271 /*
272  * This can be patched via /etc/system to allow old non-PAE aware device
273  * drivers to use kmem_alloc'd memory on 32 bit systems with > 4Gig RAM.
274  */
275 int restricted_kmemalloc = 0;

277 #ifdef VM_STATS
278 struct {
279         ulong_t pga_alloc;
280         ulong_t pga_notfullrange;
281         ulong_t pga_nulldmaattr;
282         ulong_t pga_allocok;
283         ulong_t pga_allocfailed;
284         ulong_t pgma_alloc;
285         ulong_t pgma_allocok;
286         ulong_t pgma_allocfailed;
287         ulong_t pgma_allocempty;
288 } pga_vmstats;
289 #endif

291 uint_t mmu_page_sizes;

293 /* How many page sizes the users can see */
294 uint_t mmu_exported_page_sizes;

296 /* page sizes that legacy applications can see */
297 uint_t mmu_legacy_page_sizes;

299 /*
300  * Number of pages in 1 GB.  Don't enable automatic large pages if we have
301  * fewer than this many pages.
302  */
303 pgcnt_t shm_lpg_min_physmem = 1 << (30 - MMU_PAGESHIFT);
304 pgcnt_t privm_lpg_min_physmem = 1 << (30 - MMU_PAGESHIFT);

306 /*
307  * Maximum and default segment size tunables for user private
308  * and shared anon memory, and user text and initialized data.
309  * These can be patched via /etc/system to allow large pages
310  * to be used for mapping application private and shared anon memory.
311  */
312 size_t mcntl0_lpsize = MMU_PAGESIZE;
313 size_t max_uheap_lpsize = MMU_PAGESIZE;
314 size_t default_uheap_lpsize = MMU_PAGESIZE;
315 size_t max_ustack_lpsize = MMU_PAGESIZE;
316 size_t default_ustack_lpsize = MMU_PAGESIZE;
317 size_t max_privmap_lpsize = MMU_PAGESIZE;
318 size_t max_uidata_lpsize = MMU_PAGESIZE;
319 size_t max_utext_lpsize = MMU_PAGESIZE;
320 size_t max_shm_lpsize = MMU_PAGESIZE;
```



```
323 /*
324  * initialized by page_coloring_init().
325  */
326 uint_t  page_colors;
327 uint_t  page_colors_mask;
328 uint_t  page_coloring_shift;
329 int     cpu_page_colors;
330 static uint_t   l2_colors;

332 /*
333  * Page freelists and cachelists are dynamically allocated once mnoderangecnt
334  * and page_colors are calculated from the l2 cache n-way set size.  Within a
335  * mnode range, the page freelist and cachelist are hashed into bins based on
336  * color. This makes it easier to search for a page within a specific memory
337  * range.
338  */
339 #define PAGE_COLORS_MIN 16

341 page_t ****page_freelists;
342 page_t ***page_cachelists;


345 /*
346  * Used by page layer to know about page sizes
347  */
348 hw_pagesize_t hw_page_array[MAX_NUM_LEVEL + 1];

350 kmutex_t        *fpc_mutex[NPC_MUTEX];
351 kmutex_t        *cpc_mutex[NPC_MUTEX];

353 /* Lock to protect mnoderanges array for memory DR operations. */
354 static kmutex_t mnoderange_lock;

356 /*
357  * Only let one thread at a time try to coalesce large pages, to
358  * prevent them from working against each other.
359  */
360 static kmutex_t contig_lock;
361 #define CONTIG_LOCK()   mutex_enter(&contig_lock);
362 #define CONTIG_UNLOCK() mutex_exit(&contig_lock);

364 #define PFN_16M         (mmu_btop((uint64_t)0x1000000))

366 /*
367  * Return the optimum page size for a given mapping
368  */
369 /*ARGSUSED*/
370 size_t
371 map_pgsz(int maptype, struct proc *p, caddr_t addr, size_t len, int memcntl)
372 {
373         level_t l = 0;
374         size_t pgsz = MMU_PAGESIZE;
375         size_t max_lpsize;
376         uint_t mszc;

378         ASSERT(maptype != MAPPGSZ_VA);

380         if (maptype != MAPPGSZ_ISM && physmem < privm_lpg_min_physmem) {
381                 return (MMU_PAGESIZE);
382         }

384         switch (maptype) {
385         case MAPPGSZ_HEAP:
386         case MAPPGSZ_STK:
387                 max_lpsize = memcntl ? mcntl0_lpsize : (maptype ==
388                     MAPPGSZ_HEAP ? max_uheap_lpsize : max_ustack_lpsize);
```

```
389                 if (max_lpsize == MMU_PAGESIZE) {
390                         return (MMU_PAGESIZE);
391                 }
392                 if (len == 0) {
393                         len = (maptype == MAPPGSZ_HEAP) ? p->p_brkbase +
394                             p->p_brksize - p->p_bssbase : p->p_stksize;
395                 }
396                 len = (maptype == MAPPGSZ_HEAP) ? MAX(len,
397                     default_uheap_lpsize) : MAX(len, default_ustack_lpsize);

399                 /*
400                  * use the pages size that best fits len
401                  */
402                 for (l = mmu.umax_page_level; l > 0; --l) {
403                         if (LEVEL_SIZE(l) > max_lpsize || len < LEVEL_SIZE(l)) {
404                                 continue;
405                         } else {
406                                 pgsz = LEVEL_SIZE(l);
407                         }
408                         break;
409                 }

411                 mszc = (maptype == MAPPGSZ_HEAP ? p->p_brkpageszc :
412                     p->p_stkpageszc);
413                 if (addr == 0 && (pgsz < hw_page_array[mszc].hp_size)) {
414                         pgsz = hw_page_array[mszc].hp_size;
415                 }
416                 return (pgsz);

418         case MAPPGSZ_ISM:
419                 for (l = mmu.umax_page_level; l > 0; --l) {
420                         if (len >= LEVEL_SIZE(l))
421                                 return (LEVEL_SIZE(l));
422                 }
423                 return (LEVEL_SIZE(0));
424         }
425         return (pgsz);
426 }

428 static uint_t
429 map_szcvec(caddr_t addr, size_t size, uintptr_t off, size_t max_lpsize,
430     size_t min_physmem)
431 {
432         caddr_t eaddr = addr + size;
433         uint_t szcvec = 0;
434         caddr_t raddr;
435         caddr_t readdr;
436         size_t  pgsz;
437         int i;

439         if (physmem < min_physmem || max_lpsize <= MMU_PAGESIZE) {
440                 return (0);
441         }

443         for (i = mmu_exported_page_sizes - 1; i > 0; i--) {
444                 pgsz = page_get_pagesize(i);
445                 if (pgsz > max_lpsize) {
446                         continue;
447                 }
448                 raddr = (caddr_t)P2ROUNDUP((uintptr_t)addr, pgsz);
449                 readdr = (caddr_t)P2ALIGN((uintptr_t)eaddr, pgsz);
450                 if (raddr < addr || raddr >= readdr) {
451                         continue;
452                 }
453                 if (P2PHASE((uintptr_t)addr ^ off, pgsz)) {
454                         continue;
```



```
455                 }
456                 /*
457                  * Set szcvec to the remaining page sizes.
458                  */
459                 szcvec = ((1 << (i + 1)) - 1) & ~1;
460                 break;
461         }
462         return (szcvec);
463 }

465 /*
466  * Return a bit vector of large page size codes that
467  * can be used to map [addr, addr + len) region.
468  */
469 /*ARGSUSED*/
470 uint_t
471 map_pgszcvec(caddr_t addr, size_t size, uintptr_t off, int flags, int type,
472     int memcntl)
473 {
474         size_t max_lpsize = mcntl0_lpsize;

476         if (mmu.max_page_level == 0)
477                 return (0);

479         if (flags & MAP_TEXT) {
480                 if (!memcntl)
481                         max_lpsize = max_utext_lpsize;
482                 return (map_szcvec(addr, size, off, max_lpsize,
483                     shm_lpg_min_physmem));

485         } else if (flags & MAP_INITDATA) {
486                 if (!memcntl)
487                         max_lpsize = max_uidata_lpsize;
488                 return (map_szcvec(addr, size, off, max_lpsize,
489                     privm_lpg_min_physmem));

491         } else if (type == MAPPGSZC_SHM) {
492                 if (!memcntl)
493                         max_lpsize = max_shm_lpsize;
494                 return (map_szcvec(addr, size, off, max_lpsize,
495                     shm_lpg_min_physmem));

497         } else if (type == MAPPGSZC_HEAP) {
498                 if (!memcntl)
499                         max_lpsize = max_uheap_lpsize;
500                 return (map_szcvec(addr, size, off, max_lpsize,
501                     privm_lpg_min_physmem));

503         } else if (type == MAPPGSZC_STACK) {
504                 if (!memcntl)
505                         max_lpsize = max_ustack_lpsize;
506                 return (map_szcvec(addr, size, off, max_lpsize,
507                     privm_lpg_min_physmem));

509         } else {
510                 if (!memcntl)
511                         max_lpsize = max_privmap_lpsize;
512                 return (map_szcvec(addr, size, off, max_lpsize,
513                     privm_lpg_min_physmem));
514         }
515 }

517 /*
518  * Handle a pagefault.
519  */
520 faultcode_t
```

```
pagefault(
        caddr_t addr,
        enum fault_type type,
        enum seg_rw rw,
        int iskernel)
{
        struct as *as;
        struct hat *hat;
        struct proc *p;
        kthread_t *t;
        faultcode_t res;
        caddr_t base;
        size_t len;
        int err;
        int mapped_red;
        uintptr_t ea;

        ASSERT_STACK_ALIGNED();

        if (INVALID_VADDR(addr))
                return (FC_NOMAP);

        mapped_red = segkp_map_red();

        if (iskernel) {
                as = &kas;
                hat = as->a_hat;
        } else {
                t = curthread;
                p = ttoproc(t);
                as = p->p_as;
                hat = as->a_hat;
        }

        /*
         * Dispatch pagefault.
         */
        res = as_fault(hat, as, addr, 1, type, rw);

        /*
         * If this isn't a potential unmapped hole in the user's
         * UNIX data or stack segments, just return status info.
         */
        if (res != FC_NOMAP || iskernel)
                goto out;

        /*
         * Check to see if we happened to faulted on a currently unmapped
         * part of the UNIX data or stack segments.  If so, create a zfod
         * mapping there and then try calling the fault routine again.
         */
        base = p->p_brkbase;
        len = p->p_brksize;

        if (addr < base || addr >= base + len) {                /* data seg? */
                base = (caddr_t)p->p_usrstack - p->p_stksize;
                len = p->p_stksize;
                if (addr < base || addr >= p->p_usrstack) {     /* stack seg? */
                        /* not in either UNIX data or stack segments */
                        res = FC_NOMAP;
                        goto out;
                }
        }

        /*
         * the rest of this function implements a 3.X 4.X 5.X compatibility
```



```
         * This code is probably not needed anymore
         */
        if (p->p_model == DATAMODEL_ILP32) {

                /* expand the gap to the page boundaries on each side */
                ea = P2ROUNDUP((uintptr_t)base + len, MMU_PAGESIZE);
                base = (caddr_t)P2ALIGN((uintptr_t)base, MMU_PAGESIZE);
                len = ea - (uintptr_t)base;

                as_rangelock(as);
                if (as_gap(as, MMU_PAGESIZE, &base, &len, AH_CONTAIN, addr) ==
                    0) {
                        err = as_map(as, base, len, segvn_create, zfod_argsp);
                        as_rangeunlock(as);
                        if (err) {
                                res = FC_MAKE_ERR(err);
                                goto out;
                        }
                } else {
                        /*
                         * This page is already mapped by another thread after
                         * we returned from as_fault() above.  We just fall
                         * through as_fault() below.
                         */
                        as_rangeunlock(as);
                }

                res = as_fault(hat, as, addr, 1, F_INVAL, rw);
        }

out:
        if (mapped_red)
                segkp_unmap_red();

        return (res);
}

void
map_addr(caddr_t *addrp, size_t len, offset_t off, int vacalign, uint_t flags)
{
        struct proc *p = curproc;
        caddr_t userlimit = (flags & _MAP_LOW32) ?
            (caddr_t)_userlimit32 : p->p_as->a_userlimit;

        map_addr_proc(addrp, len, off, vacalign, userlimit, curproc, flags);
}

/*ARGSUSED*/
int
map_addr_vacalign_check(caddr_t addr, u_offset_t off)
{
        return (0);
}

/*
 * The maximum amount a randomized mapping will be slewed.  We should perhaps
 * arrange things so these tunables can be separate for mmap, mmapobj, and
 * ld.so
 */
volatile size_t aslr_max_map_skew = 256 * 1024 * 1024; /* 256MB */

/*
#endif /* ! codereview */
 * map_addr_proc() is the routine called when the system is to
 * choose an address for the user.  We will pick an address
 * range which is the highest available below userlimit.
```

```
 *
 * Every mapping will have a redzone of a single page on either side of
 * the request. This is done to leave one page unmapped between segments.
 * This is not required, but it's useful for the user because if their
 * program strays across a segment boundary, it will catch a fault
 * immediately making debugging a little easier.  Currently the redzone
 * is mandatory.
 *
 * addrp is a value/result parameter.
 *      On input it is a hint from the user to be used in a completely
 *      machine dependent fashion.  We decide to completely ignore this hint.
 *      If MAP_ALIGN was specified, addrp contains the minimal alignment, which
 *      must be some "power of two" multiple of pagesize.
 *
 *      On output it is NULL if no address can be found in the current
 *      processes address space or else an address that is currently
 *      not mapped for len bytes with a page of red zone on either side.
 *
 *      vacalign is not needed on x86 (it's for viturally addressed caches)
 */
/*ARGSUSED*/
void
map_addr_proc(
        caddr_t *addrp,
        size_t len,
        offset_t off,
        int vacalign,
        caddr_t userlimit,
        struct proc *p,
        uint_t flags)
{
        struct as *as = p->p_as;
        caddr_t addr;
        caddr_t base;
        size_t slen;
        size_t align_amount;

        ASSERT32(userlimit == as->a_userlimit);

        base = p->p_brkbase;
#if defined(__amd64)
        /*
         * XX64 Yes, this needs more work.
         */
        if (p->p_model == DATAMODEL_NATIVE) {
                if (userlimit < as->a_userlimit) {
                        /*
                         * This happens when a program wants to map
                         * something in a range that's accessible to a
                         * program in a smaller address space.  For example,
                         * a 64-bit program calling mmap32(2) to guarantee
                         * that the returned address is below 4Gbytes.
                         */
                        ASSERT((uintptr_t)userlimit < ADDRESS_C(0xffffffff));

                        if (userlimit > base)
                                slen = userlimit - base;
                        else {
                                *addrp = NULL;
                                return;
                        }
                } else {
                        /*
                         * XX64 This layout is probably wrong .. but in
                         * the event we make the amd64 address space look
                         * like sparcv9 i.e. with the stack -above- the
```



```
                         * heap, this bit of code might even be correct.
                         */
                        slen = p->p_usrstack - base -
                            ((p->p_stk_ctl + PAGEOFFSET) & PAGEMASK);
                }
        } else
#endif
                slen = userlimit - base;

        /* Make len be a multiple of PAGESIZE */
        len = (len + PAGEOFFSET) & PAGEMASK;

        /*
         * figure out what the alignment should be
         *
         * XX64 -- is there an ELF_AMD64_MAXPGSZ or is it the same????
         */
        if (len <= ELF_386_MAXPGSZ) {
                /*
                 * Align virtual addresses to ensure that ELF shared libraries
                 * are mapped with the appropriate alignment constraints by
                 * the run-time linker.
                 */
                align_amount = ELF_386_MAXPGSZ;
        } else {
                /*
                 * For 32-bit processes, only those which have specified
                 * MAP_ALIGN and an addr will be aligned on a larger page size.
                 * Not doing so can potentially waste up to 1G of process
                 * address space.
                 */
                int lvl = (p->p_model == DATAMODEL_ILP32) ? 1 :
                    mmu.umax_page_level;

                while (lvl && len < LEVEL_SIZE(lvl))
                        --lvl;

                align_amount = LEVEL_SIZE(lvl);
        }
        if ((flags & MAP_ALIGN) && ((uintptr_t)*addrp > align_amount))
                align_amount = (uintptr_t)*addrp;

        ASSERT(ISP2(align_amount));
        ASSERT(align_amount == 0 || align_amount >= PAGESIZE);

        off = off & (align_amount - 1);

#endif /* ! codereview */
        /*
         * Look for a large enough hole starting below userlimit.
         * After finding it, use the upper part.
         */
        if (as_gap_aligned(as, len, &base, &slen, AH_HI, NULL, align_amount,
            PAGESIZE, off) == 0) {
                caddr_t as_addr;

                /*
                 * addr is the highest possible address to use since we have
                 * a PAGESIZE redzone at the beginning and end.
                 */
                addr = base + slen - (PAGESIZE + len);
                as_addr = addr;
                /*
                 * Round address DOWN to the alignment amount and
                 * add the offset in.
                 * If addr is greater than as_addr, len would not be large
```

```
785                  * enough to include the redzone, so we must adjust down
786                  * by the alignment amount.
787                  */
788                 addr = (caddr_t)((uintptr_t)addr & (~(align_amount - 1)));
789                 addr += (uintptr_t)off;
790                 if (addr > as_addr) {
791                         addr -= align_amount;
792                 }
793
794                 /*
795                  * If randomization is requested, slew the allocation
796                  * backwards, within the same gap, by a random amount.
797                  *
798                  * XXX: This will fall over in processes like Java, which
799                  * commonly have a great many small mappings.
800                  */
801                 if (flags & _MAP_RANDOMIZE) {
802                         uint32_t slew;
803
804                         (void) random_get_pseudo_bytes((uint8_t *)&slew,
805                             sizeof (slew));
806
807                         slew = slew % MIN(aslr_max_map_skew, (addr - base));
808                         addr -= P2ALIGN(slew, align_amount);
809                 }
810
811 #endif /* ! codereview */
812                 ASSERT(addr > base);
813                 ASSERT(addr + len < base + slen);
814                 ASSERT(((uintptr_t)addr & (align_amount - 1)) ==
815                     ((uintptr_t)(off)));
816                 *addrp = addr;
817         } else {
818                 *addrp = NULL;  /* no more virtual space */
819         }
820 }
821
822 int valid_va_range_aligned_wraparound;
823
824 /*
825  * Determine whether [*basep, *basep + *lenp) contains a mappable range of
826  * addresses at least "minlen" long, where the base of the range is at "off"
827  * phase from an "align" boundary and there is space for a "redzone"-sized
828  * redzone on either side of the range.  On success, 1 is returned and *basep
829  * and *lenp are adjusted to describe the acceptable range (including
830  * the redzone).  On failure, 0 is returned.
831  */
832 /*ARGSUSED3*/
833 int
834 valid_va_range_aligned(caddr_t *basep, size_t *lenp, size_t minlen, int dir,
835     size_t align, size_t redzone, size_t off)
836 {
837         uintptr_t hi, lo;
838         size_t tot_len;
839
840         ASSERT(align == 0 ? off == 0 : off < align);
841         ASSERT(ISP2(align));
842         ASSERT(align == 0 || align >= PAGESIZE);
843
844         lo = (uintptr_t)*basep;
845         hi = lo + *lenp;
846         tot_len = minlen + 2 * redzone; /* need at least this much space */
847
848         /*
849          * If hi rolled over the top, try cutting back.
850          */
```



```
851         if (hi < lo) {
852                 *lenp = 0UL - lo - 1UL;
853                 /* See if this really happens. If so, then we figure out why */
854                 valid_va_range_aligned_wraparound++;
855                 hi = lo + *lenp;
856         }
857         if (*lenp < tot_len) {
858                 return (0);
859         }
860
861 #if defined(__amd64)
862         /*
863          * Deal with a possible hole in the address range between
864          * hole_start and hole_end that should never be mapped.
865          */
866         if (lo < hole_start) {
867                 if (hi > hole_start) {
868                         if (hi < hole_end) {
869                                 hi = hole_start;
870                         } else {
871                                 /* lo < hole_start && hi >= hole_end */
872                                 if (dir == AH_LO) {
873                                         /*
874                                          * prefer lowest range
875                                          */
876                                         if (hole_start - lo >= tot_len)
877                                                 hi = hole_start;
878                                         else if (hi - hole_end >= tot_len)
879                                                 lo = hole_end;
880                                         else
881                                                 return (0);
882                                 } else {
883                                         /*
884                                          * prefer highest range
885                                          */
886                                         if (hi - hole_end >= tot_len)
887                                                 lo = hole_end;
888                                         else if (hole_start - lo >= tot_len)
889                                                 hi = hole_start;
890                                         else
891                                                 return (0);
892                                 }
893                         }
894                 }
895         } else {
896                 /* lo >= hole_start */
897                 if (hi < hole_end)
898                         return (0);
899                 if (lo < hole_end)
900                         lo = hole_end;
901         }
902 #endif
903
904         if (hi - lo < tot_len)
905                 return (0);
906
907         if (align > 1) {
908                 uintptr_t tlo = lo + redzone;
909                 uintptr_t thi = hi - redzone;
910                 tlo = (uintptr_t)P2PHASEUP(tlo, align, off);
911                 if (tlo < lo + redzone) {
912                         return (0);
913                 }
914                 if (thi < tlo || thi - tlo < minlen) {
915                         return (0);
916                 }
```

```
917         }

919         *basep = (caddr_t)lo;
920         *lenp = hi - lo;
921         return (1);
922 }

924 /*
925  * Determine whether [*basep, *basep + *lenp) contains a mappable range of
926  * addresses at least "minlen" long.  On success, 1 is returned and *basep
927  * and *lenp are adjusted to describe the acceptable range.  On failure, 0
928  * is returned.
929  */
930 int
931 valid_va_range(caddr_t *basep, size_t *lenp, size_t minlen, int dir)
932 {
933         return (valid_va_range_aligned(basep, lenp, minlen, dir, 0, 0, 0));
934 }

936 /*
937  * Determine whether [addr, addr+len] are valid user addresses.
938  */
939 /*ARGSUSED*/
940 int
941 valid_usr_range(caddr_t addr, size_t len, uint_t prot, struct as *as,
942     caddr_t userlimit)
943 {
944         caddr_t eaddr = addr + len;

946         if (eaddr <= addr || addr >= userlimit || eaddr > userlimit)
947                 return (RANGE_BADADDR);

949 #if defined(__amd64)
950         /*
951          * Check for the VA hole
952          */
953         if (eaddr > (caddr_t)hole_start && addr < (caddr_t)hole_end)
954                 return (RANGE_BADADDR);
955 #endif

957         return (RANGE_OKAY);
958 }

960 /*
961  * Return 1 if the page frame is onboard memory, else 0.
962  */
963 int
964 pf_is_memory(pfn_t pf)
965 {
966         if (pfn_is_foreign(pf))
967                 return (0);
968         return (address_in_memlist(phys_install, pfn_to_pa(pf), 1));
969 }

971 /*
972  * return the memrange containing pfn
973  */
974 int
975 memrange_num(pfn_t pfn)
976 {
977         int n;

979         for (n = 0; n < nranges - 1; ++n) {
980                 if (pfn >= memranges[n])
981                         break;
982         }
```



```
983         return (n);
984 }

986 /*
987  * return the mnoderange containing pfn
988  */
989 /*ARGSUSED*/
990 int
991 pfn_2_mtype(pfn_t pfn)
992 {
993 #if defined(__xpv)
994         return (0);
995 #else
996         int     n;

998         /* Always start from highest pfn and work our way down */
999         for (n = mtypetop; n != -1; n = mnoderanges[n].mnr_next) {
1000                 if (pfn >= mnoderanges[n].mnr_pfnlo) {
1001                         break;
1002                 }
1003         }
1004         return (n);
1005 #endif
1006 }

1008 #if !defined(__xpv)
1009 /*
1010  * is_contigpage_free:
1011  *      returns a page list of contiguous pages. It minimally has to return
1012  *      minctg pages. Caller determines minctg based on the scatter-gather
1013  *      list length.
1014  *
1015  *      pfnp is set to the next page frame to search on return.
1016  */
1017 static page_t *
1018 is_contigpage_free(
1019         pfn_t *pfnp,
1020         pgcnt_t *pgcnt,
1021         pgcnt_t minctg,
1022         uint64_t pfnseg,
1023         int iolock)
1024 {
1025         int     i = 0;
1026         pfn_t   pfn = *pfnp;
1027         page_t  *pp;
1028         page_t  *plist = NULL;

1030         /*
1031          * fail if pfn + minctg crosses a segment boundary.
1032          * Adjust for next starting pfn to begin at segment boundary.
1033          */

1035         if (((*pfnp + minctg - 1) & pfnseg) < (*pfnp & pfnseg)) {
1036                 *pfnp = roundup(*pfnp, pfnseg + 1);
1037                 return (NULL);
1038         }

1040         do {
1041 retry:
1042                 pp = page_numtopp_nolock(pfn + i);
1043                 if ((pp == NULL) || IS_DUMP_PAGE(pp) ||
1044                     (page_trylock(pp, SE_EXCL) == 0)) {
1045                         (*pfnp)++;
1046                         break;
1047                 }
1048                 if (page_pptonum(pp) != pfn + i) {
```

```
1049                            page_unlock(pp);
1050                            goto retry;
1051                    }

1053                    if (!(PP_ISFREE(pp))) {
1054                            page_unlock(pp);
1055                            (*pfnp)++;
1056                            break;
1057                    }

1059                    if (!PP_ISAGED(pp)) {
1060                            page_list_sub(pp, PG_CACHE_LIST);
1061                            page_hashout(pp, (kmutex_t *)NULL);
1062                    } else {
1063                            page_list_sub(pp, PG_FREE_LIST);
1064                    }

1066                    if (iolock)
1067                            page_io_lock(pp);
1068                    page_list_concat(&plist, &pp);

1070                    /*
1071                     * exit loop when pgcnt satisfied or segment boundary reached.
1072                     */

1074            } while ((++i < *pgcnt) && ((pfn + i) & pfnseg));

1076            *pfnp += i;             /* set to next pfn to search */

1078            if (i >= minctg) {
1079                    *pgcnt -= i;
1080                    return (plist);
1081            }

1083            /*
1084             * failure: minctg not satisfied.
1085             *
1086             * if next request crosses segment boundary, set next pfn
1087             * to search from the segment boundary.
1088             */
1089            if (((*pfnp + minctg - 1) & pfnseg) < (*pfnp & pfnseg))
1090                    *pfnp = roundup(*pfnp, pfnseg + 1);

1092            /* clean up any pages already allocated */

1094            while (plist) {
1095                    pp = plist;
1096                    page_sub(&plist, pp);
1097                    page_list_add(pp, PG_FREE_LIST | PG_LIST_TAIL);
1098                    if (iolock)
1099                            page_io_unlock(pp);
1100                    page_unlock(pp);
1101            }

1103            return (NULL);
1104    }
1105    #endif  /* !__xpv */

1107    /*
1108     * verify that pages being returned from allocator have correct DMA attribute
1109     */
1110    #ifndef DEBUG
1111    #define check_dma(a, b, c) (void)(0)
1112    #else
1113    static void
1114    check_dma(ddi_dma_attr_t *dma_attr, page_t *pp, int cnt)
```



```
1115    {
1116            if (dma_attr == NULL)
1117                    return;

1119            while (cnt-- > 0) {
1120                    if (pa_to_ma(pfn_to_pa(pp->p_pagenum)) <
1121                        dma_attr->dma_attr_addr_lo)
1122                            panic("PFN (pp=%p) below dma_attr_addr_lo", (void *)pp);
1123                    if (pa_to_ma(pfn_to_pa(pp->p_pagenum)) >=
1124                        dma_attr->dma_attr_addr_hi)
1125                            panic("PFN (pp=%p) above dma_attr_addr_hi", (void *)pp);
1126                    pp = pp->p_next;
1127            }
1128    }
1129    #endif

1131    #if !defined(__xpv)
1132    static page_t *
1133    page_get_contigpage(pgcnt_t *pgcnt, ddi_dma_attr_t *mattr, int iolock)
1134    {
1135            pfn_t           pfn;
1136            int             sgllen;
1137            uint64_t        pfnseg;
1138            pgcnt_t         minctg;
1139            page_t          *pplist = NULL, *plist;
1140            uint64_t        lo, hi;
1141            pgcnt_t         pfnalign = 0;
1142            static pfn_t    startpfn;
1143            static pgcnt_t  lastctgcnt;
1144            uintptr_t       align;

1146            CONTIG_LOCK();

1148            if (mattr) {
1149                    lo = mmu_btop((mattr->dma_attr_addr_lo + MMU_PAGEOFFSET));
1150                    hi = mmu_btop(mattr->dma_attr_addr_hi);
1151                    if (hi >= physmax)
1152                            hi = physmax - 1;
1153                    sgllen = mattr->dma_attr_sgllen;
1154                    pfnseg = mmu_btop(mattr->dma_attr_seg);

1156                    align = maxbit(mattr->dma_attr_align, mattr->dma_attr_minxfer);
1157                    if (align > MMU_PAGESIZE)
1158                            pfnalign = mmu_btop(align);

1160                    /*
1161                     * in order to satisfy the request, must minimally
1162                     * acquire minctg contiguous pages
1163                     */
1164                    minctg = howmany(*pgcnt, sgllen);

1166                    ASSERT(hi >= lo);

1168                    /*
1169                     * start from where last searched if the minctg >= lastctgcnt
1170                     */
1171                    if (minctg < lastctgcnt || startpfn < lo || startpfn > hi)
1172                            startpfn = lo;
1173            } else {
1174                    hi = physmax - 1;
1175                    lo = 0;
1176                    sgllen = 1;
1177                    pfnseg = mmu.highest_pfn;
1178                    minctg = *pgcnt;

1180                    if (minctg < lastctgcnt)
```

```
1181                        startpfn = lo;
1182        }
1183        lastctgcnt = minctg;

1185        ASSERT(pfnseg + 1 >= (uint64_t)minctg);

1187        /* conserve 16m memory - start search above 16m when possible */
1188        if (hi > PFN_16M && startpfn < PFN_16M)
1189                startpfn = PFN_16M;

1191        pfn = startpfn;
1192        if (pfnalign)
1193                pfn = P2ROUNDUP(pfn, pfnalign);

1195        while (pfn + minctg - 1 <= hi) {

1197                plist = is_contigpage_free(&pfn, pgcnt, minctg, pfnseg, iolock);
1198                if (plist) {
1199                        page_list_concat(&pplist, &plist);
1200                        sgllen--;
1201                        /*
1202                         * return when contig pages no longer needed
1203                         */
1204                        if (!*pgcnt || ((*pgcnt <= sgllen) && !pfnalign)) {
1205                                startpfn = pfn;
1206                                CONTIG_UNLOCK();
1207                                check_dma(mattr, pplist, *pgcnt);
1208                                return (pplist);
1209                        }
1210                        minctg = howmany(*pgcnt, sgllen);
1211                }
1212                if (pfnalign)
1213                        pfn = P2ROUNDUP(pfn, pfnalign);
1214        }

1216        /* cannot find contig pages in specified range */
1217        if (startpfn == lo) {
1218                CONTIG_UNLOCK();
1219                return (NULL);
1220        }

1222        /* did not start with lo previously */
1223        pfn = lo;
1224        if (pfnalign)
1225                pfn = P2ROUNDUP(pfn, pfnalign);

1227        /* allow search to go above startpfn */
1228        while (pfn < startpfn) {

1230                plist = is_contigpage_free(&pfn, pgcnt, minctg, pfnseg, iolock);
1231                if (plist != NULL) {

1233                        page_list_concat(&pplist, &plist);
1234                        sgllen--;

1236                        /*
1237                         * return when contig pages no longer needed
1238                         */
1239                        if (!*pgcnt || ((*pgcnt <= sgllen) && !pfnalign)) {
1240                                startpfn = pfn;
1241                                CONTIG_UNLOCK();
1242                                check_dma(mattr, pplist, *pgcnt);
1243                                return (pplist);
1244                        }
1245                        minctg = howmany(*pgcnt, sgllen);
1246                }
```



```
1247                if (pfnalign)
1248                        pfn = P2ROUNDUP(pfn, pfnalign);
1249        }
1250        CONTIG_UNLOCK();
1251        return (NULL);
1252 }
1253 #endif  /* !__xpv */

1255 /*
1256  * mnode_range_cnt() calculates the number of memory ranges for mnode and
1257  * memranges[]. Used to determine the size of page lists and mnoderanges.
1258  */
1259 int
1260 mnode_range_cnt(int mnode)
1261 {
1262 #if defined(__xpv)
1263        ASSERT(mnode == 0);
1264        return (1);
1265 #else   /* __xpv */
1266        int     mri;
1267        int     mnrcnt = 0;

1269        if (mem_node_config[mnode].exists != 0) {
1270                mri = nranges - 1;

1272                /* find the memranges index below contained in mnode range */

1274                while (MEMRANGEHI(mri) < mem_node_config[mnode].physbase)
1275                        mri--;

1277                /*
1278                 * increment mnode range counter when memranges or mnode
1279                 * boundary is reached.
1280                 */
1281                while (mri >= 0 &&
1282                    mem_node_config[mnode].physmax >= MEMRANGELO(mri)) {
1283                        mnrcnt++;
1284                        if (mem_node_config[mnode].physmax > MEMRANGEHI(mri))
1285                                mri--;
1286                        else
1287                                break;
1288                }
1289        }
1290        ASSERT(mnrcnt <= MAX_MNODE_MRANGES);
1291        return (mnrcnt);
1292 #endif  /* __xpv */
1293 }

1295 /*
1296  * mnode_range_setup() initializes mnoderanges.
1297  */
1298 void
1299 mnode_range_setup(mnoderange_t *mnoderanges)
1300 {
1301        mnoderange_t *mp = mnoderanges;
1302        int     mnode, mri;
1303        int     mindex = 0;     /* current index into mnoderanges array */
1304        int     i, j;
1305        pfn_t   hipfn;
1306        int     last, hi;

1308        for (mnode = 0; mnode < max_mem_nodes; mnode++) {
1309                if (mem_node_config[mnode].exists == 0)
1310                        continue;

1312                mri = nranges - 1;
```

```
1314                while (MEMRANGEHI(mri) < mem_node_config[mnode].physbase)
1315                        mri--;
1316
1317                while (mri >= 0 && mem_node_config[mnode].physmax >=
1318                    MEMRANGELO(mri)) {
1319                        mnoderanges->mnr_pfnlo = MAX(MEMRANGELO(mri),
1320                            mem_node_config[mnode].physbase);
1321                        mnoderanges->mnr_pfnhi = MIN(MEMRANGEHI(mri),
1322                            mem_node_config[mnode].physmax);
1323                        mnoderanges->mnr_mnode = mnode;
1324                        mnoderanges->mnr_memrange = mri;
1325                        mnoderanges->mnr_exists = 1;
1326                        mnoderanges++;
1327                        mindex++;
1328                        if (mem_node_config[mnode].physmax > MEMRANGEHI(mri))
1329                                mri--;
1330                        else
1331                                break;
1332                }
1333        }
1334
1335        /*
1336         * For now do a simple sort of the mnoderanges array to fill in
1337         * the mnr_next fields.  Since mindex is expected to be relatively
1338         * small, using a simple O(N^2) algorithm.
1339         */
1340        for (i = 0; i < mindex; i++) {
1341                if (mp[i].mnr_pfnlo == 0)       /* find lowest */
1342                        break;
1343        }
1344        ASSERT(i < mindex);
1345        last = i;
1346        mtype16m = last;
1347        mp[last].mnr_next = -1;
1348        for (i = 0; i < mindex - 1; i++) {
1349                hipfn = (pfn_t)(-1);
1350                hi = -1;
1351                /* find next highest mnode range */
1352                for (j = 0; j < mindex; j++) {
1353                        if (mp[j].mnr_pfnlo > mp[last].mnr_pfnlo &&
1354                            mp[j].mnr_pfnlo < hipfn) {
1355                                hipfn = mp[j].mnr_pfnlo;
1356                                hi = j;
1357                        }
1358                }
1359                mp[hi].mnr_next = last;
1360                last = hi;
1361        }
1362        mtypetop = last;
1363 }
1364
1365 #ifndef __xpv
1366 /*
1367  * Update mnoderanges for memory hot-add DR operations.
1368  */
1369 static void
1370 mnode_range_add(int mnode)
1371 {
1372        int     *prev;
1373        int     n, mri;
1374        pfn_t   start, end;
1375        extern  void membar_sync(void);
1376
1377        ASSERT(0 <= mnode && mnode < max_mem_nodes);
1378        ASSERT(mem_node_config[mnode].exists);
```



```
1379        start = mem_node_config[mnode].physbase;
1380        end = mem_node_config[mnode].physmax;
1381        ASSERT(start <= end);
1382        mutex_enter(&mnoderange_lock);
1383
1384 #ifdef  DEBUG
1385        /* Check whether it interleaves with other memory nodes. */
1386        for (n = mtypetop; n != -1; n = mnoderanges[n].mnr_next) {
1387                ASSERT(mnoderanges[n].mnr_exists);
1388                if (mnoderanges[n].mnr_mnode == mnode)
1389                        continue;
1390                ASSERT(start > mnoderanges[n].mnr_pfnhi ||
1391                    end < mnoderanges[n].mnr_pfnlo);
1392        }
1393 #endif  /* DEBUG */
1394
1395        mri = nranges - 1;
1396        while (MEMRANGEHI(mri) < mem_node_config[mnode].physbase)
1397                mri--;
1398        while (mri >= 0 && mem_node_config[mnode].physmax >= MEMRANGELO(mri)) {
1399                /* Check whether mtype already exists. */
1400                for (n = mtypetop; n != -1; n = mnoderanges[n].mnr_next) {
1401                        if (mnoderanges[n].mnr_mnode == mnode &&
1402                            mnoderanges[n].mnr_memrange == mri) {
1403                                mnoderanges[n].mnr_pfnlo = MAX(MEMRANGELO(mri),
1404                                    start);
1405                                mnoderanges[n].mnr_pfnhi = MIN(MEMRANGEHI(mri),
1406                                    end);
1407                                break;
1408                        }
1409                }
1410
1411                /* Add a new entry if it doesn't exist yet. */
1412                if (n == -1) {
1413                        /* Try to find an unused entry in mnoderanges array. */
1414                        for (n = 0; n < mnoderangecnt; n++) {
1415                                if (mnoderanges[n].mnr_exists == 0)
1416                                        break;
1417                        }
1418                        ASSERT(n < mnoderangecnt);
1419                        mnoderanges[n].mnr_pfnlo = MAX(MEMRANGELO(mri), start);
1420                        mnoderanges[n].mnr_pfnhi = MIN(MEMRANGEHI(mri), end);
1421                        mnoderanges[n].mnr_mnode = mnode;
1422                        mnoderanges[n].mnr_memrange = mri;
1423                        mnoderanges[n].mnr_exists = 1;
1424                        /* Page 0 should always be present. */
1425                        for (prev = &mtypetop;
1426                            mnoderanges[*prev].mnr_pfnlo > start;
1427                            prev = &mnoderanges[*prev].mnr_next) {
1428                                ASSERT(mnoderanges[*prev].mnr_next >= 0);
1429                                ASSERT(mnoderanges[*prev].mnr_pfnlo > end);
1430                        }
1431                        mnoderanges[n].mnr_next = *prev;
1432                        membar_sync();
1433                        *prev = n;
1434                }
1435
1436                if (mem_node_config[mnode].physmax > MEMRANGEHI(mri))
1437                        mri--;
1438                else
1439                        break;
1440        }
1441
1442        mutex_exit(&mnoderange_lock);
1443 }
```

```
1445 /*
1446  * Update mnoderanges for memory hot-removal DR operations.
1447  */
1448 static void
1449 mnode_range_del(int mnode)
1450 {
1451         _NOTE(ARGUNUSED(mnode));
1452         ASSERT(0 <= mnode && mnode < max_mem_nodes);
1453         /* TODO: support deletion operation. */
1454         ASSERT(0);
1455 }

1457 void
1458 plat_slice_add(pfn_t start, pfn_t end)
1459 {
1460         mem_node_add_slice(start, end);
1461         if (plat_dr_enabled()) {
1462                 mnode_range_add(PFN_2_MEM_NODE(start));
1463         }
1464 }

1466 void
1467 plat_slice_del(pfn_t start, pfn_t end)
1468 {
1469         ASSERT(PFN_2_MEM_NODE(start) == PFN_2_MEM_NODE(end));
1470         ASSERT(plat_dr_enabled());
1471         mnode_range_del(PFN_2_MEM_NODE(start));
1472         mem_node_del_slice(start, end);
1473 }
1474 #endif  /* __xpv */

1476 /*ARGSUSED*/
1477 int
1478 mtype_init(vnode_t *vp, caddr_t vaddr, uint_t *flags, size_t pgsz)
1479 {
1480         int mtype = mtypetop;

1482 #if !defined(__xpv)
1483 #if defined(__i386)
1484         /*
1485          * set the mtype range
1486          * - kmem requests need to be below 4g if restricted_kmemalloc is set.
1487          * - for non kmem requests, set range to above 4g if memory below 4g
1488          * runs low.
1489          */
1490         if (restricted_kmemalloc && VN_ISKAS(vp) &&
1491             (caddr_t)(vaddr) >= kernelheap &&
1492             (caddr_t)(vaddr) < ekernelheap) {
1493                 ASSERT(physmax4g);
1494                 mtype = mtype4g;
1495                 if (RESTRICT16M_ALLOC(freemem4g - btop(pgsz),
1496                     btop(pgsz), *flags)) {
1497                         *flags |= PGI_MT_RANGE16M;
1498                 } else {
1499                         VM_STAT_ADD(vmm_vmstats.unrestrict16mcnt);
1500                         VM_STAT_COND_ADD((*flags & PG_PANIC),
1501                             vmm_vmstats.pgpanicalloc);
1502                         *flags |= PGI_MT_RANGE0;
1503                 }
1504                 return (mtype);
1505         }
1506 #endif  /* __i386 */

1508         if (RESTRICT4G_ALLOC) {
1509                 VM_STAT_ADD(vmm_vmstats.restrict4gcnt);
1510                 /* here only for > 4g systems */
```



```
1511                 *flags |= PGI_MT_RANGE4G;
1512         } else if (RESTRICT16M_ALLOC(freemem, btop(pgsz), *flags)) {
1513                 *flags |= PGI_MT_RANGE16M;
1514         } else {
1515                 VM_STAT_ADD(vmm_vmstats.unrestrict16mcnt);
1516                 VM_STAT_COND_ADD((*flags & PG_PANIC), vmm_vmstats.pgpanicalloc);
1517                 *flags |= PGI_MT_RANGE0;
1518         }
1519 #endif /* !__xpv */
1520         return (mtype);
1521 }


1524 /* mtype init for page_get_replacement_page */
1525 /*ARGSUSED*/
1526 int
1527 mtype_pgr_init(int *flags, page_t *pp, int mnode, pgcnt_t pgcnt)
1528 {
1529         int mtype = mtypetop;
1530 #if !defined(__xpv)
1531         if (RESTRICT16M_ALLOC(freemem, pgcnt, *flags)) {
1532                 *flags |= PGI_MT_RANGE16M;
1533         } else {
1534                 VM_STAT_ADD(vmm_vmstats.unrestrict16mcnt);
1535                 *flags |= PGI_MT_RANGE0;
1536         }
1537 #endif
1538         return (mtype);
1539 }

1541 /*
1542  * Determine if the mnode range specified in mtype contains memory belonging
1543  * to memory node mnode.  If flags & PGI_MT_RANGE is set then mtype contains
1544  * the range from high pfn to 0, 16m or 4g.
1545  *
1546  * Return first mnode range type index found otherwise return -1 if none found.
1547  */
1548 int
1549 mtype_func(int mnode, int mtype, uint_t flags)
1550 {
1551         if (flags & PGI_MT_RANGE) {
1552                 int     mnr_lim = MRI_0;

1554                 if (flags & PGI_MT_NEXT) {
1555                         mtype = mnoderanges[mtype].mnr_next;
1556                 }
1557                 if (flags & PGI_MT_RANGE4G)
1558                         mnr_lim = MRI_4G;       /* exclude 0-4g range */
1559                 else if (flags & PGI_MT_RANGE16M)
1560                         mnr_lim = MRI_16M;      /* exclude 0-16m range */
1561                 while (mtype != -1 &&
1562                     mnoderanges[mtype].mnr_memrange <= mnr_lim) {
1563                         if (mnoderanges[mtype].mnr_mnode == mnode)
1564                                 return (mtype);
1565                         mtype = mnoderanges[mtype].mnr_next;
1566                 }
1567         } else if (mnoderanges[mtype].mnr_mnode == mnode) {
1568                 return (mtype);
1569         }
1570         return (-1);
1571 }

1573 /*
1574  * Update the page list max counts with the pfn range specified by the
1575  * input parameters.
1576  */
```

```
1577 void
1578 mtype_modify_max(pfn_t startpfn, long cnt)
1579 {
1580         int             mtype;
1581         pgcnt_t         inc;
1582         spgcnt_t        scnt = (spgcnt_t)(cnt);
1583         pgcnt_t         acnt = ABS(scnt);
1584         pfn_t           endpfn = startpfn + acnt;
1585         pfn_t           pfn, lo;

1587         if (!physmax4g)
1588                 return;

1590         mtype = mtypetop;
1591         for (pfn = endpfn; pfn > startpfn; ) {
1592                 ASSERT(mtype != -1);
1593                 lo = mnoderanges[mtype].mnr_pfnlo;
1594                 if (pfn > lo) {
1595                         if (startpfn >= lo) {
1596                                 inc = pfn - startpfn;
1597                         } else {
1598                                 inc = pfn - lo;
1599                         }
1600                         if (mnoderanges[mtype].mnr_memrange != MRI_4G) {
1601                                 if (scnt > 0)
1602                                         maxmem4g += inc;
1603                                 else
1604                                         maxmem4g -= inc;
1605                         }
1606                         pfn -= inc;
1607                 }
1608                 mtype = mnoderanges[mtype].mnr_next;
1609         }
1610 }

1612 int
1613 mtype_2_mrange(int mtype)
1614 {
1615         return (mnoderanges[mtype].mnr_memrange);
1616 }

1618 void
1619 mnodetype_2_pfn(int mnode, int mtype, pfn_t *pfnlo, pfn_t *pfnhi)
1620 {
1621         _NOTE(ARGUNUSED(mnode));
1622         ASSERT(mnoderanges[mtype].mnr_mnode == mnode);
1623         *pfnlo = mnoderanges[mtype].mnr_pfnlo;
1624         *pfnhi = mnoderanges[mtype].mnr_pfnhi;
1625 }

1627 size_t
1628 plcnt_sz(size_t ctrs_sz)
1629 {
1630 #ifdef DEBUG
1631         int     szc, colors;

1633         ctrs_sz += mnoderangecnt * sizeof (struct mnr_mts) * mmu_page_sizes;
1634         for (szc = 0; szc < mmu_page_sizes; szc++) {
1635                 colors = page_get_pagecolors(szc);
1636                 ctrs_sz += mnoderangecnt * sizeof (pgcnt_t) * colors;
1637         }
1638 #endif
1639         return (ctrs_sz);
1640 }

1642 caddr_t
```



```
1643 plcnt_init(caddr_t addr)
1644 {
1645 #ifdef DEBUG
1646         int     mt, szc, colors;

1648         for (mt = 0; mt < mnoderangecnt; mt++) {
1649                 mnoderanges[mt].mnr_mts = (struct mnr_mts *)addr;
1650                 addr += (sizeof (struct mnr_mts) * mmu_page_sizes);
1651                 for (szc = 0; szc < mmu_page_sizes; szc++) {
1652                         colors = page_get_pagecolors(szc);
1653                         mnoderanges[mt].mnr_mts[szc].mnr_mts_colors = colors;
1654                         mnoderanges[mt].mnr_mts[szc].mnr_mtsc_pgcnt =
1655                             (pgcnt_t *)addr;
1656                         addr += (sizeof (pgcnt_t) * colors);
1657                 }
1658         }
1659 #endif
1660         return (addr);
1661 }

1663 void
1664 plcnt_inc_dec(page_t *pp, int mtype, int szc, long cnt, int flags)
1665 {
1666         _NOTE(ARGUNUSED(pp));
1667 #ifdef DEBUG
1668         int     bin = PP_2_BIN(pp);

1670         atomic_add_long(&mnoderanges[mtype].mnr_mts[szc].mnr_mts_pgcnt, cnt);
1671         atomic_add_long(&mnoderanges[mtype].mnr_mts[szc].mnr_mtsc_pgcnt[bin],
1672             cnt);
1673 #endif
1674         ASSERT(mtype == PP_2_MTYPE(pp));
1675         if (physmax4g && mnoderanges[mtype].mnr_memrange != MRI_4G)
1676                 atomic_add_long(&freemem4g, cnt);
1677         if (flags & PG_CACHE_LIST)
1678                 atomic_add_long(&mnoderanges[mtype].mnr_mt_clpgcnt, cnt);
1679         else
1680                 atomic_add_long(&mnoderanges[mtype].mnr_mt_flpgcnt[szc], cnt);
1681         atomic_add_long(&mnoderanges[mtype].mnr_mt_totcnt, cnt);
1682 }

1684 /*
1685  * Returns the free page count for mnode
1686  */
1687 int
1688 mnode_pgcnt(int mnode)
1689 {
1690         int     mtype = mtypetop;
1691         int     flags = PGI_MT_RANGE0;
1692         pgcnt_t pgcnt = 0;

1694         mtype = mtype_func(mnode, mtype, flags);

1696         while (mtype != -1) {
1697                 pgcnt += MTYPE_FREEMEM(mtype);
1698                 mtype = mtype_func(mnode, mtype, flags | PGI_MT_NEXT);
1699         }
1700         return (pgcnt);
1701 }

1703 /*
1704  * Initialize page coloring variables based on the l2 cache parameters.
1705  * Calculate and return memory needed for page coloring data structures.
1706  */
1707 size_t
1708 page_coloring_init(uint_t l2_sz, int l2_linesz, int l2_assoc)
```

```
{
        _NOTE(ARGUNUSED(l2_linesz));
        size_t  colorsz = 0;
        int     i;
        int     colors;

#if defined(__xpv)
        /*
         * Hypervisor domains currently don't have any concept of NUMA.
         * Hence we'll act like there is only 1 memrange.
         */
        i = memrange_num(1);
#else /* !__xpv */
        /*
         * Reduce the memory ranges lists if we don't have large amounts
         * of memory. This avoids searching known empty free lists.
         * To support memory DR operations, we need to keep memory ranges
         * for possible memory hot-add operations.
         */
        if (plat_dr_physmax > physmax)
                i = memrange_num(plat_dr_physmax);
        else
                i = memrange_num(physmax);
#if defined(__i386)
        if (i > MRI_4G)
                restricted_kmemalloc = 0;
#endif
        /* physmax greater than 4g */
        if (i == MRI_4G)
                physmax4g = 1;
#endif /* !__xpv */
        memranges += i;
        nranges -= i;

        ASSERT(mmu_page_sizes <= MMU_PAGE_SIZES);

        ASSERT(ISP2(l2_linesz));
        ASSERT(l2_sz > MMU_PAGESIZE);

        /* l2_assoc is 0 for fully associative l2 cache */
        if (l2_assoc)
                l2_colors = MAX(1, l2_sz / (l2_assoc * MMU_PAGESIZE));
        else
                l2_colors = 1;

        ASSERT(ISP2(l2_colors));

        /* for scalability, configure at least PAGE_COLORS_MIN color bins */
        page_colors = MAX(l2_colors, PAGE_COLORS_MIN);

        /*
         * cpu_page_colors is non-zero when a page color may be spread across
         * multiple bins.
         */
        if (l2_colors < page_colors)
                cpu_page_colors = l2_colors;

        ASSERT(ISP2(page_colors));

        page_colors_mask = page_colors - 1;

        ASSERT(ISP2(CPUSETSIZE()));
        page_coloring_shift = lowbit(CPUSETSIZE());

        /* initialize number of colors per page size */
        for (i = 0; i <= mmu.max_page_level; i++) {
```



```
                hw_page_array[i].hp_size = LEVEL_SIZE(i);
                hw_page_array[i].hp_shift = LEVEL_SHIFT(i);
                hw_page_array[i].hp_pgcnt = LEVEL_SIZE(i) >> LEVEL_SHIFT(0);
                hw_page_array[i].hp_colors = (page_colors_mask >>
                    (hw_page_array[i].hp_shift - hw_page_array[0].hp_shift))
                    + 1;
                colorequivszc[i] = 0;
        }

        /*
         * The value of cpu_page_colors determines if additional color bins
         * need to be checked for a particular color in the page_get routines.
         */
        if (cpu_page_colors != 0) {

                int a = lowbit(page_colors) - lowbit(cpu_page_colors);
                ASSERT(a > 0);
                ASSERT(a < 16);

                for (i = 0; i <= mmu.max_page_level; i++) {
                        if ((colors = hw_page_array[i].hp_colors) <= 1) {
                                colorequivszc[i] = 0;
                                continue;
                        }
                        while ((colors >> a) == 0)
                                a--;
                        ASSERT(a >= 0);

                        /* higher 4 bits encodes color equiv mask */
                        colorequivszc[i] = (a << 4);
                }
        }

        /* factor in colorequiv to check additional 'equivalent' bins. */
        if (colorequiv > 1) {

                int a = lowbit(colorequiv) - 1;
                if (a > 15)
                        a = 15;

                for (i = 0; i <= mmu.max_page_level; i++) {
                        if ((colors = hw_page_array[i].hp_colors) <= 1) {
                                continue;
                        }
                        while ((colors >> a) == 0)
                                a--;
                        if ((a << 4) > colorequivszc[i]) {
                                colorequivszc[i] = (a << 4);
                        }
                }
        }

        /* size for mnoderanges */
        for (mnoderangecnt = 0, i = 0; i < max_mem_nodes; i++)
                mnoderangecnt += mnode_range_cnt(i);
        if (plat_dr_support_memory()) {
                /*
                 * Reserve enough space for memory DR operations.
                 * Two extra mnoderanges for possbile fragmentations,
                 * one for the 2G boundary and the other for the 4G boundary.
                 * We don't expect a memory board crossing the 16M boundary
                 * for memory hot-add operations on x86 platforms.
                 */
                mnoderangecnt += 2 + max_mem_nodes - lgrp_plat_node_cnt;
        }
        colorsz = mnoderangecnt * sizeof (mnoderange_t);
```

```
1842         /* size for fpc_mutex and cpc_mutex */
1843         colorsz += (2 * max_mem_nodes * sizeof (kmutex_t) * NPC_MUTEX);

1845         /* size of page_freelists */
1846         colorsz += mnoderangecnt * sizeof (page_t ***);
1847         colorsz += mnoderangecnt * mmu_page_sizes * sizeof (page_t **);

1849         for (i = 0; i < mmu_page_sizes; i++) {
1850                 colors = page_get_pagecolors(i);
1851                 colorsz += mnoderangecnt * colors * sizeof (page_t *);
1852         }

1854         /* size of page_cachelists */
1855         colorsz += mnoderangecnt * sizeof (page_t **);
1856         colorsz += mnoderangecnt * page_colors * sizeof (page_t *);

1858         return (colorsz);
1859 }

1861 /*
1862  * Called once at startup to configure page_coloring data structures and
1863  * does the 1st page_free()/page_freelist_add().
1864  */
1865 void
1866 page_coloring_setup(caddr_t pcmemaddr)
1867 {
1868         int     i;
1869         int     j;
1870         int     k;
1871         caddr_t addr;
1872         int     colors;

1874         /*
1875          * do page coloring setup
1876          */
1877         addr = pcmemaddr;

1879         mnoderanges = (mnoderange_t *)addr;
1880         addr += (mnoderangecnt * sizeof (mnoderange_t));

1882         mnode_range_setup(mnoderanges);

1884         if (physmax4g)
1885                 mtype4g = pfn_2_mtype(0xfffff);

1887         for (k = 0; k < NPC_MUTEX; k++) {
1888                 fpc_mutex[k] = (kmutex_t *)addr;
1889                 addr += (max_mem_nodes * sizeof (kmutex_t));
1890         }
1891         for (k = 0; k < NPC_MUTEX; k++) {
1892                 cpc_mutex[k] = (kmutex_t *)addr;
1893                 addr += (max_mem_nodes * sizeof (kmutex_t));
1894         }
1895         page_freelists = (page_t ****)addr;
1896         addr += (mnoderangecnt * sizeof (page_t ***));

1898         page_cachelists = (page_t ***)addr;
1899         addr += (mnoderangecnt * sizeof (page_t **));

1901         for (i = 0; i < mnoderangecnt; i++) {
1902                 page_freelists[i] = (page_t ***)addr;
1903                 addr += (mmu_page_sizes * sizeof (page_t **));

1905                 for (j = 0; j < mmu_page_sizes; j++) {
1906                         colors = page_get_pagecolors(j);
```



```
1907                         page_freelists[i][j] = (page_t **)addr;
1908                         addr += (colors * sizeof (page_t *));
1909                 }
1910                 page_cachelists[i] = (page_t **)addr;
1911                 addr += (page_colors * sizeof (page_t *));
1912         }
1913 }

1915 #if defined(__xpv)
1916 /*
1917  * Give back 10% of the io_pool pages to the free list.
1918  * Don't shrink the pool below some absolute minimum.
1919  */
1920 static void
1921 page_io_pool_shrink()
1922 {
1923         int retcnt;
1924         page_t *pp, *pp_first, *pp_last, **curpool;
1925         mfn_t mfn;
1926         int bothpools = 0;

1928         mutex_enter(&io_pool_lock);
1929         io_pool_shrink_attempts++;      /* should be a kstat? */
1930         retcnt = io_pool_cnt / 10;
1931         if (io_pool_cnt - retcnt < io_pool_cnt_min)
1932                 retcnt = io_pool_cnt - io_pool_cnt_min;
1933         if (retcnt <= 0)
1934                 goto done;
1935         io_pool_shrinks++;      /* should be a kstat? */
1936         curpool = &io_pool_4g;
1937 domore:
1938         /*
1939          * Loop through taking pages from the end of the list
1940          * (highest mfns) till amount to return reached.
1941          */
1942         for (pp = *curpool; pp && retcnt > 0; ) {
1943                 pp_first = pp_last = pp->p_prev;
1944                 if (pp_first == *curpool)
1945                         break;
1946                 retcnt--;
1947                 io_pool_cnt--;
1948                 page_io_pool_sub(curpool, pp_first, pp_last);
1949                 if ((mfn = pfn_to_mfn(pp->p_pagenum)) < start_mfn)
1950                         start_mfn = mfn;
1951                 page_free(pp_first, 1);
1952                 pp = *curpool;
1953         }
1954         if (retcnt != 0 && !bothpools) {
1955                 /*
1956                  * If not enough found in less constrained pool try the
1957                  * more constrained one.
1958                  */
1959                 curpool = &io_pool_16m;
1960                 bothpools = 1;
1961                 goto domore;
1962         }
1963 done:
1964         mutex_exit(&io_pool_lock);
1965 }

1967 #endif  /* __xpv */

1969 uint_t
1970 page_create_update_flags_x86(uint_t flags)
1971 {
1972 #if defined(__xpv)
```

```
1973         /*
1974          * Check this is an urgent allocation and free pages are depleted.
1975          */
1976         if (!(flags & PG_WAIT) && freemem < desfree)
1977                 page_io_pool_shrink();
1978 #else /* !__xpv */
1979         /*
1980          * page_create_get_something may call this because 4g memory may be
1981          * depleted. Set flags to allow for relocation of base page below
1982          * 4g if necessary.
1983          */
1984         if (physmax4g)
1985                 flags |= (PGI_PGCPSZC0 | PGI_PGCPHIPRI);
1986 #endif /* __xpv */
1987         return (flags);
1988 }

1990 /*ARGSUSED*/
1991 int
1992 bp_color(struct buf *bp)
1993 {
1994         return (0);
1995 }

1997 #if defined(__xpv)

1999 /*
2000  * Take pages out of an io_pool
2001  */
2002 static void
2003 page_io_pool_sub(page_t **poolp, page_t *pp_first, page_t *pp_last)
2004 {
2005         if (*poolp == pp_first) {
2006                 *poolp = pp_last->p_next;
2007                 if (*poolp == pp_first)
2008                         *poolp = NULL;
2009         }
2010         pp_first->p_prev->p_next = pp_last->p_next;
2011         pp_last->p_next->p_prev = pp_first->p_prev;
2012         pp_first->p_prev = pp_last;
2013         pp_last->p_next = pp_first;
2014 }

2016 /*
2017  * Put a page on the io_pool list. The list is ordered by increasing MFN.
2018  */
2019 static void
2020 page_io_pool_add(page_t **poolp, page_t *pp)
2021 {
2022         page_t  *look;
2023         mfn_t   mfn = mfn_list[pp->p_pagenum];

2025         if (*poolp == NULL) {
2026                 *poolp = pp;
2027                 pp->p_next = pp;
2028                 pp->p_prev = pp;
2029                 return;
2030         }

2032         /*
2033          * Since we try to take pages from the high end of the pool
2034          * chances are good that the pages to be put on the list will
2035          * go at or near the end of the list. so start at the end and
2036          * work backwards.
2037          */
2038         look = (*poolp)->p_prev;
```



```
2039         while (mfn < mfn_list[look->p_pagenum]) {
2040                 look = look->p_prev;
2041                 if (look == (*poolp)->p_prev)
2042                         break; /* backed all the way to front of list */
2043         }

2045         /* insert after look */
2046         pp->p_prev = look;
2047         pp->p_next = look->p_next;
2048         pp->p_next->p_prev = pp;
2049         look->p_next = pp;
2050         if (mfn < mfn_list[(*poolp)->p_pagenum]) {
2051                 /*
2052                  * we inserted a new first list element
2053                  * adjust pool pointer to newly inserted element
2054                  */
2055                 *poolp = pp;
2056         }
2057 }

2059 /*
2060  * Add a page to the io_pool.  Setting the force flag will force the page
2061  * into the io_pool no matter what.
2062  */
2063 static void
2064 add_page_to_pool(page_t *pp, int force)
2065 {
2066         page_t *highest;
2067         page_t *freep = NULL;

2069         mutex_enter(&io_pool_lock);
2070         /*
2071          * Always keep the scarce low memory pages
2072          */
2073         if (mfn_list[pp->p_pagenum] < PFN_16MEG) {
2074                 ++io_pool_cnt;
2075                 page_io_pool_add(&io_pool_16m, pp);
2076                 goto done;
2077         }
2078         if (io_pool_cnt < io_pool_cnt_max || force || io_pool_4g == NULL) {
2079                 ++io_pool_cnt;
2080                 page_io_pool_add(&io_pool_4g, pp);
2081         } else {
2082                 highest = io_pool_4g->p_prev;
2083                 if (mfn_list[pp->p_pagenum] < mfn_list[highest->p_pagenum]) {
2084                         page_io_pool_sub(&io_pool_4g, highest, highest);
2085                         page_io_pool_add(&io_pool_4g, pp);
2086                         freep = highest;
2087                 } else {
2088                         freep = pp;
2089                 }
2090         }
2091 done:
2092         mutex_exit(&io_pool_lock);
2093         if (freep)
2094                 page_free(freep, 1);
2095 }


2098 int contig_pfn_cnt;     /* no of pfns in the contig pfn list */
2099 int contig_pfn_max;     /* capacity of the contig pfn list */
2100 int next_alloc_pfn;     /* next position in list to start a contig search */
2101 int contig_pfnlist_updates;     /* pfn list update count */
2102 int contig_pfnlist_builds;      /* how many times have we (re)built list */
2103 int contig_pfnlist_buildfailed; /* how many times has list build failed */
2104 int create_contig_pending;      /* nonzero means taskq creating contig list */
```

```
2105 pfn_t *contig_pfn_list = NULL;  /* list of contig pfns in ascending mfn order */

2107 /*
2108  * Function to use in sorting a list of pfns by their underlying mfns.
2109  */
2110 static int
2111 mfn_compare(const void *pfnp1, const void *pfnp2)
2112 {
2113         mfn_t mfn1 = mfn_list[*(pfn_t *)pfnp1];
2114         mfn_t mfn2 = mfn_list[*(pfn_t *)pfnp2];

2116         if (mfn1 > mfn2)
2117                 return (1);
2118         if (mfn1 < mfn2)
2119                 return (-1);
2120         return (0);
2121 }

2123 /*
2124  * Compact the contig_pfn_list by tossing all the non-contiguous
2125  * elements from the list.
2126  */
2127 static void
2128 compact_contig_pfn_list(void)
2129 {
2130         pfn_t pfn, lapfn, prev_lapfn;
2131         mfn_t mfn;
2132         int i, newcnt = 0;

2134         prev_lapfn = 0;
2135         for (i = 0; i < contig_pfn_cnt - 1; i++) {
2136                 pfn = contig_pfn_list[i];
2137                 lapfn = contig_pfn_list[i + 1];
2138                 mfn = mfn_list[pfn];
2139                 /*
2140                  * See if next pfn is for a contig mfn
2141                  */
2142                 if (mfn_list[lapfn] != mfn + 1)
2143                         continue;
2144                 /*
2145                  * pfn and lookahead are both put in list
2146                  * unless pfn is the previous lookahead.
2147                  */
2148                 if (pfn != prev_lapfn)
2149                         contig_pfn_list[newcnt++] = pfn;
2150                 contig_pfn_list[newcnt++] = lapfn;
2151                 prev_lapfn = lapfn;
2152         }
2153         for (i = newcnt; i < contig_pfn_cnt; i++)
2154                 contig_pfn_list[i] = 0;
2155         contig_pfn_cnt = newcnt;
2156 }

2158 /*ARGSUSED*/
2159 static void
2160 call_create_contiglist(void *arg)
2161 {
2162         (void) create_contig_pfnlist(PG_WAIT);
2163 }

2165 /*
2166  * Create list of freelist pfns that have underlying
2167  * contiguous mfns.  The list is kept in ascending mfn order.
2168  * returns 1 if list created else 0.
2169  */
2170 static int
```



```
2171 create_contig_pfnlist(uint_t flags)
2172 {
2173         pfn_t pfn;
2174         page_t *pp;
2175         int ret = 1;

2177         mutex_enter(&contig_list_lock);
2178         if (contig_pfn_list != NULL)
2179                 goto out;
2180         contig_pfn_max = freemem + (freemem / 10);
2181         contig_pfn_list = kmem_zalloc(contig_pfn_max * sizeof (pfn_t),
2182             (flags & PG_WAIT) ? KM_SLEEP : KM_NOSLEEP);
2183         if (contig_pfn_list == NULL) {
2184                 /*
2185                  * If we could not create the contig list (because
2186                  * we could not sleep for memory).  Dispatch a taskq that can
2187                  * sleep to get the memory.
2188                  */
2189                 if (!create_contig_pending) {
2190                         if (taskq_dispatch(system_taskq, call_create_contiglist,
2191                             NULL, TQ_NOSLEEP) != NULL)
2192                                 create_contig_pending = 1;
2193                 }
2194                 contig_pfnlist_buildfailed++;   /* count list build failures */
2195                 ret = 0;
2196                 goto out;
2197         }
2198         create_contig_pending = 0;
2199         ASSERT(contig_pfn_cnt == 0);
2200         for (pfn = 0; pfn < mfn_count; pfn++) {
2201                 pp = page_numtopp_nolock(pfn);
2202                 if (pp == NULL || !PP_ISFREE(pp))
2203                         continue;
2204                 contig_pfn_list[contig_pfn_cnt] = pfn;
2205                 if (++contig_pfn_cnt == contig_pfn_max)
2206                         break;
2207         }
2208         /*
2209          * Sanity check the new list.
2210          */
2211         if (contig_pfn_cnt < 2) { /* no contig pfns */
2212                 contig_pfn_cnt = 0;
2213                 contig_pfnlist_buildfailed++;
2214                 kmem_free(contig_pfn_list, contig_pfn_max * sizeof (pfn_t));
2215                 contig_pfn_list = NULL;
2216                 contig_pfn_max = 0;
2217                 ret = 0;
2218                 goto out;
2219         }
2220         qsort(contig_pfn_list, contig_pfn_cnt, sizeof (pfn_t), mfn_compare);
2221         compact_contig_pfn_list();
2222         /*
2223          * Make sure next search of the newly created contiguous pfn
2224          * list starts at the beginning of the list.
2225          */
2226         next_alloc_pfn = 0;
2227         contig_pfnlist_builds++;        /* count list builds */
2228 out:
2229         mutex_exit(&contig_list_lock);
2230         return (ret);
2231 }


2234 /*
2235  * Toss the current contig pfnlist.  Someone is about to do a massive
2236  * update to pfn<->mfn mappings.  So we have them destroy the list and lock
```

```
 * it till they are done with their update.
 */
void
clear_and_lock_contig_pfnlist()
{
        pfn_t *listp = NULL;
        size_t listsize;

        mutex_enter(&contig_list_lock);
        if (contig_pfn_list != NULL) {
                listp = contig_pfn_list;
                listsize = contig_pfn_max * sizeof (pfn_t);
                contig_pfn_list = NULL;
                contig_pfn_max = contig_pfn_cnt = 0;
        }
        if (listp != NULL)
                kmem_free(listp, listsize);
}

/*
 * Unlock the contig_pfn_list.  The next attempted use of it will cause
 * it to be re-created.
 */
void
unlock_contig_pfnlist()
{
        mutex_exit(&contig_list_lock);
}

/*
 * Update the contiguous pfn list in response to a pfn <-> mfn reassignment
 */
void
update_contig_pfnlist(pfn_t pfn, mfn_t oldmfn, mfn_t newmfn)
{
        int probe_hi, probe_lo, probe_pos, insert_after, insert_point;
        pfn_t probe_pfn;
        mfn_t probe_mfn;
        int drop_lock = 0;

        if (mutex_owner(&contig_list_lock) != curthread) {
                drop_lock = 1;
                mutex_enter(&contig_list_lock);
        }
        if (contig_pfn_list == NULL)
                goto done;
        contig_pfnlist_updates++;
        /*
         * Find the pfn in the current list.  Use a binary chop to locate it.
         */
        probe_hi = contig_pfn_cnt - 1;
        probe_lo = 0;
        probe_pos = (probe_hi + probe_lo) / 2;
        while ((probe_pfn = contig_pfn_list[probe_pos]) != pfn) {
                if (probe_pos == probe_lo) { /* pfn not in list */
                        probe_pos = -1;
                        break;
                }
                if (pfn_to_mfn(probe_pfn) <= oldmfn)
                        probe_lo = probe_pos;
                else
                        probe_hi = probe_pos;
                probe_pos = (probe_hi + probe_lo) / 2;
        }
        if (probe_pos >= 0) {
                /*
```



```
                 * Remove pfn from list and ensure next alloc
                 * position stays in bounds.
                 */
                if (--contig_pfn_cnt <= next_alloc_pfn)
                        next_alloc_pfn = 0;
                if (contig_pfn_cnt < 2) { /* no contig pfns */
                        contig_pfn_cnt = 0;
                        kmem_free(contig_pfn_list,
                            contig_pfn_max * sizeof (pfn_t));
                        contig_pfn_list = NULL;
                        contig_pfn_max = 0;
                        goto done;
                }
                ovbcopy(&contig_pfn_list[probe_pos + 1],
                    &contig_pfn_list[probe_pos],
                    (contig_pfn_cnt - probe_pos) * sizeof (pfn_t));
        }
        if (newmfn == MFN_INVALID)
                goto done;
        /*
         * Check if new mfn has adjacent mfns in the list
         */
        probe_hi = contig_pfn_cnt - 1;
        probe_lo = 0;
        insert_after = -2;
        do {
                probe_pos = (probe_hi + probe_lo) / 2;
                probe_mfn = pfn_to_mfn(contig_pfn_list[probe_pos]);
                if (newmfn == probe_mfn + 1)
                        insert_after = probe_pos;
                else if (newmfn == probe_mfn - 1)
                        insert_after = probe_pos - 1;
                if (probe_pos == probe_lo)
                        break;
                if (probe_mfn <= newmfn)
                        probe_lo = probe_pos;
                else
                        probe_hi = probe_pos;
        } while (insert_after == -2);
        /*
         * If there is space in the list and there are adjacent mfns
         * insert the pfn in to its proper place in the list.
         */
        if (insert_after != -2 && contig_pfn_cnt + 1 <= contig_pfn_max) {
                insert_point = insert_after + 1;
                ovbcopy(&contig_pfn_list[insert_point],
                    &contig_pfn_list[insert_point + 1],
                    (contig_pfn_cnt - insert_point) * sizeof (pfn_t));
                contig_pfn_list[insert_point] = pfn;
                contig_pfn_cnt++;
        }
done:
        if (drop_lock)
                mutex_exit(&contig_list_lock);
}

/*
 * Called to (re-)populate the io_pool from the free page lists.
 */
long
populate_io_pool(void)
{
        pfn_t pfn;
        mfn_t mfn, max_mfn;
        page_t *pp;
```

```
2369         /*
2370          * Figure out the bounds of the pool on first invocation.
2371          * We use a percentage of memory for the io pool size.
2372          * we allow that to shrink, but not to less than a fixed minimum
2373          */
2374         if (io_pool_cnt_max == 0) {
2375                 io_pool_cnt_max = physmem / (100 / io_pool_physmem_pct);
2376                 io_pool_cnt_lowater = io_pool_cnt_max;
2377                 /*
2378                  * This is the first time in populate_io_pool, grab a va to use
2379                  * when we need to allocate pages.
2380                  */
2381                 io_pool_kva = vmem_alloc(heap_arena, PAGESIZE, VM_SLEEP);
2382         }
2383         /*
2384          * If we are out of pages in the pool, then grow the size of the pool
2385          */
2386         if (io_pool_cnt == 0) {
2387                 /*
2388                  * Grow the max size of the io pool by 5%, but never more than
2389                  * 25% of physical memory.
2390                  */
2391                 if (io_pool_cnt_max < physmem / 4)
2392                         io_pool_cnt_max += io_pool_cnt_max / 20;
2393         }
2394         io_pool_grows++;        /* should be a kstat? */

2396         /*
2397          * Get highest mfn on this platform, but limit to the 32 bit DMA max.
2398          */
2399         (void) mfn_to_pfn(start_mfn);
2400         max_mfn = MIN(cached_max_mfn, PFN_4GIG);
2401         for (mfn = start_mfn; mfn < max_mfn; start_mfn = ++mfn) {
2402                 pfn = mfn_to_pfn(mfn);
2403                 if (pfn & PFN_IS_FOREIGN_MFN)
2404                         continue;
2405                 /*
2406                  * try to allocate it from free pages
2407                  */
2408                 pp = page_numtopp_alloc(pfn);
2409                 if (pp == NULL)
2410                         continue;
2411                 PP_CLRFREE(pp);
2412                 add_page_to_pool(pp, 1);
2413                 if (io_pool_cnt >= io_pool_cnt_max)
2414                         break;
2415         }

2417         return (io_pool_cnt);
2418 }

2420 /*
2421  * Destroy a page that was being used for DMA I/O. It may or
2422  * may not actually go back to the io_pool.
2423  */
2424 void
2425 page_destroy_io(page_t *pp)
2426 {
2427         mfn_t mfn = mfn_list[pp->p_pagenum];

2429         /*
2430          * When the page was alloc'd a reservation was made, release it now
2431          */
2432         page_unresv(1);
2433         /*
2434          * Unload translations, if any, then hash out the
```



```
2435          * page to erase its identity.
2436          */
2437         (void) hat_pageunload(pp, HAT_FORCE_PGUNLOAD);
2438         page_hashout(pp, NULL);

2440         /*
2441          * If the page came from the free lists, just put it back to them.
2442          * DomU pages always go on the free lists as well.
2443          */
2444         if (!DOMAIN_IS_INITDOMAIN(xen_info) || mfn >= PFN_4GIG) {
2445                 page_free(pp, 1);
2446                 return;
2447         }

2449         add_page_to_pool(pp, 0);
2450 }


2453 long contig_searches;           /* count of times contig pages requested */
2454 long contig_search_restarts;    /* count of contig ranges tried */
2455 long contig_search_failed;      /* count of contig alloc failures */

2457 /*
2458  * Free partial page list
2459  */
2460 static void
2461 free_partial_list(page_t **pplist)
2462 {
2463         page_t *pp;

2465         while (*pplist != NULL) {
2466                 pp = *pplist;
2467                 page_io_pool_sub(pplist, pp, pp);
2468                 page_free(pp, 1);
2469         }
2470 }

2472 /*
2473  * Look thru the contiguous pfns that are not part of the io_pool for
2474  * contiguous free pages.  Return a list of the found pages or NULL.
2475  */
2476 page_t *
2477 find_contig_free(uint_t npages, uint_t flags, uint64_t pfnseg,
2478     pgcnt_t pfnalign)
2479 {
2480         page_t *pp, *plist = NULL;
2481         mfn_t mfn, prev_mfn, start_mfn;
2482         pfn_t pfn;
2483         int pages_needed, pages_requested;
2484         int search_start;

2486         /*
2487          * create the contig pfn list if not already done
2488          */
2489 retry:
2490         mutex_enter(&contig_list_lock);
2491         if (contig_pfn_list == NULL) {
2492                 mutex_exit(&contig_list_lock);
2493                 if (!create_contig_pfnlist(flags)) {
2494                         return (NULL);
2495                 }
2496                 goto retry;
2497         }
2498         contig_searches++;
2499         /*
2500          * Search contiguous pfn list for physically contiguous pages not in
```

```
2501          * the io_pool.  Start the search where the last search left off.
2502          */
2503         pages_requested = pages_needed = npages;
2504         search_start = next_alloc_pfn;
2505         start_mfn = prev_mfn = 0;
2506         while (pages_needed) {
2507                 pfn = contig_pfn_list[next_alloc_pfn];
2508                 mfn = pfn_to_mfn(pfn);
2509                 /*
2510                  * Check if mfn is first one or contig to previous one and
2511                  * if page corresponding to mfn is free and that mfn
2512                  * range is not crossing a segment boundary.
2513                  */
2514                 if ((prev_mfn == 0 || mfn == prev_mfn + 1) &&
2515                     (pp = page_numtopp_alloc(pfn)) != NULL &&
2516                     !((mfn & pfnseg) < (start_mfn & pfnseg))) {
2517                         PP_CLRFREE(pp);
2518                         page_io_pool_add(&plist, pp);
2519                         pages_needed--;
2520                         if (prev_mfn == 0) {
2521                                 if (pfnalign &&
2522                                     mfn != P2ROUNDUP(mfn, pfnalign)) {
2523                                         /*
2524                                          * not properly aligned
2525                                          */
2526                                         contig_search_restarts++;
2527                                         free_partial_list(&plist);
2528                                         pages_needed = pages_requested;
2529                                         start_mfn = prev_mfn = 0;
2530                                         goto skip;
2531                                 }
2532                                 start_mfn = mfn;
2533                         }
2534                         prev_mfn = mfn;
2535                 } else {
2536                         contig_search_restarts++;
2537                         free_partial_list(&plist);
2538                         pages_needed = pages_requested;
2539                         start_mfn = prev_mfn = 0;
2540                 }
2541 skip:
2542                 if (++next_alloc_pfn == contig_pfn_cnt)
2543                         next_alloc_pfn = 0;
2544                 if (next_alloc_pfn == search_start)
2545                         break; /* all pfns searched */
2546         }
2547         mutex_exit(&contig_list_lock);
2548         if (pages_needed) {
2549                 contig_search_failed++;
2550                 /*
2551                  * Failed to find enough contig pages.
2552                  * free partial page list
2553                  */
2554                 free_partial_list(&plist);
2555         }
2556         return (plist);
2557 }

2559 /*
2560  * Search the reserved io pool pages for a page range with the
2561  * desired characteristics.
2562  */
2563 page_t *
2564 page_io_pool_alloc(ddi_dma_attr_t *mattr, int contig, pgcnt_t minctg)
2565 {
2566         page_t *pp_first, *pp_last;
```



```
2567         page_t *pp, **poolp;
2568         pgcnt_t nwanted, pfnalign;
2569         uint64_t pfnseg;
2570         mfn_t mfn, tmfn, hi_mfn, lo_mfn;
2571         int align, attempt = 0;

2573         if (minctg == 1)
2574                 contig = 0;
2575         lo_mfn = mmu_btop(mattr->dma_attr_addr_lo);
2576         hi_mfn = mmu_btop(mattr->dma_attr_addr_hi);
2577         pfnseg = mmu_btop(mattr->dma_attr_seg);
2578         align = maxbit(mattr->dma_attr_align, mattr->dma_attr_minxfer);
2579         if (align > MMU_PAGESIZE)
2580                 pfnalign = mmu_btop(align);
2581         else
2582                 pfnalign = 0;

2584 try_again:
2585         /*
2586          * See if we want pages for a legacy device
2587          */
2588         if (hi_mfn < PFN_16MEG)
2589                 poolp = &io_pool_16m;
2590         else
2591                 poolp = &io_pool_4g;
2592 try_smaller:
2593         /*
2594          * Take pages from I/O pool. We'll use pages from the highest
2595          * MFN range possible.
2596          */
2597         pp_first = pp_last = NULL;
2598         mutex_enter(&io_pool_lock);
2599         nwanted = minctg;
2600         for (pp = *poolp; pp && nwanted > 0; ) {
2601                 pp = pp->p_prev;

2603                 /*
2604                  * skip pages above allowable range
2605                  */
2606                 mfn = mfn_list[pp->p_pagenum];
2607                 if (hi_mfn < mfn)
2608                         goto skip;

2610                 /*
2611                  * stop at pages below allowable range
2612                  */
2613                 if (lo_mfn > mfn)
2614                         break;
2615 restart:
2616                 if (pp_last == NULL) {
2617                         /*
2618                          * Check alignment
2619                          */
2620                         tmfn = mfn - (minctg - 1);
2621                         if (pfnalign && tmfn != P2ROUNDUP(tmfn, pfnalign))
2622                                 goto skip; /* not properly aligned */
2623                         /*
2624                          * Check segment
2625                          */
2626                         if ((mfn & pfnseg) < (tmfn & pfnseg))
2627                                 goto skip; /* crosses seg boundary */
2628                         /*
2629                          * Start building page list
2630                          */
2631                         pp_first = pp_last = pp;
2632                         nwanted--;
```

```
2633                 } else {
2634                         /*
2635                          * check physical contiguity if required
2636                          */
2637                         if (contig &&
2638                             mfn_list[pp_first->p_pagenum] != mfn + 1) {
2639                                 /*
2640                                  * not a contiguous page, restart list.
2641                                  */
2642                                 pp_last = NULL;
2643                                 nwanted = minctg;
2644                                 goto restart;
2645                         } else { /* add page to list */
2646                                 pp_first = pp;
2647                                 nwanted--;
2648                         }
2649                 }
2650 skip:
2651                 if (pp == *poolp)
2652                         break;
2653         }

2655         /*
2656          * If we didn't find memory. Try the more constrained pool, then
2657          * sweep free pages into the DMA pool and try again.
2658          */
2659         if (nwanted != 0) {
2660                 mutex_exit(&io_pool_lock);
2661                 /*
2662                  * If we were looking in the less constrained pool and
2663                  * didn't find pages, try the more constrained pool.
2664                  */
2665                 if (poolp == &io_pool_4g) {
2666                         poolp = &io_pool_16m;
2667                         goto try_smaller;
2668                 }
2669                 kmem_reap();
2670                 if (++attempt < 4) {
2671                         /*
2672                          * Grab some more io_pool pages
2673                          */
2674                         (void) populate_io_pool();
2675                         goto try_again; /* go around and retry */
2676                 }
2677                 return (NULL);
2678         }
2679         /*
2680          * Found the pages, now snip them from the list
2681          */
2682         page_io_pool_sub(poolp, pp_first, pp_last);
2683         io_pool_cnt -= minctg;
2684         /*
2685          * reset low water mark
2686          */
2687         if (io_pool_cnt < io_pool_cnt_lowater)
2688                 io_pool_cnt_lowater = io_pool_cnt;
2689         mutex_exit(&io_pool_lock);
2690         return (pp_first);
2691 }

2693 page_t *
2694 page_swap_with_hypervisor(struct vnode *vp, u_offset_t off, caddr_t vaddr,
2695     ddi_dma_attr_t *mattr, uint_t flags, pgcnt_t minctg)
2696 {
2697         uint_t kflags;
2698         int order, extra, extpages, i, contig, nbits, extents;
```



```
2699         page_t *pp, *expp, *pp_first, **pplist = NULL;
2700         mfn_t *mfnlist = NULL;

2702         contig = flags & PG_PHYSCONTIG;
2703         if (minctg == 1)
2704                 contig = 0;
2705         flags &= ~PG_PHYSCONTIG;
2706         kflags = flags & PG_WAIT ? KM_SLEEP : KM_NOSLEEP;
2707         /*
2708          * Hypervisor will allocate extents, if we want contig
2709          * pages extent must be >= minctg
2710          */
2711         if (contig) {
2712                 order = highbit(minctg) - 1;
2713                 if (minctg & ((1 << order) - 1))
2714                         order++;
2715                 extpages = 1 << order;
2716         } else {
2717                 order = 0;
2718                 extpages = minctg;
2719         }
2720         if (extpages > minctg) {
2721                 extra = extpages - minctg;
2722                 if (!page_resv(extra, kflags))
2723                         return (NULL);
2724         }
2725         pp_first = NULL;
2726         pplist = kmem_alloc(extpages * sizeof (page_t *), kflags);
2727         if (pplist == NULL)
2728                 goto balloon_fail;
2729         mfnlist = kmem_alloc(extpages * sizeof (mfn_t), kflags);
2730         if (mfnlist == NULL)
2731                 goto balloon_fail;
2732         pp = page_create_va(vp, off, minctg * PAGESIZE, flags, &kvseg, vaddr);
2733         if (pp == NULL)
2734                 goto balloon_fail;
2735         pp_first = pp;
2736         if (extpages > minctg) {
2737                 /*
2738                  * fill out the rest of extent pages to swap
2739                  * with the hypervisor
2740                  */
2741                 for (i = 0; i < extra; i++) {
2742                         expp = page_create_va(vp,
2743                             (u_offset_t)(uintptr_t)io_pool_kva,
2744                             PAGESIZE, flags, &kvseg, io_pool_kva);
2745                         if (expp == NULL)
2746                                 goto balloon_fail;
2747                         (void) hat_pageunload(expp, HAT_FORCE_PGUNLOAD);
2748                         page_io_unlock(expp);
2749                         page_hashout(expp, NULL);
2750                         page_io_lock(expp);
2751                         /*
2752                          * add page to end of list
2753                          */
2754                         expp->p_prev = pp_first->p_prev;
2755                         expp->p_next = pp_first;
2756                         expp->p_prev->p_next = expp;
2757                         pp_first->p_prev = expp;
2758                 }

2760         }
2761         for (i = 0; i < extpages; i++) {
2762                 pplist[i] = pp;
2763                 pp = pp->p_next;
2764         }
```

```
2765         nbits = highbit(mattr->dma_attr_addr_hi);
2766         extents = contig ? 1 : minctg;
2767         if (balloon_replace_pages(extents, pplist, nbits, order,
2768             mfnlist) != extents) {
2769                 if (ioalloc_dbg)
2770                         cmn_err(CE_NOTE, "request to hypervisor"
2771                             " for %d pages, maxaddr %" PRIx64 " failed",
2772                             extpages, mattr->dma_attr_addr_hi);
2773                 goto balloon_fail;
2774         }

2776         kmem_free(pplist, extpages * sizeof (page_t *));
2777         kmem_free(mfnlist, extpages * sizeof (mfn_t));
2778         /*
2779          * Return any excess pages to free list
2780          */
2781         if (extpages > minctg) {
2782                 for (i = 0; i < extra; i++) {
2783                         pp = pp_first->p_prev;
2784                         page_sub(&pp_first, pp);
2785                         page_io_unlock(pp);
2786                         page_unresv(1);
2787                         page_free(pp, 1);
2788                 }
2789         }
2790         return (pp_first);
2791 balloon_fail:
2792         /*
2793          * Return pages to free list and return failure
2794          */
2795         while (pp_first != NULL) {
2796                 pp = pp_first;
2797                 page_sub(&pp_first, pp);
2798                 page_io_unlock(pp);
2799                 if (pp->p_vnode != NULL)
2800                         page_hashout(pp, NULL);
2801                 page_free(pp, 1);
2802         }
2803         if (pplist)
2804                 kmem_free(pplist, extpages * sizeof (page_t *));
2805         if (mfnlist)
2806                 kmem_free(mfnlist, extpages * sizeof (mfn_t));
2807         page_unresv(extpages - minctg);
2808         return (NULL);
2809 }

2811 static void
2812 return_partial_alloc(page_t *plist)
2813 {
2814         page_t *pp;

2816         while (plist != NULL) {
2817                 pp = plist;
2818                 page_sub(&plist, pp);
2819                 page_io_unlock(pp);
2820                 page_destroy_io(pp);
2821         }
2822 }

2824 static page_t *
2825 page_get_contigpages(
2826         struct vnode    *vp,
2827         u_offset_t      off,
2828         int             *npagesp,
2829         uint_t          flags,
2830         caddr_t         vaddr,
```



```
2831         ddi_dma_attr_t  *mattr)
2832 {
2833         mfn_t   max_mfn = HYPERVISOR_memory_op(XENMEM_maximum_ram_page, NULL);
2834         page_t  *plist; /* list to return */
2835         page_t  *pp, *mcpl;
2836         int     contig, anyaddr, npages, getone = 0;
2837         mfn_t   lo_mfn;
2838         mfn_t   hi_mfn;
2839         pgcnt_t pfnalign = 0;
2840         int     align, sgllen;
2841         uint64_t pfnseg;
2842         pgcnt_t minctg;

2844         npages = *npagesp;
2845         ASSERT(mattr != NULL);
2846         lo_mfn = mmu_btop(mattr->dma_attr_addr_lo);
2847         hi_mfn = mmu_btop(mattr->dma_attr_addr_hi);
2848         sgllen = mattr->dma_attr_sgllen;
2849         pfnseg = mmu_btop(mattr->dma_attr_seg);
2850         align = maxbit(mattr->dma_attr_align, mattr->dma_attr_minxfer);
2851         if (align > MMU_PAGESIZE)
2852                 pfnalign = mmu_btop(align);

2854         contig = flags & PG_PHYSCONTIG;
2855         if (npages == -1) {
2856                 npages = 1;
2857                 pfnalign = 0;
2858         }
2859         /*
2860          * Clear the contig flag if only one page is needed.
2861          */
2862         if (npages == 1) {
2863                 getone = 1;
2864                 contig = 0;
2865         }

2867         /*
2868          * Check if any page in the system is fine.
2869          */
2870         anyaddr = lo_mfn == 0 && hi_mfn >= max_mfn;
2871         if (!contig && anyaddr && !pfnalign) {
2872                 flags &= ~PG_PHYSCONTIG;
2873                 plist = page_create_va(vp, off, npages * MMU_PAGESIZE,
2874                     flags, &kvseg, vaddr);
2875                 if (plist != NULL) {
2876                         *npagesp = 0;
2877                         return (plist);
2878                 }
2879         }
2880         plist = NULL;
2881         minctg = howmany(npages, sgllen);
2882         while (npages > sgllen || getone) {
2883                 if (minctg > npages)
2884                         minctg = npages;
2885                 mcpl = NULL;
2886                 /*
2887                  * We could want contig pages with no address range limits.
2888                  */
2889                 if (anyaddr && contig) {
2890                         /*
2891                          * Look for free contig pages to satisfy the request.
2892                          */
2893                         mcpl = find_contig_free(minctg, flags, pfnseg,
2894                             pfnalign);
2895                 }
2896                 /*
```

```
2897                  * Try the reserved io pools next
2898                  */
2899                 if (mcpl == NULL)
2900                         mcpl = page_io_pool_alloc(mattr, contig, minctg);
2901                 if (mcpl != NULL) {
2902                         pp = mcpl;
2903                         do {
2904                                 if (!page_hashin(pp, vp, off, NULL)) {
2905                                         panic("page_get_contigpages:"
2906                                             " hashin failed"
2907                                             " pp %p, vp %p, off %llx",
2908                                             (void *)pp, (void *)vp, off);
2909                                 }
2910                                 off += MMU_PAGESIZE;
2911                                 PP_CLRFREE(pp);
2912                                 PP_CLRAGED(pp);
2913                                 page_set_props(pp, P_REF);
2914                                 page_io_lock(pp);
2915                                 pp = pp->p_next;
2916                         } while (pp != mcpl);
2917                 } else {
2918                         /*
2919                          * Hypervisor exchange doesn't handle segment or
2920                          * alignment constraints
2921                          */
2922                         if (mattr->dma_attr_seg < mattr->dma_attr_addr_hi ||
2923                             pfnalign)
2924                                 goto fail;
2925                         /*
2926                          * Try exchanging pages with the hypervisor
2927                          */
2928                         mcpl = page_swap_with_hypervisor(vp, off, vaddr, mattr,
2929                             flags, minctg);
2930                         if (mcpl == NULL)
2931                                 goto fail;
2932                         off += minctg * MMU_PAGESIZE;
2933                 }
2934                 check_dma(mattr, mcpl, minctg);
2935                 /*
2936                  * Here with a minctg run of contiguous pages, add them to the
2937                  * list we will return for this request.
2938                  */
2939                 page_list_concat(&plist, &mcpl);
2940                 npages -= minctg;
2941                 *npagesp = npages;
2942                 sgllen--;
2943                 if (getone)
2944                         break;
2945         }
2946         return (plist);
2947 fail:
2948         return_partial_alloc(plist);
2949         return (NULL);
2950 }

2952 /*
2953  * Allocator for domain 0 I/O pages. We match the required
2954  * DMA attributes and contiguity constraints.
2955  */
2956 /*ARGSUSED*/
2957 page_t *
2958 page_create_io(
2959         struct vnode    *vp,
2960         u_offset_t      off,
2961         uint_t          bytes,
2962         uint_t          flags,
```



```
2963         struct as       *as,
2964         caddr_t         vaddr,
2965         ddi_dma_attr_t  *mattr)
2966 {
2967         page_t  *plist = NULL, *pp;
2968         int     npages = 0, contig, anyaddr, pages_req;
2969         mfn_t   lo_mfn;
2970         mfn_t   hi_mfn;
2971         pgcnt_t pfnalign = 0;
2972         int     align;
2973         int     is_domu = 0;
2974         int     dummy, bytes_got;
2975         mfn_t   max_mfn = HYPERVISOR_memory_op(XENMEM_maximum_ram_page, NULL);

2977         ASSERT(mattr != NULL);
2978         lo_mfn = mmu_btop(mattr->dma_attr_addr_lo);
2979         hi_mfn = mmu_btop(mattr->dma_attr_addr_hi);
2980         align = maxbit(mattr->dma_attr_align, mattr->dma_attr_minxfer);
2981         if (align > MMU_PAGESIZE)
2982                 pfnalign = mmu_btop(align);

2984         /*
2985          * Clear the contig flag if only one page is needed or the scatter
2986          * gather list length is >= npages.
2987          */
2988         pages_req = npages = mmu_btopr(bytes);
2989         contig = (flags & PG_PHYSCONTIG);
2990         bytes = P2ROUNDUP(bytes, MMU_PAGESIZE);
2991         if (bytes == MMU_PAGESIZE || mattr->dma_attr_sgllen >= npages)
2992                 contig = 0;

2994         /*
2995          * Check if any old page in the system is fine.
2996          * DomU should always go down this path.
2997          */
2998         is_domu = !DOMAIN_IS_INITDOMAIN(xen_info);
2999         anyaddr = lo_mfn == 0 && hi_mfn >= max_mfn && !pfnalign;
3000         if ((!contig && anyaddr) || is_domu) {
3001                 flags &= ~PG_PHYSCONTIG;
3002                 plist = page_create_va(vp, off, bytes, flags, &kvseg, vaddr);
3003                 if (plist != NULL)
3004                         return (plist);
3005                 else if (is_domu)
3006                         return (NULL); /* no memory available */
3007         }
3008         /*
3009          * DomU should never reach here
3010          */
3011         if (contig) {
3012                 plist = page_get_contigpages(vp, off, &npages, flags, vaddr,
3013                     mattr);
3014                 if (plist == NULL)
3015                         goto fail;
3016                 bytes_got = (pages_req - npages) << MMU_PAGESHIFT;
3017                 vaddr += bytes_got;
3018                 off += bytes_got;
3019                 /*
3020                  * We now have all the contiguous pages we need, but
3021                  * we may still need additional non-contiguous pages.
3022                  */
3023         }
3024         /*
3025          * now loop collecting the requested number of pages, these do
3026          * not have to be contiguous pages but we will use the contig
3027          * page alloc code to get the pages since it will honor any
3028          * other constraints the pages may have.
```

```
3029          */
3030         while (npages--) {
3031                 dummy = -1;
3032                 pp = page_get_contigpages(vp, off, &dummy, flags, vaddr, mattr);
3033                 if (pp == NULL)
3034                         goto fail;
3035                 page_add(&plist, pp);
3036                 vaddr += MMU_PAGESIZE;
3037                 off += MMU_PAGESIZE;
3038         }
3039         return (plist);
3040 fail:
3041         /*
3042          * Failed to get enough pages, return ones we did get
3043          */
3044         return_partial_alloc(plist);
3045         return (NULL);
3046 }

3048 /*
3049  * Lock and return the page with the highest mfn that we can find.  last_mfn
3050  * holds the last one found, so the next search can start from there.  We
3051  * also keep a counter so that we don't loop forever if the machine has no
3052  * free pages.
3053  *
3054  * This is called from the balloon thread to find pages to give away.  new_high
3055  * is used when new mfn's have been added to the system - we will reset our
3056  * search if the new mfn's are higher than our current search position.
3057  */
3058 page_t *
3059 page_get_high_mfn(mfn_t new_high)
3060 {
3061         static mfn_t last_mfn = 0;
3062         pfn_t pfn;
3063         page_t *pp;
3064         ulong_t loop_count = 0;

3066         if (new_high > last_mfn)
3067                 last_mfn = new_high;

3069         for (; loop_count < mfn_count; loop_count++, last_mfn--) {
3070                 if (last_mfn == 0) {
3071                         last_mfn = cached_max_mfn;
3072                 }

3074                 pfn = mfn_to_pfn(last_mfn);
3075                 if (pfn & PFN_IS_FOREIGN_MFN)
3076                         continue;

3078                 /* See if the page is free.  If so, lock it. */
3079                 pp = page_numtopp_alloc(pfn);
3080                 if (pp == NULL)
3081                         continue;
3082                 PP_CLRFREE(pp);

3084                 ASSERT(PAGE_EXCL(pp));
3085                 ASSERT(pp->p_vnode == NULL);
3086                 ASSERT(!hat_page_is_mapped(pp));
3087                 last_mfn--;
3088                 return (pp);
3089         }
3090         return (NULL);
3091 }

3093 #else /* !__xpv */
```



```
3095 /*
3096  * get a page from any list with the given mnode
3097  */
3098 static page_t *
3099 page_get_mnode_anylist(ulong_t origbin, uchar_t szc, uint_t flags,
3100     int mnode, int mtype, ddi_dma_attr_t *dma_attr)
3101 {
3102         kmutex_t                *pcm;
3103         int                     i;
3104         page_t                  *pp;
3105         page_t                  *first_pp;
3106         uint64_t                pgaddr;
3107         ulong_t                 bin;
3108         int                     mtypestart;
3109         int                     plw_initialized;
3110         page_list_walker_t      plw;

3112         VM_STAT_ADD(pga_vmstats.pgma_alloc);

3114         ASSERT((flags & PG_MATCH_COLOR) == 0);
3115         ASSERT(szc == 0);
3116         ASSERT(dma_attr != NULL);

3118         MTYPE_START(mnode, mtype, flags);
3119         if (mtype < 0) {
3120                 VM_STAT_ADD(pga_vmstats.pgma_allocempty);
3121                 return (NULL);
3122         }

3124         mtypestart = mtype;

3126         bin = origbin;

3128         /*
3129          * check up to page_colors + 1 bins - origbin may be checked twice
3130          * because of BIN_STEP skip
3131          */
3132         do {
3133                 plw_initialized = 0;

3135                 for (plw.plw_count = 0;
3136                     plw.plw_count < page_colors; plw.plw_count++) {

3138                         if (PAGE_FREELISTS(mnode, szc, bin, mtype) == NULL)
3139                                 goto nextfreebin;

3141                         pcm = PC_BIN_MUTEX(mnode, bin, PG_FREE_LIST);
3142                         mutex_enter(pcm);
3143                         pp = PAGE_FREELISTS(mnode, szc, bin, mtype);
3144                         first_pp = pp;
3145                         while (pp != NULL) {
3146                                 if (IS_DUMP_PAGE(pp) || page_trylock(pp,
3147                                     SE_EXCL) == 0) {
3148                                         pp = pp->p_next;
3149                                         if (pp == first_pp) {
3150                                                 pp = NULL;
3151                                         }
3152                                         continue;
3153                                 }

3155                                 ASSERT(PP_ISFREE(pp));
3156                                 ASSERT(PP_ISAGED(pp));
3157                                 ASSERT(pp->p_vnode == NULL);
3158                                 ASSERT(pp->p_hash == NULL);
3159                                 ASSERT(pp->p_offset == (u_offset_t)-1);
3160                                 ASSERT(pp->p_szc == szc);
```

```
3161                                ASSERT(PFN_2_MEM_NODE(pp->p_pagenum) == mnode);
3162                                /* check if page within DMA attributes */
3163                                pgaddr = pa_to_ma(pfn_to_pa(pp->p_pagenum));
3164                                if ((pgaddr >= dma_attr->dma_attr_addr_lo) &&
3165                                    (pgaddr + MMU_PAGESIZE - 1 <=
3166                                    dma_attr->dma_attr_addr_hi)) {
3167                                        break;
3168                                }

3170                                /* continue looking */
3171                                page_unlock(pp);
3172                                pp = pp->p_next;
3173                                if (pp == first_pp)
3174                                        pp = NULL;

3176                        }
3177                        if (pp != NULL) {
3178                                ASSERT(mtype == PP_2_MTYPE(pp));
3179                                ASSERT(pp->p_szc == 0);

3181                                /* found a page with specified DMA attributes */
3182                                page_sub(&PAGE_FREELISTS(mnode, szc, bin,
3183                                    mtype), pp);
3184                                page_ctr_sub(mnode, mtype, pp, PG_FREE_LIST);

3186                                if ((PP_ISFREE(pp) == 0) ||
3187                                    (PP_ISAGED(pp) == 0)) {
3188                                        cmn_err(CE_PANIC, "page %p is not free",
3189                                            (void *)pp);
3190                                }

3192                                mutex_exit(pcm);
3193                                check_dma(dma_attr, pp, 1);
3194                                VM_STAT_ADD(pga_vmstats.pgma_allocok);
3195                                return (pp);
3196                        }
3197                        mutex_exit(pcm);
3198 nextfreebin:
3199                        if (plw_initialized == 0) {
3200                                page_list_walk_init(szc, 0, bin, 1, 0, &plw);
3201                                ASSERT(plw.plw_ceq_dif == page_colors);
3202                                plw_initialized = 1;
3203                        }

3205                        if (plw.plw_do_split) {
3206                                pp = page_freelist_split(szc, bin, mnode,
3207                                    mtype,
3208                                    mmu_btop(dma_attr->dma_attr_addr_lo),
3209                                    mmu_btop(dma_attr->dma_attr_addr_hi + 1),
3210                                    &plw);
3211                                if (pp != NULL) {
3212                                        check_dma(dma_attr, pp, 1);
3213                                        return (pp);
3214                                }
3215                        }

3217                        bin = page_list_walk_next_bin(szc, bin, &plw);
3218                }

3220                MTYPE_NEXT(mnode, mtype, flags);
3221        } while (mtype >= 0);

3223        /* failed to find a page in the freelist; try it in the cachelist */

3225        /* reset mtype start for cachelist search */
3226        mtype = mtypestart;
```



```
3227        ASSERT(mtype >= 0);

3229        /* start with the bin of matching color */
3230        bin = origbin;

3232        do {
3233                for (i = 0; i <= page_colors; i++) {
3234                        if (PAGE_CACHELISTS(mnode, bin, mtype) == NULL)
3235                                goto nextcachebin;
3236                        pcm = PC_BIN_MUTEX(mnode, bin, PG_CACHE_LIST);
3237                        mutex_enter(pcm);
3238                        pp = PAGE_CACHELISTS(mnode, bin, mtype);
3239                        first_pp = pp;
3240                        while (pp != NULL) {
3241                                if (IS_DUMP_PAGE(pp) || page_trylock(pp,
3242                                    SE_EXCL) == 0) {
3243                                        pp = pp->p_next;
3244                                        if (pp == first_pp)
3245                                                pp = NULL;
3246                                        continue;
3247                                }
3248                                ASSERT(pp->p_vnode);
3249                                ASSERT(PP_ISAGED(pp) == 0);
3250                                ASSERT(pp->p_szc == 0);
3251                                ASSERT(PFN_2_MEM_NODE(pp->p_pagenum) == mnode);

3253                                /* check if page within DMA attributes */

3255                                pgaddr = pa_to_ma(pfn_to_pa(pp->p_pagenum));
3256                                if ((pgaddr >= dma_attr->dma_attr_addr_lo) &&
3257                                    (pgaddr + MMU_PAGESIZE - 1 <=
3258                                    dma_attr->dma_attr_addr_hi)) {
3259                                        break;
3260                                }

3262                                /* continue looking */
3263                                page_unlock(pp);
3264                                pp = pp->p_next;
3265                                if (pp == first_pp)
3266                                        pp = NULL;
3267                        }

3269                        if (pp != NULL) {
3270                                ASSERT(mtype == PP_2_MTYPE(pp));
3271                                ASSERT(pp->p_szc == 0);

3273                                /* found a page with specified DMA attributes */
3274                                page_sub(&PAGE_CACHELISTS(mnode, bin,
3275                                    mtype), pp);
3276                                page_ctr_sub(mnode, mtype, pp, PG_CACHE_LIST);

3278                                mutex_exit(pcm);
3279                                ASSERT(pp->p_vnode);
3280                                ASSERT(PP_ISAGED(pp) == 0);
3281                                check_dma(dma_attr, pp, 1);
3282                                VM_STAT_ADD(pga_vmstats.pgma_allocok);
3283                                return (pp);
3284                        }
3285                        mutex_exit(pcm);
3286 nextcachebin:
3287                        bin += (i == 0) ? BIN_STEP : 1;
3288                        bin &= page_colors_mask;
3289                }
3290                MTYPE_NEXT(mnode, mtype, flags);
3291        } while (mtype >= 0);
```

```
3293         VM_STAT_ADD(pga_vmstats.pgma_allocfailed);
3294         return (NULL);
3295 }

3297 /*
3298  * This function is similar to page_get_freelist()/page_get_cachelist()
3299  * but it searches both the lists to find a page with the specified
3300  * color (or no color) and DMA attributes. The search is done in the
3301  * freelist first and then in the cache list within the highest memory
3302  * range (based on DMA attributes) before searching in the lower
3303  * memory ranges.
3304  *
3305  * Note: This function is called only by page_create_io().
3306  */
3307 /*ARGSUSED*/
3308 static page_t *
3309 page_get_anylist(struct vnode *vp, u_offset_t off, struct as *as, caddr_t vaddr,
3310     size_t size, uint_t flags, ddi_dma_attr_t *dma_attr, lgrp_t *lgrp)
3311 {
3312         uint_t          bin;
3313         int             mtype;
3314         page_t          *pp;
3315         int             n;
3316         int             m;
3317         int             szc;
3318         int             fullrange;
3319         int             mnode;
3320         int             local_failed_stat = 0;
3321         lgrp_mnode_cookie_t     lgrp_cookie;

3323         VM_STAT_ADD(pga_vmstats.pga_alloc);

3325         /* only base pagesize currently supported */
3326         if (size != MMU_PAGESIZE)
3327                 return (NULL);

3329         /*
3330          * If we're passed a specific lgroup, we use it.  Otherwise,
3331          * assume first-touch placement is desired.
3332          */
3333         if (!LGRP_EXISTS(lgrp))
3334                 lgrp = lgrp_home_lgrp();

3336         /* LINTED */
3337         AS_2_BIN(as, seg, vp, vaddr, bin, 0);

3339         /*
3340          * Only hold one freelist or cachelist lock at a time, that way we
3341          * can start anywhere and not have to worry about lock
3342          * ordering.
3343          */
3344         if (dma_attr == NULL) {
3345                 n = mtype16m;
3346                 m = mtypetop;
3347                 fullrange = 1;
3348                 VM_STAT_ADD(pga_vmstats.pga_nulldmaattr);
3349         } else {
3350                 pfn_t pfnlo = mmu_btop(dma_attr->dma_attr_addr_lo);
3351                 pfn_t pfnhi = mmu_btop(dma_attr->dma_attr_addr_hi);

3353                 /*
3354                  * We can guarantee alignment only for page boundary.
3355                  */
3356                 if (dma_attr->dma_attr_align > MMU_PAGESIZE)
3357                         return (NULL);
```



```
3359                 /* Sanity check the dma_attr */
3360                 if (pfnlo > pfnhi)
3361                         return (NULL);

3363                 n = pfn_2_mtype(pfnlo);
3364                 m = pfn_2_mtype(pfnhi);

3366                 fullrange = ((pfnlo == mnoderanges[n].mnr_pfnlo) &&
3367                     (pfnhi >= mnoderanges[m].mnr_pfnhi));
3368         }
3369         VM_STAT_COND_ADD(fullrange == 0, pga_vmstats.pga_notfullrange);

3371         szc = 0;

3373         /* cylcing thru mtype handled by RANGE0 if n == mtype16m */
3374         if (n == mtype16m) {
3375                 flags |= PGI_MT_RANGE0;
3376                 n = m;
3377         }

3379         /*
3380          * Try local memory node first, but try remote if we can't
3381          * get a page of the right color.
3382          */
3383         LGRP_MNODE_COOKIE_INIT(lgrp_cookie, lgrp, LGRP_SRCH_HIER);
3384         while ((mnode = lgrp_memnode_choose(&lgrp_cookie)) >= 0) {
3385                 /*
3386                  * allocate pages from high pfn to low.
3387                  */
3388                 mtype = m;
3389                 do {
3390                         if (fullrange != 0) {
3391                                 pp = page_get_mnode_freelist(mnode,
3392                                     bin, mtype, szc, flags);
3393                                 if (pp == NULL) {
3394                                         pp = page_get_mnode_cachelist(
3395                                             bin, flags, mnode, mtype);
3396                                 }
3397                         } else {
3398                                 pp = page_get_mnode_anylist(bin, szc,
3399                                     flags, mnode, mtype, dma_attr);
3400                         }
3401                         if (pp != NULL) {
3402                                 VM_STAT_ADD(pga_vmstats.pga_allocok);
3403                                 check_dma(dma_attr, pp, 1);
3404                                 return (pp);
3405                         }
3406                 } while (mtype != n &&
3407                     (mtype = mnoderanges[mtype].mnr_next) != -1);
3408                 if (!local_failed_stat) {
3409                         lgrp_stat_add(lgrp->lgrp_id, LGRP_NUM_ALLOC_FAIL, 1);
3410                         local_failed_stat = 1;
3411                 }
3412         }
3413         VM_STAT_ADD(pga_vmstats.pga_allocfailed);

3415         return (NULL);
3416 }

3418 /*
3419  * page_create_io()
3420  *
3421  * This function is a copy of page_create_va() with an additional
3422  * argument 'mattr' that specifies DMA memory requirements to
3423  * the page list functions. This function is used by the segkmem
3424  * allocator so it is only to create new pages (i.e PG_EXCL is
```

```
 * set).
 *
 * Note: This interface is currently used by x86 PSM only and is
 *       not fully specified so the commitment level is only for
 *       private interface specific to x86. This interface uses PSM
 *       specific page_get_anylist() interface.
 */

#define PAGE_HASH_SEARCH(index, pp, vp, off) { \
        for ((pp) = page_hash[(index)]; (pp); (pp) = (pp)->p_hash) { \
                if ((pp)->p_vnode == (vp) && (pp)->p_offset == (off)) \
                        break; \
        } \
}


page_t *
page_create_io(
        struct vnode    *vp,
        u_offset_t      off,
        uint_t          bytes,
        uint_t          flags,
        struct as       *as,
        caddr_t         vaddr,
        ddi_dma_attr_t  *mattr) /* DMA memory attributes if any */
{
        page_t          *plist = NULL;
        uint_t          plist_len = 0;
        pgcnt_t         npages;
        page_t          *npp = NULL;
        uint_t          pages_req;
        page_t          *pp;
        kmutex_t        *phm = NULL;
        uint_t          index;

        TRACE_4(TR_FAC_VM, TR_PAGE_CREATE_START,
            "page_create_start:vp %p off %llx bytes %u flags %x",
            vp, off, bytes, flags);

        ASSERT((flags & ~(PG_EXCL | PG_WAIT | PG_PHYSCONTIG)) == 0);

        pages_req = npages = mmu_btopr(bytes);

        /*
         * Do the freemem and pcf accounting.
         */
        if (!page_create_wait(npages, flags)) {
                return (NULL);
        }

        TRACE_2(TR_FAC_VM, TR_PAGE_CREATE_SUCCESS,
            "page_create_success:vp %p off %llx", vp, off);

        /*
         * If satisfying this request has left us with too little
         * memory, start the wheels turning to get some back.  The
         * first clause of the test prevents waking up the pageout
         * daemon in situations where it would decide that there's
         * nothing to do.
         */
        if (nscan < desscan && freemem < minfree) {
                TRACE_1(TR_FAC_VM, TR_PAGEOUT_CV_SIGNAL,
                    "pageout_cv_signal:freemem %ld", freemem);
                cv_signal(&proc_pageout->p_cv);
        }
```



```
        if (flags & PG_PHYSCONTIG) {

                plist = page_get_contigpage(&npages, mattr, 1);
                if (plist == NULL) {
                        page_create_putback(npages);
                        return (NULL);
                }

                pp = plist;

                do {
                        if (!page_hashin(pp, vp, off, NULL)) {
                                panic("pg_creat_io: hashin failed %p %p %llx",
                                    (void *)pp, (void *)vp, off);
                        }
                        VM_STAT_ADD(page_create_new);
                        off += MMU_PAGESIZE;
                        PP_CLRFREE(pp);
                        PP_CLRAGED(pp);
                        page_set_props(pp, P_REF);
                        pp = pp->p_next;
                } while (pp != plist);

                if (!npages) {
                        check_dma(mattr, plist, pages_req);
                        return (plist);
                } else {
                        vaddr += (pages_req - npages) << MMU_PAGESHIFT;
                }

                /*
                 * fall-thru:
                 *
                 * page_get_contigpage returns when npages <= sgllen.
                 * Grab the rest of the non-contig pages below from anylist.
                 */
        }

        /*
         * Loop around collecting the requested number of pages.
         * Most of the time, we have to `create' a new page. With
         * this in mind, pull the page off the free list before
         * getting the hash lock.  This will minimize the hash
         * lock hold time, nesting, and the like.  If it turns
         * out we don't need the page, we put it back at the end.
         */
        while (npages--) {
                phm = NULL;

                index = PAGE_HASH_FUNC(vp, off);
top:
                ASSERT(phm == NULL);
                ASSERT(index == PAGE_HASH_FUNC(vp, off));
                ASSERT(MUTEX_NOT_HELD(page_vnode_mutex(vp)));

                if (npp == NULL) {
                        /*
                         * Try to get the page of any color either from
                         * the freelist or from the cache list.
                         */
                        npp = page_get_anylist(vp, off, as, vaddr, MMU_PAGESIZE,
                            flags & ~PG_MATCH_COLOR, mattr, NULL);
                        if (npp == NULL) {
                                if (mattr == NULL) {
                                        /*
                                         * Not looking for a special page;
```

```
3557                                     * panic!
3558                                     */
3559                                    panic("no page found %d", (int)npages);
3560                            }
3561                            /*
3562                             * No page found! This can happen
3563                             * if we are looking for a page
3564                             * within a specific memory range
3565                             * for DMA purposes. If PG_WAIT is
3566                             * specified then we wait for a
3567                             * while and then try again. The
3568                             * wait could be forever if we
3569                             * don't get the page(s) we need.
3570                             *
3571                             * Note: XXX We really need a mechanism
3572                             * to wait for pages in the desired
3573                             * range. For now, we wait for any
3574                             * pages and see if we can use it.
3575                             */
3576
3577                            if ((mattr != NULL) && (flags & PG_WAIT)) {
3578                                    delay(10);
3579                                    goto top;
3580                            }
3581                            goto fail; /* undo accounting stuff */
3582                    }
3583
3584                    if (PP_ISAGED(npp) == 0) {
3585                            /*
3586                             * Since this page came from the
3587                             * cachelist, we must destroy the
3588                             * old vnode association.
3589                             */
3590                            page_hashout(npp, (kmutex_t *)NULL);
3591                    }
3592            }
3593
3594            /*
3595             * We own this page!
3596             */
3597            ASSERT(PAGE_EXCL(npp));
3598            ASSERT(npp->p_vnode == NULL);
3599            ASSERT(!hat_page_is_mapped(npp));
3600            PP_CLRFREE(npp);
3601            PP_CLRAGED(npp);
3602
3603            /*
3604             * Here we have a page in our hot little mits and are
3605             * just waiting to stuff it on the appropriate lists.
3606             * Get the mutex and check to see if it really does
3607             * not exist.
3608             */
3609            phm = PAGE_HASH_MUTEX(index);
3610            mutex_enter(phm);
3611            PAGE_HASH_SEARCH(index, pp, vp, off);
3612            if (pp == NULL) {
3613                    VM_STAT_ADD(page_create_new);
3614                    pp = npp;
3615                    npp = NULL;
3616                    if (!page_hashin(pp, vp, off, phm)) {
3617                            /*
3618                             * Since we hold the page hash mutex and
3619                             * just searched for this page, page_hashin
3620                             * had better not fail.  If it does, that
3621                             * means somethread did not follow the
3622                             * page hash mutex rules.  Panic now and
```



```
3623                                     * get it over with.  As usual, go down
3624                                     * holding all the locks.
3625                                     */
3626                                    ASSERT(MUTEX_HELD(phm));
3627                                    panic("page_create: hashin fail %p %p %llx %p",
3628                                        (void *)pp, (void *)vp, off, (void *)phm);
3629
3630                            }
3631                            ASSERT(MUTEX_HELD(phm));
3632                            mutex_exit(phm);
3633                            phm = NULL;
3634
3635                            /*
3636                             * Hat layer locking need not be done to set
3637                             * the following bits since the page is not hashed
3638                             * and was on the free list (i.e., had no mappings).
3639                             *
3640                             * Set the reference bit to protect
3641                             * against immediate pageout
3642                             *
3643                             * XXXmh modify freelist code to set reference
3644                             * bit so we don't have to do it here.
3645                             */
3646                            page_set_props(pp, P_REF);
3647                    } else {
3648                            ASSERT(MUTEX_HELD(phm));
3649                            mutex_exit(phm);
3650                            phm = NULL;
3651                            /*
3652                             * NOTE: This should not happen for pages associated
3653                             *       with kernel vnode 'kvp'.
3654                             */
3655                            /* XX64 - to debug why this happens! */
3656                            ASSERT(!VN_ISKAS(vp));
3657                            if (VN_ISKAS(vp))
3658                                    cmn_err(CE_NOTE,
3659                                        "page_create: page not expected "
3660                                        "in hash list for kernel vnode - pp 0x%p",
3661                                        (void *)pp);
3662                            VM_STAT_ADD(page_create_exists);
3663                            goto fail;
3664                    }
3665
3666                    /*
3667                     * Got a page!  It is locked.  Acquire the i/o
3668                     * lock since we are going to use the p_next and
3669                     * p_prev fields to link the requested pages together.
3670                     */
3671                    page_io_lock(pp);
3672                    page_add(&plist, pp);
3673                    plist = plist->p_next;
3674                    off += MMU_PAGESIZE;
3675                    vaddr += MMU_PAGESIZE;
3676            }
3677
3678            check_dma(mattr, plist, pages_req);
3679            return (plist);
3680
3681    fail:
3682            if (npp != NULL) {
3683                    /*
3684                     * Did not need this page after all.
3685                     * Put it back on the free list.
3686                     */
3687                    VM_STAT_ADD(page_create_putbacks);
3688                    PP_SETFREE(npp);
```

```
3689                 PP_SETAGED(npp);
3690                 npp->p_offset = (u_offset_t)-1;
3691                 page_list_add(npp, PG_FREE_LIST | PG_LIST_TAIL);
3692                 page_unlock(npp);
3693         }

3695         /*
3696          * Give up the pages we already got.
3697          */
3698         while (plist != NULL) {
3699                 pp = plist;
3700                 page_sub(&plist, pp);
3701                 page_io_unlock(pp);
3702                 plist_len++;
3703                 /*LINTED: constant in conditional ctx*/
3704                 VN_DISPOSE(pp, B_INVAL, 0, kcred);
3705         }

3707         /*
3708          * VN_DISPOSE does freemem accounting for the pages in plist
3709          * by calling page_free. So, we need to undo the pcf accounting
3710          * for only the remaining pages.
3711          */
3712         VM_STAT_ADD(page_create_putbacks);
3713         page_create_putback(pages_req - plist_len);

3715         return (NULL);
3716 }
3717 #endif /* !__xpv */


3720 /*
3721  * Copy the data from the physical page represented by "frompp" to
3722  * that represented by "topp". ppcopy uses CPU->cpu_caddr1 and
3723  * CPU->cpu_caddr2.  It assumes that no one uses either map at interrupt
3724  * level and no one sleeps with an active mapping there.
3725  *
3726  * Note that the ref/mod bits in the page_t's are not affected by
3727  * this operation, hence it is up to the caller to update them appropriately.
3728  */
3729 int
3730 ppcopy(page_t *frompp, page_t *topp)
3731 {
3732         caddr_t         pp_addr1;
3733         caddr_t         pp_addr2;
3734         hat_mempte_t    pte1;
3735         hat_mempte_t    pte2;
3736         kmutex_t        *ppaddr_mutex;
3737         label_t         ljb;
3738         int             ret = 1;

3740         ASSERT_STACK_ALIGNED();
3741         ASSERT(PAGE_LOCKED(frompp));
3742         ASSERT(PAGE_LOCKED(topp));

3744         if (kpm_enable) {
3745                 pp_addr1 = hat_kpm_page2va(frompp, 0);
3746                 pp_addr2 = hat_kpm_page2va(topp, 0);
3747                 kpreempt_disable();
3748         } else {
3749                 /*
3750                  * disable pre-emption so that CPU can't change
3751                  */
3752                 kpreempt_disable();

3754                 pp_addr1 = CPU->cpu_caddr1;
```



```
3755                 pp_addr2 = CPU->cpu_caddr2;
3756                 pte1 = CPU->cpu_caddr1pte;
3757                 pte2 = CPU->cpu_caddr2pte;

3759                 ppaddr_mutex = &CPU->cpu_ppaddr_mutex;
3760                 mutex_enter(ppaddr_mutex);

3762                 hat_mempte_remap(page_pptonum(frompp), pp_addr1, pte1,
3763                     PROT_READ | HAT_STORECACHING_OK, HAT_LOAD_NOCONSIST);
3764                 hat_mempte_remap(page_pptonum(topp), pp_addr2, pte2,
3765                     PROT_READ | PROT_WRITE | HAT_STORECACHING_OK,
3766                     HAT_LOAD_NOCONSIST);
3767         }

3769         if (on_fault(&ljb)) {
3770                 ret = 0;
3771                 goto faulted;
3772         }
3773         if (use_sse_pagecopy)
3774 #ifdef __xpv
3775                 page_copy_no_xmm(pp_addr2, pp_addr1);
3776 #else
3777                 hwblkpagecopy(pp_addr1, pp_addr2);
3778 #endif
3779         else
3780                 bcopy(pp_addr1, pp_addr2, PAGESIZE);

3782         no_fault();
3783 faulted:
3784         if (!kpm_enable) {
3785 #ifdef __xpv
3786                 /*
3787                  * We can't leave unused mappings laying about under the
3788                  * hypervisor, so blow them away.
3789                  */
3790                 if (HYPERVISOR_update_va_mapping((uintptr_t)pp_addr1, 0,
3791                     UVMF_INVLPG | UVMF_LOCAL) < 0)
3792                         panic("HYPERVISOR_update_va_mapping() failed");
3793                 if (HYPERVISOR_update_va_mapping((uintptr_t)pp_addr2, 0,
3794                     UVMF_INVLPG | UVMF_LOCAL) < 0)
3795                         panic("HYPERVISOR_update_va_mapping() failed");
3796 #endif
3797                 mutex_exit(ppaddr_mutex);
3798         }
3799         kpreempt_enable();
3800         return (ret);
3801 }

3803 void
3804 pagezero(page_t *pp, uint_t off, uint_t len)
3805 {
3806         ASSERT(PAGE_LOCKED(pp));
3807         pfnzero(page_pptonum(pp), off, len);
3808 }

3810 /*
3811  * Zero the physical page from off to off + len given by pfn
3812  * without changing the reference and modified bits of page.
3813  *
3814  * We use this using CPU private page address #2, see ppcopy() for more info.
3815  * pfnzero() must not be called at interrupt level.
3816  */
3817 void
3818 pfnzero(pfn_t pfn, uint_t off, uint_t len)
3819 {
3820         caddr_t         pp_addr2;
```

```
3821         hat_mempte_t    pte2;
3822         kmutex_t        *ppaddr_mutex = NULL;

3824         ASSERT_STACK_ALIGNED();
3825         ASSERT(len <= MMU_PAGESIZE);
3826         ASSERT(off <= MMU_PAGESIZE);
3827         ASSERT(off + len <= MMU_PAGESIZE);

3829         if (kpm_enable && !pfn_is_foreign(pfn)) {
3830                 pp_addr2 = hat_kpm_pfn2va(pfn);
3831                 kpreempt_disable();
3832         } else {
3833                 kpreempt_disable();

3835                 pp_addr2 = CPU->cpu_caddr2;
3836                 pte2 = CPU->cpu_caddr2pte;

3838                 ppaddr_mutex = &CPU->cpu_ppaddr_mutex;
3839                 mutex_enter(ppaddr_mutex);

3841                 hat_mempte_remap(pfn, pp_addr2, pte2,
3842                     PROT_READ | PROT_WRITE | HAT_STORECACHING_OK,
3843                     HAT_LOAD_NOCONSIST);
3844         }

3846         if (use_sse_pagezero) {
3847 #ifdef __xpv
3848                 uint_t rem;

3850                 /*
3851                  * zero a byte at a time until properly aligned for
3852                  * block_zero_no_xmm().
3853                  */
3854                 while (!P2NPHASE(off, ((uint_t)BLOCKZEROALIGN)) && len-- > 0)
3855                         pp_addr2[off++] = 0;

3857                 /*
3858                  * Now use faster block_zero_no_xmm() for any range
3859                  * that is properly aligned and sized.
3860                  */
3861                 rem = P2PHASE(len, ((uint_t)BLOCKZEROALIGN));
3862                 len -= rem;
3863                 if (len != 0) {
3864                         block_zero_no_xmm(pp_addr2 + off, len);
3865                         off += len;
3866                 }

3868                 /*
3869                  * zero remainder with byte stores.
3870                  */
3871                 while (rem-- > 0)
3872                         pp_addr2[off++] = 0;
3873 #else
3874                 hwblkclr(pp_addr2 + off, len);
3875 #endif
3876         } else {
3877                 bzero(pp_addr2 + off, len);
3878         }

3880         if (!kpm_enable || pfn_is_foreign(pfn)) {
3881 #ifdef __xpv
3882                 /*
3883                  * On the hypervisor this page might get used for a page
3884                  * table before any intervening change to this mapping,
3885                  * so blow it away.
3886                  */
```



```
3887                 if (HYPERVISOR_update_va_mapping((uintptr_t)pp_addr2, 0,
3888                     UVMF_INVLPG) < 0)
3889                         panic("HYPERVISOR_update_va_mapping() failed");
3890 #endif
3891                 mutex_exit(ppaddr_mutex);
3892         }

3894         kpreempt_enable();
3895 }

3897 /*
3898  * Platform-dependent page scrub call.
3899  */
3900 void
3901 pagescrub(page_t *pp, uint_t off, uint_t len)
3902 {
3903         /*
3904          * For now, we rely on the fact that pagezero() will
3905          * always clear UEs.
3906          */
3907         pagezero(pp, off, len);
3908 }

3910 /*
3911  * set up two private addresses for use on a given CPU for use in ppcopy()
3912  */
3913 void
3914 setup_vaddr_for_ppcopy(struct cpu *cpup)
3915 {
3916         void *addr;
3917         hat_mempte_t pte_pa;

3919         addr = vmem_alloc(heap_arena, mmu_ptob(1), VM_SLEEP);
3920         pte_pa = hat_mempte_setup(addr);
3921         cpup->cpu_caddr1 = addr;
3922         cpup->cpu_caddr1pte = pte_pa;

3924         addr = vmem_alloc(heap_arena, mmu_ptob(1), VM_SLEEP);
3925         pte_pa = hat_mempte_setup(addr);
3926         cpup->cpu_caddr2 = addr;
3927         cpup->cpu_caddr2pte = pte_pa;

3929         mutex_init(&cpup->cpu_ppaddr_mutex, NULL, MUTEX_DEFAULT, NULL);
3930 }

3932 /*
3933  * Undo setup_vaddr_for_ppcopy
3934  */
3935 void
3936 teardown_vaddr_for_ppcopy(struct cpu *cpup)
3937 {
3938         mutex_destroy(&cpup->cpu_ppaddr_mutex);

3940         hat_mempte_release(cpup->cpu_caddr2, cpup->cpu_caddr2pte);
3941         cpup->cpu_caddr2pte = 0;
3942         vmem_free(heap_arena, cpup->cpu_caddr2, mmu_ptob(1));
3943         cpup->cpu_caddr2 = 0;

3945         hat_mempte_release(cpup->cpu_caddr1, cpup->cpu_caddr1pte);
3946         cpup->cpu_caddr1pte = 0;
3947         vmem_free(heap_arena, cpup->cpu_caddr1, mmu_ptob(1));
3948         cpup->cpu_caddr1 = 0;
3949 }

3951 /*
3952  * Function for flushing D-cache when performing module relocations
```

```
3953  * to an alternate mapping.  Unnecessary on Intel / AMD platforms.
3954  */
3955 void
3956 dcache_flushall()
3957 {}

  62 size_t
  63 exec_get_spslew(void)
  64 {
  65         return (0);
  66 }

3959 /*
3960  * Allocate a memory page.  The argument 'seed' can be any pseudo-random
3961  * number to vary where the pages come from.  This is quite a hacked up
3962  * method -- it works for now, but really needs to be fixed up a bit.
3963  *
3964  * We currently use page_create_va() on the kvp with fake offsets,
3965  * segments and virt address.  This is pretty bogus, but was copied from the
3966  * old hat_i86.c code.  A better approach would be to specify either mnode
3967  * random or mnode local and takes a page from whatever color has the MOST
3968  * available - this would have a minimal impact on page coloring.
3969  */
3970 page_t *
3971 page_get_physical(uintptr_t seed)
3972 {
3973         page_t *pp;
3974         u_offset_t offset;
3975         static struct seg tmpseg;
3976         static uintptr_t ctr = 0;

3978         /*
3979          * This code is gross, we really need a simpler page allocator.
3980          *
3981          * We need to assign an offset for the page to call page_create_va()
3982          * To avoid conflicts with other pages, we get creative with the offset.
3983          * For 32 bits, we need an offset > 4Gig
3984          * For 64 bits, need an offset somewhere in the VA hole.
3985          */
3986         offset = seed;
3987         if (offset > kernelbase)
3988                 offset -= kernelbase;
3989         offset <<= MMU_PAGESHIFT;
3990 #if defined(__amd64)
3991         offset += mmu.hole_start;       /* something in VA hole */
3992 #else
3993         offset += 1ULL << 40;   /* something > 4 Gig */
3994 #endif

3996         if (page_resv(1, KM_NOSLEEP) == 0)
3997                 return (NULL);

3999 #ifdef  DEBUG
4000         pp = page_exists(&kvp, offset);
4001         if (pp != NULL)
4002                 panic("page already exists %p", (void *)pp);
4003 #endif

4005         pp = page_create_va(&kvp, offset, MMU_PAGESIZE, PG_EXCL,
4006             &tmpseg, (caddr_t)(ctr += MMU_PAGESIZE));   /* changing VA usage */
4007         if (pp != NULL) {
4008                 page_io_unlock(pp);
4009                 page_downgrade(pp);
4010         }
4011         return (pp);
4012 }
```

```
**********************************************************
    3376 Wed May 27 19:49:36 2015
new/usr/src/uts/intel/os/name_to_sysnum
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
nosys                   0
rexit                   1
psecflags               2
#endif /* ! codereview */
read                    3
write                   4
open                    5
close                   6
linkat                  7
link                    9
unlink                  10
symlinkat               11
chdir                   12
gtime                   13
mknod                   14
chmod                   15
chown                   16
brk                     17
stat                    18
lseek                   19
getpid                  20
mount                   21
readlinkat              22
setuid                  23
getuid                  24
stime                   25
pcsample                26
alarm                   27
fstat                   28
pause                   29
stty                    31
gtty                    32
access                  33
nice                    34
statfs                  35
syssync                 36
kill                    37
fstatfs                 38
setpgrp                 39
uucopystr               40
pipe                    42
times                   43
profil                  44
faccessat               45
setgid                  46
getgid                  47
mknodat                 48
msgsys                  49
sysi86                  50
sysacct                 51
shmsys                  52
semsys                  53
ioctl                   54
uadmin                  55
fchownat                56
utssys                  57
fdsync                  58
exece                   59
```



```
umask                   60
chroot                  61
fcntl                   62
ulimit                  63
renameat                64
unlinkat                65
fstatat                 66
fstatat64               67
openat                  68
openat64                69
tasksys                 70
acctctl                 71
exacctsys               72
getpagesizes            73
rctlsys                 74
sidsys                  75
lwp_park                77
sendfilev               78
rmdir                   79
mkdir                   80
getdents                81
privsys                 82
ucredsys                83
sysfs                   84
getmsg                  85
putmsg                  86
lstat                   88
symlink                 89
readlink                90
setgroups               91
getgroups               92
fchmod                  93
fchown                  94
sigprocmask             95
sigsuspend              96
sigaltstack             97
sigaction               98
sigpending              99
setcontext              100
fchmodat                101
mkdirat                 102
statvfs                 103
fstatvfs                104
getloadavg              105
nfs                     106
waitsys                 107
sigsendsys              108
hrtsys                  109
utimesys                110
sigresend               111
priocntlsys             112
pathconf                113
mincore                 114
mmap                    115
mprotect                116
munmap                  117
fpathconf               118
vfork                   119
fchdir                  120
readv                   121
writev                  122
preadv                  123
pwritev                 124
getrandom               126
mmapobj                 127
setrlimit               128
```

```
125 getrlimit               129
126 lchown                  130
127 memcntl                 131
128 getpmsg                 132
129 putpmsg                 133
130 rename                  134
131 uname                   135
132 setegid                 136
133 sysconfig               137
134 adjtime                 138
135 systeminfo              139
136 sharefs                 140
137 seteuid                 141
138 forksys                 142
139 sigwait                 144
140 lwp_info                145
141 yield                   146
142 lwp_sema_post           148
143 lwp_sema_trywait        149
144 lwp_detach              150
145 corectl                 151
146 modctl                  152
147 fchroot                 153
148 vhangup                 155
149 gettimeofday            156
150 getitimer               157
151 setitimer               158
152 lwp_create              159
153 lwp_exit                160
154 lwp_suspend             161
155 lwp_continue            162
156 lwp_kill                163
157 lwp_self                164
158 lwp_sigmask             165
159 lwp_wait                167
160 lwp_mutex_wakeup        168
161 lwp_cond_wait           170
162 lwp_cond_signal         171
163 lwp_cond_broadcast      172
164 pread                   173
165 pwrite                  174
166 llseek                  175
167 inst_sync               176
168 brandsys                177
169 kaio                    178
170 cpc                     179
171 meminfosys              180
172 rusagesys               181
173 portfs                  182
174 pollsys                 183
175 labelsys                184
176 acl                     185
177 c2audit                 186
178 processor_bind          187
179 processor_info          188
180 p_online                189
181 sigqueue                190
182 clock_gettime           191
183 clock_settime           192
184 clock_getres            193
185 timer_create            194
186 timer_delete            195
187 timer_settime           196
188 timer_gettime           197
189 timer_getoverrun        198
190 nanosleep               199
```



```
191 facl                    200
192 doorfs                  201
193 setreuid                202
194 setregid                203
195 install_utrap           204
196 signotify               205
197 schedctl                206
198 pset                    207
199 resolvepath             209
200 lwp_mutex_timedlock     210
201 lwp_sema_timedwait      211
202 lwp_rwlock_sys          212
203 getdents64              213
204 mmap64                  214
205 stat64                  215
206 lstat64                 216
207 fstat64                 217
208 statvfs64               218
209 fstatvfs64              219
210 setrlimit64             220
211 getrlimit64             221
212 pread64                 222
213 pwrite64                223
214 open64                  225
215 rpcmod                  226
216 zone                    227
217 autofs                  228
218 getcwd                  229
219 so_socket               230
220 so_socketpair           231
221 bind                    232
222 listen                  233
223 accept                  234
224 connect                 235
225 shutdown                236
226 recv                    237
227 recvfrom                238
228 recvmsg                 239
229 send                    240
230 sendmsg                 241
231 sendto                  242
232 getpeername             243
233 getsockname             244
234 getsockopt              245
235 setsockopt              246
236 sockconfig              247
237 ntp_gettime             248
238 ntp_adjtime             249
239 lwp_mutex_unlock        250
240 lwp_mutex_trylock       251
241 lwp_mutex_register      252
242 cladm                   253
243 uucopy                  254
244 umount2                 255
```

```
**********************************************************
    3375 Wed May 27 19:49:37 2015
new/usr/src/uts/sparc/os/name_to_sysnum
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
nosys                   0
rexit                   1
psecflags               2
#endif /* ! codereview */
read                    3
write                   4
open                    5
close                   6
linkat                  7
link                    9
unlink                  10
symlinkat               11
chdir                   12
gtime                   13
mknod                   14
chmod                   15
chown                   16
brk                     17
stat                    18
lseek                   19
getpid                  20
mount                   21
readlinkat              22
setuid                  23
getuid                  24
stime                   25
pcsample                26
alarm                   27
fstat                   28
pause                   29
stty                    31
gtty                    32
access                  33
nice                    34
statfs                  35
syssync                 36
kill                    37
fstatfs                 38
setpgrp                 39
uucopystr               40
pipe                    42
times                   43
profil                  44
faccessat               45
setgid                  46
getgid                  47
mknodat                 48
msgsys                  49
sysacct                 51
shmsys                  52
semsys                  53
ioctl                   54
uadmin                  55
fchownat                56
utssys                  57
fdsync                  58
exece                   59
umask                   60
```



```
chroot                  61
fcntl                   62
ulimit                  63
renameat                64
unlinkat                65
fstatat                 66
fstatat64               67
openat                  68
openat64                69
tasksys                 70
acctctl                 71
exacctsys               72
getpagesizes            73
rctlsys                 74
sidsys                  75
lwp_park                77
sendfilev               78
rmdir                   79
mkdir                   80
getdents                81
privsys                 82
ucredsys                83
sysfs                   84
getmsg                  85
putmsg                  86
lstat                   88
symlink                 89
readlink                90
setgroups               91
getgroups               92
fchmod                  93
fchown                  94
sigprocmask             95
sigsuspend              96
sigaltstack             97
sigaction               98
sigpending              99
setcontext              100
fchmodat                101
mkdirat                 102
statvfs                 103
fstatvfs                104
getloadavg              105
nfs                     106
waitsys                 107
sigsendsys              108
utimesys                110
sigresend               111
priocntlsys             112
pathconf                113
mincore                 114
mmap                    115
mprotect                116
munmap                  117
fpathconf               118
vfork                   119
fchdir                  120
readv                   121
writev                  122
preadv                  123
pwritev                 124
getrandom               126
mmapobj                 127
setrlimit               128
getrlimit               129
lchown                  130
```

```
125 memcntl                 131
126 getpmsg                 132
127 putpmsg                 133
128 rename                  134
129 uname                   135
130 setegid                 136
131 sysconfig               137
132 adjtime                 138
133 systeminfo              139
134 sharefs                 140
135 seteuid                 141
136 forksys                 142
137 sigwait                 144
138 lwp_info                145
139 yield                   146
140 lwp_sema_post           148
141 lwp_sema_trywait        149
142 lwp_detach              150
143 corectl                 151
144 modctl                  152
145 fchroot                 153
146 vhangup                 155
147 gettimeofday            156
148 getitimer               157
149 setitimer               158
150 lwp_create              159
151 lwp_exit                160
152 lwp_suspend             161
153 lwp_continue            162
154 lwp_kill                163
155 lwp_self                164
156 lwp_sigmask             165
157 lwp_wait                167
158 lwp_mutex_wakeup        168
159 lwp_cond_wait           170
160 lwp_cond_signal         171
161 lwp_cond_broadcast      172
162 pread                   173
163 pwrite                  174
164 llseek                  175
165 inst_sync               176
166 brandsys                177
167 kaio                    178
168 cpc                     179
169 meminfosys              180
170 rusagesys               181
171 portfs                  182
172 pollsys                 183
173 labelsys                184
174 acl                     185
175 c2audit                 186
176 processor_bind          187
177 processor_info          188
178 p_online                189
179 sigqueue                190
180 clock_gettime           191
181 clock_settime           192
182 clock_getres            193
183 timer_create            194
184 timer_delete            195
185 timer_settime           196
186 timer_gettime           197
187 timer_getoverrun        198
188 nanosleep               199
189 facl                    200
190 doorfs                  201
```



```
191 setreuid                202
192 setregid                203
193 install_utrap           204
194 signotify               205
195 schedctl                206
196 pset                    207
197 sparc_utrap_install     208
198 resolvepath             209
199 lwp_mutex_timedlock     210
200 lwp_sema_timedwait      211
201 lwp_rwlock_sys          212
202 getdents64              213
203 mmap64                  214
204 stat64                  215
205 lstat64                 216
206 fstat64                 217
207 statvfs64               218
208 fstatvfs64              219
209 setrlimit64             220
210 getrlimit64             221
211 pread64                 222
212 pwrite64                223
213 open64                  225
214 rpcmod                  226
215 zone                    227
216 autofs                  228
217 getcwd                  229
218 so_socket               230
219 so_socketpair           231
220 bind                    232
221 listen                  233
222 accept                  234
223 connect                 235
224 shutdown                236
225 recv                    237
226 recvfrom                238
227 recvmsg                 239
228 send                    240
229 sendmsg                 241
230 sendto                  242
231 getpeername             243
232 getsockname             244
233 getsockopt              245
234 setsockopt              246
235 sockconfig              247
236 ntp_gettime             248
237 ntp_adjtime             249
238 lwp_mutex_unlock        250
239 lwp_mutex_trylock       251
240 lwp_mutex_register      252
241 cladm                   253
242 uucopy                  254
243 umount2                 255
```

```
**********************************************************
   11664 Wed May 27 19:49:37 2015
new/usr/src/uts/sun4u/vm/mach_vm_dep.c
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

/* Copyright (c) 1984, 1986, 1987, 1988, 1989 AT&T */
/*      All Rights Reserved   */

/*
 * Portions of this source code were derived from Berkeley 4.3 BSD
 * under license from the Regents of the University of California.
 */

/*
 * UNIX machine dependent virtual memory support.
 */

#include <sys/vm.h>
#include <sys/exec.h>
#include <sys/cmn_err.h>
#include <sys/cpu_module.h>
#include <sys/cpu.h>
#include <sys/elf_SPARC.h>
#include <sys/archsystm.h>
#include <vm/hat_sfmmu.h>
#include <sys/memnode.h>
#include <sys/mem_cage.h>
#include <vm/vm_dep.h>
#include <sys/random.h>
#endif /* ! codereview */

#if defined(__sparcv9) && defined(SF_ERRATA_57)
caddr_t errata57_limit;
#endif

uint_t page_colors = 0;
uint_t page_colors_mask = 0;
uint_t page_coloring_shift = 0;
```



```
int consistent_coloring;
int update_proc_pgcolorbase_after_fork = 0;

uint_t mmu_page_sizes = DEFAULT_MMU_PAGE_SIZES;
uint_t max_mmu_page_sizes = MMU_PAGE_SIZES;
uint_t mmu_hashcnt = DEFAULT_MAX_HASHCNT;
uint_t max_mmu_hashcnt = MAX_HASHCNT;
size_t mmu_ism_pagesize = DEFAULT_ISM_PAGESIZE;

/*
 * The sun4u hardware mapping sizes which will always be supported are
 * 8K, 64K, 512K and 4M.  If sun4u based machines need to support other
 * page sizes, platform or cpu specific routines need to modify the value.
 * The base pagesize (p_szc == 0) must always be supported by the hardware.
 */
int mmu_exported_pagesize_mask = (1 << TTE8K) | (1 << TTE64K) |
        (1 << TTE512K) | (1 << TTE4M);
uint_t mmu_exported_page_sizes;

uint_t szc_2_userszc[MMU_PAGE_SIZES];
uint_t userszc_2_szc[MMU_PAGE_SIZES];

extern uint_t vac_colors_mask;
extern int vac_shift;

hw_pagesize_t hw_page_array[] = {
        {MMU_PAGESIZE, MMU_PAGESHIFT, 0, MMU_PAGESIZE >> MMU_PAGESHIFT},
        {MMU_PAGESIZE64K, MMU_PAGESHIFT64K, 0,
            MMU_PAGESIZE64K >> MMU_PAGESHIFT},
        {MMU_PAGESIZE512K, MMU_PAGESHIFT512K, 0,
            MMU_PAGESIZE512K >> MMU_PAGESHIFT},
        {MMU_PAGESIZE4M, MMU_PAGESHIFT4M, 0, MMU_PAGESIZE4M >> MMU_PAGESHIFT},
        {MMU_PAGESIZE32M, MMU_PAGESHIFT32M, 0,
            MMU_PAGESIZE32M >> MMU_PAGESHIFT},
        {MMU_PAGESIZE256M, MMU_PAGESHIFT256M, 0,
            MMU_PAGESIZE256M >> MMU_PAGESHIFT},
        {0, 0, 0, 0}
};

/*
 * Maximum page size used to map 64-bit memory segment kmem64_base..kmem64_end
 */
int     max_bootlp_tteszc = TTE4M;

/*
 * use_text_pgsz64k and use_text_pgsz512k allow the user to turn on these
 * additional text page sizes for USIII-IV+ and OPL by changing the default
 * values via /etc/system.
 */
int     use_text_pgsz64K = 0;
int     use_text_pgsz512K = 0;

/*
 * Maximum and default segment size tunables for user heap, stack, private
 * and shared anonymous memory, and user text and initialized data.
 */
size_t max_uheap_lpsize = MMU_PAGESIZE4M;
size_t default_uheap_lpsize = MMU_PAGESIZE;
size_t max_ustack_lpsize = MMU_PAGESIZE4M;
size_t default_ustack_lpsize = MMU_PAGESIZE;
size_t max_privmap_lpsize = MMU_PAGESIZE4M;
size_t max_uidata_lpsize = MMU_PAGESIZE;
size_t max_utext_lpsize = MMU_PAGESIZE4M;
size_t max_shm_lpsize = MMU_PAGESIZE4M;

void
```

```
125 adjust_data_maxlpsize(size_t ismpagesize)
126 {
127         if (max_uheap_lpsize == MMU_PAGESIZE4M) {
128                 max_uheap_lpsize = ismpagesize;
129         }
130         if (max_ustack_lpsize == MMU_PAGESIZE4M) {
131                 max_ustack_lpsize = ismpagesize;
132         }
133         if (max_privmap_lpsize == MMU_PAGESIZE4M) {
134                 max_privmap_lpsize = ismpagesize;
135         }
136         if (max_shm_lpsize == MMU_PAGESIZE4M) {
137                 max_shm_lpsize = ismpagesize;
138         }
139 }

141 /*
142  * The maximum amount a randomized mapping will be slewed.  We should perhaps
143  * arrange things so these tunables can be separate for mmap, mmapobj, and
144  * ld.so
145  */
146 volatile size_t aslr_max_map_skew = 256 * 1024 * 1024; /* 256MB */

148 /*
149 #endif /* ! codereview */
150  * map_addr_proc() is the routine called when the system is to
151  * choose an address for the user.  We will pick an address
152  * range which is just below the current stack limit.  The
153  * algorithm used for cache consistency on machines with virtual
154  * address caches is such that offset 0 in the vnode is always
155  * on a shm_alignment'ed aligned address.  Unfortunately, this
156  * means that vnodes which are demand paged will not be mapped
157  * cache consistently with the executable images.  When the
158  * cache alignment for a given object is inconsistent, the
159  * lower level code must manage the translations so that this
160  * is not seen here (at the cost of efficiency, of course).
161  *
162  * Every mapping will have a redzone of a single page on either side of
163  * the request. This is done to leave one page unmapped between segments.
164  * This is not required, but it's useful for the user because if their
165  * program strays across a segment boundary, it will catch a fault
166  * immediately making debugging a little easier.  Currently the redzone
167  * is mandatory.
168  *
169  *
170  * addrp is a value/result parameter.
171  *      On input it is a hint from the user to be used in a completely
172  *      machine dependent fashion.  For MAP_ALIGN, addrp contains the
173  *      minimal alignment, which must be some "power of two" multiple of
174  *      pagesize.
175  *
176  *      On output it is NULL if no address can be found in the current
177  *      processes address space or else an address that is currently
178  *      not mapped for len bytes with a page of red zone on either side.
179  *      If vacalign is true, then the selected address will obey the alignment
180  *      constraints of a vac machine based on the given off value.
181  */
182 /*ARGSUSED4*/
183 void
184 map_addr_proc(caddr_t *addrp, size_t len, offset_t off, int vacalign,
185     caddr_t userlimit, struct proc *p, uint_t flags)
186 {
187         struct as *as = p->p_as;
188         caddr_t addr;
189         caddr_t base;
190         size_t slen;
```



```
191         uintptr_t align_amount;
192         int allow_largepage_alignment = 1;

194         base = p->p_brkbase;
195         if (userlimit < as->a_userlimit) {
196                 /*
197                  * This happens when a program wants to map something in
198                  * a range that's accessible to a program in a smaller
199                  * address space.  For example, a 64-bit program might
200                  * be calling mmap32(2) to guarantee that the returned
201                  * address is below 4Gbytes.
202                  */
203                 ASSERT(userlimit > base);
204                 slen = userlimit - base;
205         } else {
206                 slen = p->p_usrstack - base -
207                     ((p->p_stk_ctl + PAGEOFFSET) & PAGEMASK);
208         }

210         /* Make len be a multiple of PAGESIZE */
211         len = (len + PAGEOFFSET) & PAGEMASK;

213         /*
214          *  If the request is larger than the size of a particular
215          *  mmu level, then we use that level to map the request.
216          *  But this requires that both the virtual and the physical
217          *  addresses be aligned with respect to that level, so we
218          *  do the virtual bit of nastiness here.
219          *
220          *  For 32-bit processes, only those which have specified
221          *  MAP_ALIGN or an addr will be aligned on a page size > 4MB. Otherwise
222          *  we can potentially waste up to 256MB of the 4G process address
223          *  space just for alignment.
224          */
225         if (p->p_model == DATAMODEL_ILP32 && ((flags & MAP_ALIGN) == 0 ||
226             ((uintptr_t)*addrp) != 0)) {
227                 allow_largepage_alignment = 0;
228         }
229         if ((mmu_page_sizes == max_mmu_page_sizes) &&
230             allow_largepage_alignment &&
231             (len >= MMU_PAGESIZE256M)) {        /* 256MB mappings */
232                 align_amount = MMU_PAGESIZE256M;
233         } else if ((mmu_page_sizes == max_mmu_page_sizes) &&
234             allow_largepage_alignment &&
235             (len >= MMU_PAGESIZE32M)) { /* 32MB mappings */
236                 align_amount = MMU_PAGESIZE32M;
237         } else if (len >= MMU_PAGESIZE4M) {  /* 4MB mappings */
238                 align_amount = MMU_PAGESIZE4M;
239         } else if (len >= MMU_PAGESIZE512K) { /* 512KB mappings */
240                 align_amount = MMU_PAGESIZE512K;
241         } else if (len >= MMU_PAGESIZE64K) { /* 64KB mappings */
242                 align_amount = MMU_PAGESIZE64K;
243         } else  {
244                 /*
245                  * Align virtual addresses on a 64K boundary to ensure
246                  * that ELF shared libraries are mapped with the appropriate
247                  * alignment constraints by the run-time linker.
248                  */
249                 align_amount = ELF_SPARC_MAXPGSZ;
250                 if ((flags & MAP_ALIGN) && ((uintptr_t)*addrp != 0) &&
251                     ((uintptr_t)*addrp < align_amount))
252                         align_amount = (uintptr_t)*addrp;
253         }

255         /*
256          * 64-bit processes require 1024K alignment of ELF shared libraries.
```

```
257          */
258         if (p->p_model == DATAMODEL_LP64)
259                 align_amount = MAX(align_amount, ELF_SPARCV9_MAXPGSZ);
260 #ifdef VAC
261         if (vac && vacalign && (align_amount < shm_alignment))
262                 align_amount = shm_alignment;
263 #endif

265         if ((flags & MAP_ALIGN) && ((uintptr_t)*addrp > align_amount)) {
266                 align_amount = (uintptr_t)*addrp;
267         }

269         ASSERT(ISP2(align_amount));
270         ASSERT(align_amount == 0 || align_amount >= PAGESIZE);

272         /*
273          * Look for a large enough hole starting below the stack limit.
274          * After finding it, use the upper part.
275          */
276         as_purge(as);
277         off = off & (align_amount - 1);

279 #endif /* ! codereview */
280         if (as_gap_aligned(as, len, &base, &slen, AH_HI, NULL, align_amount,
281             PAGESIZE, off) == 0) {
282                 caddr_t as_addr;

284                 /*
285                  * addr is the highest possible address to use since we have
286                  * a PAGESIZE redzone at the beginning and end.
287                  */
288                 addr = base + slen - (PAGESIZE + len);
289                 as_addr = addr;
290                 /*
291                  * Round address DOWN to the alignment amount and
292                  * add the offset in.
293                  * If addr is greater than as_addr, len would not be large
294                  * enough to include the redzone, so we must adjust down
295                  * by the alignment amount.
296                  */
297                 addr = (caddr_t)((uintptr_t)addr & (~(align_amount - 1l)));
298                 addr += (long)off;
299                 if (addr > as_addr) {
300                         addr -= align_amount;
301                 }

303                 /*
304                  * If randomization is requested, slew the allocation
305                  * backwards, within the same gap, by a random amount.
306                  *
307                  * XXX: This will fall over in processes like Java, which
308                  * commonly have a great many small mappings.
309                  */
310                 if (flags & _MAP_RANDOMIZE) {
311                         uint32_t slew;
312                         uint32_t maxslew;

314                         (void) random_get_pseudo_bytes((uint8_t *)&slew,
315                             sizeof (slew));

317                         maxslew = MIN(aslr_max_map_skew, (addr - base));
318                         /*
319                          * Don't allow ASLR to cause mappings to fail below
320                          * because of SF erratum #57
321                          */
322                         maxslew = MIN(maxslew, (addr - errata57_limit));
```



```
324                         slew = slew % MIN(MIN(aslr_max_map_skew, (addr - base)),
325                             addr - errata57_limit);
326                         addr -= P2ALIGN(slew, align_amount);
327                 }

329 #endif /* ! codereview */
330                 ASSERT(addr > base);
331                 ASSERT(addr + len < base + slen);
332                 ASSERT(((uintptr_t)addr & (align_amount - 1l)) ==
333                     ((uintptr_t)(off)));
334                 *addrp = addr;

336 #if defined(SF_ERRATA_57)
337                 if (AS_TYPE_64BIT(as) && addr < errata57_limit) {
338                         *addrp = NULL;
339                 }
340 #endif
341         } else {
342                 *addrp = NULL;  /* no more virtual space */
343         }
344 }

346 /*
347  * Platform-dependent page scrub call.
348  */
349 void
350 pagescrub(page_t *pp, uint_t off, uint_t len)
351 {
352         /*
353          * For now, we rely on the fact that pagezero() will
354          * always clear UEs.
355          */
356         pagezero(pp, off, len);
357 }

359 /*ARGSUSED*/
360 void
361 sync_data_memory(caddr_t va, size_t len)
362 {
363         cpu_flush_ecache();
364 }

366 /*
367  * platform specific large pages for kernel heap support
368  */
369 void
370 mmu_init_kcontext()
371 {
372         extern void set_kcontextreg();

374         if (kcontextreg)
375                 set_kcontextreg();
376 }

378 void
379 contig_mem_init(void)
380 {
381         /* not applicable to sun4u */
382 }

384 /*ARGSUSED*/
385 caddr_t
386 contig_mem_prealloc(caddr_t alloc_base, pgcnt_t npages)
387 {
388         /* not applicable to sun4u */
```

```
389         return (alloc_base);
390 }
```

```
 50 size_t
 51 exec_get_spslew(void)
 52 {
 53         return (0);
 54 }
```

```
**********************************************************
   25221 Wed May 27 19:49:38 2015
new/usr/src/uts/sun4v/vm/mach_vm_dep.c
uts: Allow for address space randomisation.
Randomise the base addresses of shared objects, non-fixed mappings, the
stack and the heap.  Introduce a service, svc:/system/process-security,
and a tool psecflags(1) to control and observe it
**********************************************************
```

```
   1 /*
   2  * CDDL HEADER START
   3  *
   4  * The contents of this file are subject to the terms of the
   5  * Common Development and Distribution License (the "License").
   6  * You may not use this file except in compliance with the License.
   7  *
   8  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
   9  * or http://www.opensolaris.org/os/licensing.
  10  * See the License for the specific language governing permissions
  11  * and limitations under the License.
  12  *
  13  * When distributing Covered Code, include this CDDL HEADER in each
  14  * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  15  * If applicable, add the following below this CDDL HEADER, with the
  16  * fields enclosed by brackets "[]" replaced with your own identifying
  17  * information: Portions Copyright [yyyy] [name of copyright owner]
  18  *
  19  * CDDL HEADER END
  20  */
  21 /*
  22  * Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
  23  * Use is subject to license terms.
  24  */

  26 /* Copyright (c) 1984, 1986, 1987, 1988, 1989 AT&T */
  27 /*      All Rights Reserved   */

  29 /*
  30  * Portions of this source code were derived from Berkeley 4.3 BSD
  31  * under license from the Regents of the University of California.
  32  */

  34 /*
  35  * UNIX machine dependent virtual memory support.
  36  */

  38 #include <sys/vm.h>
  39 #include <sys/exec.h>
  40 #include <sys/cmn_err.h>
  41 #include <sys/cpu_module.h>
  42 #include <sys/cpu.h>
  43 #include <sys/elf_SPARC.h>
  44 #include <sys/archsystm.h>
  45 #include <vm/hat_sfmmu.h>
  46 #include <sys/memnode.h>
  47 #include <sys/mem_cage.h>
  48 #include <vm/vm_dep.h>
  49 #include <sys/error.h>
  50 #include <sys/machsystm.h>
  51 #include <vm/seg_kmem.h>
  52 #include <sys/stack.h>
  53 #include <sys/atomic.h>
  54 #include <sys/promif.h>
  55 #include <sys/random.h>
  56 #endif /* ! codereview */

  58 uint_t page_colors = 0;
```



```
  59 uint_t page_colors_mask = 0;
  60 uint_t page_coloring_shift = 0;
  61 int consistent_coloring;
  62 int update_proc_pgcolorbase_after_fork = 1;

  64 uint_t mmu_page_sizes = MMU_PAGE_SIZES;
  65 uint_t max_mmu_page_sizes = MMU_PAGE_SIZES;
  66 uint_t mmu_hashcnt = MAX_HASHCNT;
  67 uint_t max_mmu_hashcnt = MAX_HASHCNT;
  68 size_t mmu_ism_pagesize = DEFAULT_ISM_PAGESIZE;

  70 /*
  71  * A bitmask of the page sizes supported by hardware based upon szc.
  72  * The base pagesize (p_szc == 0) must always be supported by the hardware.
  73  */
  74 int mmu_exported_pagesize_mask;
  75 uint_t mmu_exported_page_sizes;

  77 uint_t szc_2_userszc[MMU_PAGE_SIZES];
  78 uint_t userszc_2_szc[MMU_PAGE_SIZES];

  80 extern uint_t vac_colors_mask;
  81 extern int vac_shift;

  83 hw_pagesize_t hw_page_array[] = {
  84         {MMU_PAGESIZE, MMU_PAGESHIFT, 0, MMU_PAGESIZE >> MMU_PAGESHIFT},
  85         {MMU_PAGESIZE64K, MMU_PAGESHIFT64K, 0,
  86             MMU_PAGESIZE64K >> MMU_PAGESHIFT},
  87         {MMU_PAGESIZE512K, MMU_PAGESHIFT512K, 0,
  88             MMU_PAGESIZE512K >> MMU_PAGESHIFT},
  89         {MMU_PAGESIZE4M, MMU_PAGESHIFT4M, 0, MMU_PAGESIZE4M >> MMU_PAGESHIFT},
  90         {MMU_PAGESIZE32M, MMU_PAGESHIFT32M, 0,
  91             MMU_PAGESIZE32M >> MMU_PAGESHIFT},
  92         {MMU_PAGESIZE256M, MMU_PAGESHIFT256M, 0,
  93             MMU_PAGESIZE256M >> MMU_PAGESHIFT},
  94         {0, 0, 0, 0}
  95 };

  97 /*
  98  * Maximum page size used to map 64-bit memory segment kmem64_base..kmem64_end
  99  */
 100 int     max_bootlp_tteszc = TTE256M;

 102 /*
 103  * Maximum and default segment size tunables for user heap, stack, private
 104  * and shared anonymous memory, and user text and initialized data.
 105  */
 106 size_t max_uheap_lpsize = MMU_PAGESIZE64K;
 107 size_t default_uheap_lpsize = MMU_PAGESIZE64K;
 108 size_t max_ustack_lpsize = MMU_PAGESIZE64K;
 109 size_t default_ustack_lpsize = MMU_PAGESIZE64K;
 110 size_t max_privmap_lpsize = MMU_PAGESIZE64K;
 111 size_t max_uidata_lpsize = MMU_PAGESIZE64K;
 112 size_t max_utext_lpsize = MMU_PAGESIZE4M;
 113 size_t max_shm_lpsize = MMU_PAGESIZE4M;

 115 /*
 116  * Contiguous memory allocator data structures and variables.
 117  *
 118  * The sun4v kernel must provide a means to allocate physically
 119  * contiguous, non-relocatable memory. The contig_mem_arena
 120  * and contig_mem_slab_arena exist for this purpose. Allocations
 121  * that require physically contiguous non-relocatable memory should
 122  * be made using contig_mem_alloc() or contig_mem_alloc_align()
 123  * which return memory from contig_mem_arena or contig_mem_reloc_arena.
 124  * These arenas import memory from the contig_mem_slab_arena one
```

```
125  * contiguous chunk at a time.
126  *
127  * When importing slabs, an attempt is made to allocate a large page
128  * to use as backing. As a result of the non-relocatable requirement,
129  * slabs are allocated from the kernel cage freelists. If the cage does
130  * not contain any free contiguous chunks large enough to satisfy the
131  * slab allocation, the slab size will be downsized and the operation
132  * retried. Large slab sizes are tried first to minimize cage
133  * fragmentation. If the slab allocation is unsuccessful still, the slab
134  * is allocated from outside the kernel cage. This is undesirable because,
135  * until slabs are freed, it results in non-relocatable chunks scattered
136  * throughout physical memory.
137  *
138  * Allocations from the contig_mem_arena are backed by slabs from the
139  * cage. Allocations from the contig_mem_reloc_arena are backed by
140  * slabs allocated outside the cage. Slabs are left share locked while
141  * in use to prevent non-cage slabs from being relocated.
142  *
143  * Since there is no guarantee that large pages will be available in
144  * the kernel cage, contiguous memory is reserved and added to the
145  * contig_mem_arena at boot time, making it available for later
146  * contiguous memory allocations. This reserve will be used to satisfy
147  * contig_mem allocations first and it is only when the reserve is
148  * completely allocated that new slabs will need to be imported.
149  */
150 static  vmem_t          *contig_mem_slab_arena;
151 static  vmem_t          *contig_mem_arena;
152 static  vmem_t          *contig_mem_reloc_arena;
153 static  kmutex_t        contig_mem_lock;
154 #define CONTIG_MEM_ARENA_QUANTUM        64
155 #define CONTIG_MEM_SLAB_ARENA_QUANTUM   MMU_PAGESIZE64K

157 /* contig_mem_arena import slab sizes, in decreasing size order */
158 static size_t contig_mem_import_sizes[] = {
159         MMU_PAGESIZE4M,
160         MMU_PAGESIZE512K,
161         MMU_PAGESIZE64K
162 };
163 #define NUM_IMPORT_SIZES        \
164         (sizeof (contig_mem_import_sizes) / sizeof (size_t))
165 static size_t contig_mem_import_size_max        = MMU_PAGESIZE4M;
166 size_t contig_mem_slab_size                     = MMU_PAGESIZE4M;

168 /* Boot-time allocated buffer to pre-populate the contig_mem_arena */
169 static size_t contig_mem_prealloc_size;
170 static void *contig_mem_prealloc_buf;

172 /*
173  * The maximum amount a randomized mapping will be slewed.  We should perhaps
174  * arrange things so these tunables can be separate for mmap, mmapobj, and
175  * ld.so
176  */
177 volatile size_t aslr_max_map_skew = 256 * 1024 * 1024; /* 256MB */

179 /*
180 #endif /* ! codereview */
181  * map_addr_proc() is the routine called when the system is to
182  * choose an address for the user.  We will pick an address
183  * range which is just below the current stack limit.  The
184  * algorithm used for cache consistency on machines with virtual
185  * address caches is such that offset 0 in the vnode is always
186  * on a shm_alignment'ed aligned address.  Unfortunately, this
187  * means that vnodes which are demand paged will not be mapped
188  * cache consistently with the executable images.  When the
189  * cache alignment for a given object is inconsistent, the
190  * lower level code must manage the translations so that this
```



```
191  * is not seen here (at the cost of efficiency, of course).
192  *
193  * Every mapping will have a redzone of a single page on either side of
194  * the request. This is done to leave one page unmapped between segments.
195  * This is not required, but it's useful for the user because if their
196  * program strays across a segment boundary, it will catch a fault
197  * immediately making debugging a little easier.  Currently the redzone
198  * is mandatory.
199  *
200  * addrp is a value/result parameter.
201  *      On input it is a hint from the user to be used in a completely
202  *      machine dependent fashion.  For MAP_ALIGN, addrp contains the
203  *      minimal alignment, which must be some "power of two" multiple of
204  *      pagesize.
205  *
206  *      On output it is NULL if no address can be found in the current
207  *      processes address space or else an address that is currently
208  *      not mapped for len bytes with a page of red zone on either side.
209  *      If vacalign is true, then the selected address will obey the alignment
210  *      constraints of a vac machine based on the given off value.
211  */
212 /*ARGSUSED3*/
213 void
214 map_addr_proc(caddr_t *addrp, size_t len, offset_t off, int vacalign,
215     caddr_t userlimit, struct proc *p, uint_t flags)
216 {
217         struct as *as = p->p_as;
218         caddr_t addr;
219         caddr_t base;
220         size_t slen;
221         uintptr_t align_amount;
222         int allow_largepage_alignment = 1;

224         base = p->p_brkbase;
225         if (userlimit < as->a_userlimit) {
226                 /*
227                  * This happens when a program wants to map something in
228                  * a range that's accessible to a program in a smaller
229                  * address space.  For example, a 64-bit program might
230                  * be calling mmap32(2) to guarantee that the returned
231                  * address is below 4Gbytes.
232                  */
233                 ASSERT(userlimit > base);
234                 slen = userlimit - base;
235         } else {
236                 slen = p->p_usrstack - base -
237                     ((p->p_stk_ctl + PAGEOFFSET) & PAGEMASK);
238         }
239         /* Make len be a multiple of PAGESIZE */
240         len = (len + PAGEOFFSET) & PAGEMASK;

242         /*
243          *  If the request is larger than the size of a particular
244          *  mmu level, then we use that level to map the request.
245          *  But this requires that both the virtual and the physical
246          *  addresses be aligned with respect to that level, so we
247          *  do the virtual bit of nastiness here.
248          *
249          *  For 32-bit processes, only those which have specified
250          *  MAP_ALIGN or an addr will be aligned on a page size > 4MB. Otherwise
251          *  we can potentially waste up to 256MB of the 4G process address
252          *  space just for alignment.
253          *
254          * XXXQ Should iterate trough hw_page_array here to catch
255          * all supported pagesizes
256          */
```

```
257         if (p->p_model == DATAMODEL_ILP32 && ((flags & MAP_ALIGN) == 0 ||
258             ((uintptr_t)*addrp) != 0)) {
259                 allow_largepage_alignment = 0;
260         }
261         if ((mmu_page_sizes == max_mmu_page_sizes) &&
262             allow_largepage_alignment &&
263             (len >= MMU_PAGESIZE256M)) {        /* 256MB mappings */
264                 align_amount = MMU_PAGESIZE256M;
265         } else if ((mmu_page_sizes == max_mmu_page_sizes) &&
266             allow_largepage_alignment &&
267             (len >= MMU_PAGESIZE32M)) { /* 32MB mappings */
268                 align_amount = MMU_PAGESIZE32M;
269         } else if (len >= MMU_PAGESIZE4M) {  /* 4MB mappings */
270                 align_amount = MMU_PAGESIZE4M;
271         } else if (len >= MMU_PAGESIZE512K) { /* 512KB mappings */
272                 align_amount = MMU_PAGESIZE512K;
273         } else if (len >= MMU_PAGESIZE64K) { /* 64KB mappings */
274                 align_amount = MMU_PAGESIZE64K;
275         } else  {
276                 /*
277                  * Align virtual addresses on a 64K boundary to ensure
278                  * that ELF shared libraries are mapped with the appropriate
279                  * alignment constraints by the run-time linker.
280                  */
281                 align_amount = ELF_SPARC_MAXPGSZ;
282                 if ((flags & MAP_ALIGN) && ((uintptr_t)*addrp != 0) &&
283                     ((uintptr_t)*addrp < align_amount))
284                         align_amount = (uintptr_t)*addrp;
285         }

287         /*
288          * 64-bit processes require 1024K alignment of ELF shared libraries.
289          */
290         if (p->p_model == DATAMODEL_LP64)
291                 align_amount = MAX(align_amount, ELF_SPARCV9_MAXPGSZ);
292 #ifdef VAC
293         if (vac && vacalign && (align_amount < shm_alignment))
294                 align_amount = shm_alignment;
295 #endif

297         if ((flags & MAP_ALIGN) && ((uintptr_t)*addrp > align_amount)) {
298                 align_amount = (uintptr_t)*addrp;
299         }

301         ASSERT(ISP2(align_amount));
302         ASSERT(align_amount == 0 || align_amount >= PAGESIZE);

304         /*
305          * Look for a large enough hole starting below the stack limit.
306          * After finding it, use the upper part.
307          */
308         as_purge(as);
309         off = off & (align_amount - 1);
310         if (as_gap_aligned(as, len, &base, &slen, AH_HI, NULL, align_amount,
311             PAGESIZE, off) == 0) {
312                 caddr_t as_addr;

314                 /*
315                  * addr is the highest possible address to use since we have
316                  * a PAGESIZE redzone at the beginning and end.
317                  */
318                 addr = base + slen - (PAGESIZE + len);
319                 as_addr = addr;
320                 /*
321                  * Round address DOWN to the alignment amount and
322                  * add the offset in.
```



```
323                  * If addr is greater than as_addr, len would not be large
324                  * enough to include the redzone, so we must adjust down
325                  * by the alignment amount.
326                  */
327                 addr = (caddr_t)((uintptr_t)addr & (~(align_amount - 1l)));
328                 addr += (long)off;
329                 if (addr > as_addr) {
330                         addr -= align_amount;
331                 }

333                 /*
334                  * If randomization is requested, slew the allocation
335                  * backwards, within the same gap, by a random amount.
336                  *
337                  * XXX: This will fall over in processes like Java, which
338                  * commonly have a great many small mappings.
339                  */
340                 if (flags & _MAP_RANDOMIZE) {
341                         uint32_t slew;

343                         (void) random_get_pseudo_bytes((uint8_t *)&slew,
344                             sizeof (slew));

346                         slew = slew % MIN(aslr_max_map_skew, (addr - base));
347                         addr -= P2ALIGN(slew, align_amount);
348                 }

350 #endif /* ! codereview */
351                 ASSERT(addr > base);
352                 ASSERT(addr + len < base + slen);
353                 ASSERT(((uintptr_t)addr & (align_amount - 1l)) ==
354                     ((uintptr_t)(off)));
355                 *addrp = addr;

357         } else {
358                 *addrp = NULL;  /* no more virtual space */
359         }
360 }

362 /*
363  * Platform-dependent page scrub call.
364  * We call hypervisor to scrub the page.
365  */
366 void
367 pagescrub(page_t *pp, uint_t off, uint_t len)
368 {
369         uint64_t pa, length;

371         pa = (uint64_t)(pp->p_pagenum << MMU_PAGESHIFT + off);
372         length = (uint64_t)len;

374         (void) mem_scrub(pa, length);
375 }

377 void
378 sync_data_memory(caddr_t va, size_t len)
379 {
380         /* Call memory sync function */
381         (void) mem_sync(va, len);
382 }

384 size_t
385 mmu_get_kernel_lpsize(size_t lpsize)
386 {
387         extern int mmu_exported_pagesize_mask;
388         uint_t tte;
```

```
390         if (lpsize == 0) {
391                 /* no setting for segkmem_lpsize in /etc/system: use default */
392                 if (mmu_exported_pagesize_mask & (1 << TTE256M)) {
393                         lpsize = MMU_PAGESIZE256M;
394                 } else if (mmu_exported_pagesize_mask & (1 << TTE4M)) {
395                         lpsize = MMU_PAGESIZE4M;
396                 } else if (mmu_exported_pagesize_mask & (1 << TTE64K)) {
397                         lpsize = MMU_PAGESIZE64K;
398                 } else {
399                         lpsize = MMU_PAGESIZE;
400                 }
401
402                 return (lpsize);
403         }
404
405         for (tte = TTE8K; tte <= TTE256M; tte++) {
406
407                 if ((mmu_exported_pagesize_mask & (1 << tte)) == 0)
408                         continue;
409
410                 if (lpsize == TTEBYTES(tte))
411                         return (lpsize);
412         }
413
414         lpsize = TTEBYTES(TTE8K);
415         return (lpsize);
416 }
417
418 void
419 mmu_init_kcontext()
420 {
421 }
422
423 /*ARGSUSED*/
424 void
425 mmu_init_kernel_pgsz(struct hat *hat)
426 {
427 }
428
429 static void *
430 contig_mem_span_alloc(vmem_t *vmp, size_t size, int vmflag)
431 {
432         page_t *ppl;
433         page_t *rootpp;
434         caddr_t addr = NULL;
435         pgcnt_t npages = btopr(size);
436         page_t **ppa;
437         int pgflags;
438         spgcnt_t i = 0;
439
440
441         ASSERT(size <= contig_mem_import_size_max);
442         ASSERT((size & (size - 1)) == 0);
443
444         if ((addr = vmem_xalloc(vmp, size, size, 0, 0,
445             NULL, NULL, vmflag)) == NULL) {
446                 return (NULL);
447         }
448
449         /* The address should be slab-size aligned. */
450         ASSERT(((uintptr_t)addr & (size - 1)) == 0);
451
452         if (page_resv(npages, vmflag & VM_KMFLAGS) == 0) {
453                 vmem_xfree(vmp, addr, size);
454                 return (NULL);
```



```
455         }
456
457         pgflags = PG_EXCL;
458         if (vmflag & VM_NORELOC)
459                 pgflags |= PG_NORELOC;
460
461         ppl = page_create_va_large(&kvp, (u_offset_t)(uintptr_t)addr, size,
462             pgflags, &kvseg, addr, NULL);
463
464         if (ppl == NULL) {
465                 vmem_xfree(vmp, addr, size);
466                 page_unresv(npages);
467                 return (NULL);
468         }
469
470         rootpp = ppl;
471         ppa = kmem_zalloc(npages * sizeof (page_t *), KM_SLEEP);
472         while (ppl != NULL) {
473                 page_t *pp = ppl;
474                 ppa[i++] = pp;
475                 page_sub(&ppl, pp);
476                 ASSERT(page_iolock_assert(pp));
477                 ASSERT(PAGE_EXCL(pp));
478                 page_io_unlock(pp);
479         }
480
481         /*
482          * Load the locked entry.  It's OK to preload the entry into
483          * the TSB since we now support large mappings in the kernel TSB.
484          */
485         hat_memload_array(kas.a_hat, (caddr_t)rootpp->p_offset, size,
486             ppa, (PROT_ALL & ~PROT_USER) | HAT_NOSYNC, HAT_LOAD_LOCK);
487
488         ASSERT(i == page_get_pagecnt(ppa[0]->p_szc));
489         for (--i; i >= 0; --i) {
490                 ASSERT(ppa[i]->p_szc == ppa[0]->p_szc);
491                 ASSERT(page_pptonum(ppa[i]) == page_pptonum(ppa[0]) + i);
492                 (void) page_pp_lock(ppa[i], 0, 1);
493                 /*
494                  * Leave the page share locked. For non-cage pages,
495                  * this would prevent memory DR if it were supported
496                  * on sun4v.
497                  */
498                 page_downgrade(ppa[i]);
499         }
500
501         kmem_free(ppa, npages * sizeof (page_t *));
502         return (addr);
503 }
504
505 /*
506  * Allocates a slab by first trying to use the largest slab size
507  * in contig_mem_import_sizes and then falling back to smaller slab
508  * sizes still large enough for the allocation. The sizep argument
509  * is a pointer to the requested size. When a slab is successfully
510  * allocated, the slab size, which must be >= *sizep and <=
511  * contig_mem_import_size_max, is returned in the *sizep argument.
512  * Returns the virtual address of the new slab.
513  */
514 static void *
515 span_alloc_downsize(vmem_t *vmp, size_t *sizep, size_t align, int vmflag)
516 {
517         int i;
518
519         ASSERT(*sizep <= contig_mem_import_size_max);
```

```
521         for (i = 0; i < NUM_IMPORT_SIZES; i++) {
522                 size_t page_size = contig_mem_import_sizes[i];

524                 /*
525                  * Check that the alignment is also less than the
526                  * import (large page) size. In the case where the
527                  * alignment is larger than the size, a large page
528                  * large enough for the allocation is not necessarily
529                  * physical-address aligned to satisfy the requested
530                  * alignment. Since alignment is required to be a
531                  * power-of-2, any large page >= size && >= align will
532                  * suffice.
533                  */
534                 if (*sizep <= page_size && align <= page_size) {
535                         void *addr;
536                         addr = contig_mem_span_alloc(vmp, page_size, vmflag);
537                         if (addr == NULL)
538                                 continue;
539                         *sizep = page_size;
540                         return (addr);
541                 }
542                 return (NULL);
543         }

545         return (NULL);
546 }

548 static void *
549 contig_mem_span_xalloc(vmem_t *vmp, size_t *sizep, size_t align, int vmflag)
550 {
551         return (span_alloc_downsize(vmp, sizep, align, vmflag | VM_NORELOC));
552 }

554 static void *
555 contig_mem_reloc_span_xalloc(vmem_t *vmp, size_t *sizep, size_t align,
556     int vmflag)
557 {
558         ASSERT((vmflag & VM_NORELOC) == 0);
559         return (span_alloc_downsize(vmp, sizep, align, vmflag));
560 }

562 /*
563  * Free a span, which is always exactly one large page.
564  */
565 static void
566 contig_mem_span_free(vmem_t *vmp, void *inaddr, size_t size)
567 {
568         page_t *pp;
569         caddr_t addr = inaddr;
570         caddr_t eaddr;
571         pgcnt_t npages = btopr(size);
572         page_t *rootpp = NULL;

574         ASSERT(size <= contig_mem_import_size_max);
575         /* All slabs should be size aligned */
576         ASSERT(((uintptr_t)addr & (size - 1)) == 0);

578         hat_unload(kas.a_hat, addr, size, HAT_UNLOAD_UNLOCK);

580         for (eaddr = addr + size; addr < eaddr; addr += PAGESIZE) {
581                 pp = page_find(&kvp, (u_offset_t)(uintptr_t)addr);
582                 if (pp == NULL) {
583                         panic("contig_mem_span_free: page not found");
584                 }
585                 if (!page_tryupgrade(pp)) {
586                         page_unlock(pp);
```



```
587                         pp = page_lookup(&kvp,
588                             (u_offset_t)(uintptr_t)addr, SE_EXCL);
589                         if (pp == NULL)
590                                 panic("contig_mem_span_free: page not found");
591                 }

593                 ASSERT(PAGE_EXCL(pp));
594                 ASSERT(size == page_get_pagesize(pp->p_szc));
595                 ASSERT(rootpp == NULL || rootpp->p_szc == pp->p_szc);
596                 ASSERT(rootpp == NULL || (page_pptonum(rootpp) +
597                     (pgcnt_t)btop(addr - (caddr_t)inaddr) == page_pptonum(pp)));

599                 page_pp_unlock(pp, 0, 1);

601                 if (rootpp == NULL)
602                         rootpp = pp;
603         }
604         page_destroy_pages(rootpp);
605         page_unresv(npages);

607         if (vmp != NULL)
608                 vmem_xfree(vmp, inaddr, size);
609 }

611 static void *
612 contig_vmem_xalloc_aligned_wrapper(vmem_t *vmp, size_t *sizep, size_t align,
613     int vmflag)
614 {
615         ASSERT((align & (align - 1)) == 0);
616         return (vmem_xalloc(vmp, *sizep, align, 0, 0, NULL, NULL, vmflag));
617 }

619 /*
620  * contig_mem_alloc, contig_mem_alloc_align
621  *
622  * Caution: contig_mem_alloc and contig_mem_alloc_align should be
623  * used only when physically contiguous non-relocatable memory is
624  * required. Furthermore, use of these allocation routines should be
625  * minimized as well as should the allocation size. As described in the
626  * contig_mem_arena comment block above, slab allocations fall back to
627  * being outside of the cage. Therefore, overuse of these allocation
628  * routines can lead to non-relocatable large pages being allocated
629  * outside the cage. Such pages prevent the allocation of a larger page
630  * occupying overlapping pages. This can impact performance for
631  * applications that utilize e.g. 256M large pages.
632  */

634 /*
635  * Allocates size aligned contiguous memory up to contig_mem_import_size_max.
636  * Size must be a power of 2.
637  */
638 void *
639 contig_mem_alloc(size_t size)
640 {
641         ASSERT((size & (size - 1)) == 0);
642         return (contig_mem_alloc_align(size, size));
643 }

645 /*
646  * contig_mem_alloc_align allocates real contiguous memory with the
647  * specified alignment up to contig_mem_import_size_max. The alignment must
648  * be a power of 2 and no greater than contig_mem_import_size_max. We assert
649  * the aligment is a power of 2. For non-debug, vmem_xalloc will panic
650  * for non power of 2 alignments.
651  */
652 void *
```

```
653 contig_mem_alloc_align(size_t size, size_t align)
654 {
655         void *buf;

657         ASSERT(size <= contig_mem_import_size_max);
658         ASSERT(align <= contig_mem_import_size_max);
659         ASSERT((align & (align - 1)) == 0);

661         if (align < CONTIG_MEM_ARENA_QUANTUM)
662                 align = CONTIG_MEM_ARENA_QUANTUM;

664         /*
665          * We take the lock here to serialize span allocations.
666          * We do not lose concurrency for the common case, since
667          * allocations that don't require new span allocations
668          * are serialized by vmem_xalloc. Serializing span
669          * allocations also prevents us from trying to allocate
670          * more spans than necessary.
671          */
672         mutex_enter(&contig_mem_lock);

674         buf = vmem_xalloc(contig_mem_arena, size, align, 0, 0,
675             NULL, NULL, VM_NOSLEEP | VM_NORELOC);

677         if ((buf == NULL) && (size <= MMU_PAGESIZE)) {
678                 mutex_exit(&contig_mem_lock);
679                 return (vmem_xalloc(static_alloc_arena, size, align, 0, 0,
680                     NULL, NULL, VM_NOSLEEP));
681         }

683         if (buf == NULL) {
684                 buf = vmem_xalloc(contig_mem_reloc_arena, size, align, 0, 0,
685                     NULL, NULL, VM_NOSLEEP);
686         }

688         mutex_exit(&contig_mem_lock);

690         return (buf);
691 }

693 void
694 contig_mem_free(void *vaddr, size_t size)
695 {
696         if (vmem_contains(contig_mem_arena, vaddr, size)) {
697                 vmem_xfree(contig_mem_arena, vaddr, size);
698         } else if (size > MMU_PAGESIZE) {
699                 vmem_xfree(contig_mem_reloc_arena, vaddr, size);
700         } else {
701                 vmem_xfree(static_alloc_arena, vaddr, size);
702         }
703 }

705 /*
706  * We create a set of stacked vmem arenas to enable us to
707  * allocate large >PAGESIZE chucks of contiguous Real Address space.
708  * The vmem_xcreate interface is used to create the contig_mem_arena
709  * allowing the import routine to downsize the requested slab size
710  * and return a smaller slab.
711  */
712 void
713 contig_mem_init(void)
714 {
715         mutex_init(&contig_mem_lock, NULL, MUTEX_DEFAULT, NULL);

717         contig_mem_slab_arena = vmem_xcreate("contig_mem_slab_arena", NULL, 0,
718             CONTIG_MEM_SLAB_ARENA_QUANTUM, contig_vmem_xalloc_aligned_wrapper,
```



```
719             vmem_xfree, heap_arena, 0, VM_SLEEP | VMC_XALIGN);

721         contig_mem_arena = vmem_xcreate("contig_mem_arena", NULL, 0,
722             CONTIG_MEM_ARENA_QUANTUM, contig_mem_span_xalloc,
723             contig_mem_span_free, contig_mem_slab_arena, 0,
724             VM_SLEEP | VM_BESTFIT | VMC_XALIGN);

726         contig_mem_reloc_arena = vmem_xcreate("contig_mem_reloc_arena", NULL, 0,
727             CONTIG_MEM_ARENA_QUANTUM, contig_mem_reloc_span_xalloc,
728             contig_mem_span_free, contig_mem_slab_arena, 0,
729             VM_SLEEP | VM_BESTFIT | VMC_XALIGN);

731         if (contig_mem_prealloc_buf == NULL || vmem_add(contig_mem_arena,
732             contig_mem_prealloc_buf, contig_mem_prealloc_size, VM_SLEEP)
733             == NULL) {
734                 cmn_err(CE_WARN, "Failed to pre-populate contig_mem_arena");
735         }
736 }

738 /*
739  * In calculating how much memory to pre-allocate, we include a small
740  * amount per-CPU to account for per-CPU buffers in line with measured
741  * values for different size systems. contig_mem_prealloc_base_size is
742  * a cpu specific amount to be pre-allocated before considering per-CPU
743  * requirements and memory size. We always pre-allocate a minimum amount
744  * of memory determined by PREALLOC_MIN. Beyond that, we take the minimum
745  * of contig_mem_prealloc_base_size and a small percentage of physical
746  * memory to prevent allocating too much on smaller systems.
747  * contig_mem_prealloc_base_size is global, allowing for the CPU module
748  * to increase its value if necessary.
749  */
750 #define PREALLOC_PER_CPU        (256 * 1024)            /* 256K */
751 #define PREALLOC_PERCENT        (4)                     /* 4% */
752 #define PREALLOC_MIN            (16 * 1024 * 1024)      /* 16M */
753 size_t contig_mem_prealloc_base_size = 0;

755 /*
756  * Called at boot-time allowing pre-allocation of contiguous memory.
757  * The argument 'alloc_base' is the requested base address for the
758  * allocation and originates in startup_memlist.
759  */
760 caddr_t
761 contig_mem_prealloc(caddr_t alloc_base, pgcnt_t npages)
762 {
763         caddr_t chunkp;

765         contig_mem_prealloc_size = MIN((PREALLOC_PER_CPU * ncpu_guest_max) +
766             contig_mem_prealloc_base_size,
767             (ptob(npages) * PREALLOC_PERCENT) / 100);
768         contig_mem_prealloc_size = MAX(contig_mem_prealloc_size, PREALLOC_MIN);
769         contig_mem_prealloc_size = P2ROUNDUP(contig_mem_prealloc_size,
770             MMU_PAGESIZE4M);

772         alloc_base = (caddr_t)roundup((uintptr_t)alloc_base, MMU_PAGESIZE4M);
773         if (prom_alloc(alloc_base, contig_mem_prealloc_size,
774             MMU_PAGESIZE4M) != alloc_base) {

776                 /*
777                  * Failed.  This may mean the physical memory has holes in it
778                  * and it will be more difficult to get large contiguous
779                  * pieces of memory.  Since we only guarantee contiguous
780                  * pieces of memory contig_mem_import_size_max or smaller,
781                  * loop, getting contig_mem_import_size_max at a time, until
782                  * failure or contig_mem_prealloc_size is reached.
783                  */
784                 for (chunkp = alloc_base;
```

```
785                     (chunkp - alloc_base) < contig_mem_prealloc_size;
786                     chunkp += contig_mem_import_size_max) {

788                         if (prom_alloc(chunkp, contig_mem_import_size_max,
789                             MMU_PAGESIZE4M) != chunkp) {
790                                 break;
791                         }
792                 }
793                 contig_mem_prealloc_size = chunkp - alloc_base;
794                 ASSERT(contig_mem_prealloc_size != 0);
795         }

797         if (contig_mem_prealloc_size != 0) {
798                 contig_mem_prealloc_buf = alloc_base;
799         } else {
800                 contig_mem_prealloc_buf = NULL;
801         }
802         alloc_base += contig_mem_prealloc_size;

804         return (alloc_base);
805 }
```

```
 56 static uint_t sp_color_stride = 16;
 57 static uint_t sp_color_mask = 0x1f;
 58 static uint_t sp_current_color = (uint_t)-1;

 60 size_t
 61 exec_get_spslew(void)
 62 {
 63         uint_t spcolor = atomic_inc_32_nv(&sp_current_color);
 64         return ((size_t)((spcolor & sp_color_mask) * SA(sp_color_stride)));
 65 }
```