```
**********************************************************
   57012 Wed Jun 15 19:31:30 2016
new/usr/src/cmd/sgs/elfdump/common/corenote.c
Code review comments from jeffpc
**********************************************************
______unchanged_portion_omitted_

/*
 * Output information from prsecflags_t structure.
 */
static void
dump_secflags(note_state_t *state, const char *title)
{
        const sl_prsecflags_layout_t *layout = state->ns_arch->prsecflags;
        Conv_secflags_buf_t inv;
        Lword lw;
#endif /* ! codereview */
        Word w;

        indent_enter(state, title, &layout->pr_version);

        w = extract_as_word(state, &layout->pr_version);

        if (w != PRSECFLAGS_VERSION_1) {
                PRINT_DEC(MSG_INTL(MSG_NOTE_BAD_SECFLAGS_VER), pr_version);
                dump_hex_bytes(state->ns_data, state->ns_len, state->ns_indent,
                    4, 3);
        } else {
                PRINT_DEC(MSG_ORIG(MSG_CNOTE_T_PR_VERSION), pr_version);
                lw = extract_as_lword(state, &layout->pr_effective);
                w = extract_as_word(state, &layout->pr_effective);
                print_str(state, MSG_ORIG(MSG_CNOTE_T_PR_EFFECTIVE),
                    conv_prsecflags(lw, 0, &inv));
                    conv_prsecflags(w, 0, &inv));

                lw = extract_as_lword(state, &layout->pr_inherit);
                w = extract_as_word(state, &layout->pr_inherit);
                print_str(state, MSG_ORIG(MSG_CNOTE_T_PR_INHERIT),
                    conv_prsecflags(lw, 0, &inv));
                    conv_prsecflags(w, 0, &inv));

                lw = extract_as_lword(state, &layout->pr_lower);
                w = extract_as_word(state, &layout->pr_lower);
                print_str(state, MSG_ORIG(MSG_CNOTE_T_PR_LOWER),
                    conv_prsecflags(lw, 0, &inv));
                    conv_prsecflags(w, 0, &inv));

                lw = extract_as_lword(state, &layout->pr_upper);
                w = extract_as_word(state, &layout->pr_upper);
                print_str(state, MSG_ORIG(MSG_CNOTE_T_PR_UPPER),
                    conv_prsecflags(lw, 0, &inv));
                    conv_prsecflags(w, 0, &inv));
        }

        indent_exit(state);
}
______unchanged_portion_omitted_
```

****
   12547 Wed Jun 15 19:31:31 2016
new/usr/src/cmd/sgs/elfdump/common/struct_layout_amd64.c
Code review comments from jeffpc
****
______***unchanged_portion_omitted_***

```
static const sl_prsecflags_layout_t prsecflags_layout = {
        { 0,    40,     0,      0 },            /* sizeof (prsecflags_t) */
        { 0,    20,     0,      0 },            /* sizeof (prsecflags_t) */
        { 0,    4,      0,      0 },            /* pr_version */
        { 8,    8,      0,      0 },            /* pr_effective */
        { 16,   8,      0,      0 },            /* pr_inherit */
        { 24,   8,      0,      0 },            /* pr_lower */
        { 32,   8,      0,      0 },            /* pr_upper */
        { 4,    4,      0,      0 },            /* pr_effective */
        { 8,    4,      0,      0 },            /* pr_inherit */
        { 12,   4,      0,      0 },            /* pr_lower */
        { 16,   4,      0,      0 },            /* pr_upper */
};
```
______***unchanged_portion_omitted_***

****
12501 Wed Jun 15 19:31:32 2016
new/usr/src/cmd/sgs/elfdump/common/struct_layout_i386.c
Code review comments from jeffpc
****
______***unchanged_portion_omitted_***

```
static const sl_prsecflags_layout_t prsecflags_layout = {
        { 0,    40,     0,      0 },            /* sizeof (prsecflags_t) */
        { 0,    20,     0,      0 },            /* sizeof (prsecflags_t) */
        { 0,    4,      0,      0 },            /* pr_version */
        { 8,    8,      0,      0 },            /* pr_effective */
        { 16,   8,      0,      0 },            /* pr_inherit */
        { 24,   8,      0,      0 },            /* pr_lower */
        { 32,   8,      0,      0 },            /* pr_upper */
        { 4,    4,      0,      0 },            /* pr_effective */
        { 8,    4,      0,      0 },            /* pr_inherit */
        { 12,   4,      0,      0 },            /* pr_lower */
        { 16,   4,      0,      0 },            /* pr_upper */
};
```

______***unchanged_portion_omitted_***

```
**********************************************************
   87999 Wed Jun 15 19:31:33 2016
new/usr/src/cmd/sgs/libconv/common/corenote.c
Code review comments from jeffpc
**********************************************************
______unchanged_portion_omitted_


#define PROCSECFLGSZ    CONV_EXPN_FIELD_DEF_PREFIX_SIZE +               \
        MSG_ASLR_SIZE           + CONV_EXPN_FIELD_DEF_SEP_SIZE +        \
        MSG_FORBIDNULLMAP_SIZE  + CONV_EXPN_FIELD_DEF_SEP_SIZE +        \
        MSG_NOEXECSTACK_SIZE    + CONV_EXPN_FIELD_DEF_SEP_SIZE +        \
        CONV_INV_BUFSIZE        + CONV_EXPN_FIELD_DEF_SUFFIX_SIZE

/*
 * Ensure that Conv_cnote_pr_secflags_buf_t is large enough:
 *
 * PROCSECFLGSZ is the real minimum size of the buffer required by
 * conv_prsecflags(). However, Conv_cnote_pr_secflags_buf_t uses
 * CONV_CNOTE_PSECFLAGS_FLAG_BUFSIZE to set the buffer size. We do things this
 * way because the definition of PROCSECFLGSZ uses information that is not
 * available in the environment of other programs that include the conv.h
 * header file.
 */
#if (CONV_PRSECFLAGS_BUFSIZE != PROCSECFLGSZ) && !defined(__lint)
#define REPORT_BUFSIZE PROCSECFLGSZ
#include "report_bufsize.h"
#error "CONV_PRSECFLAGS_BUFSIZE does not match PROCSECFLGSZ"
#endif

const char *
conv_prsecflags(secflagset_t flags, Conv_fmt_flags_t fmt_flags,
    Conv_secflags_buf_t *secflags_buf)
{
        /*
         * The values are initialized later, based on position in this array
         */
        static Val_desc vda[] = {
                { 0, MSG_ASLR },
                { 0, MSG_FORBIDNULLMAP },
                { 0, MSG_NOEXECSTACK },
                { 0, 0 }
        };
        static CONV_EXPN_FIELD_ARG conv_arg = {
            NULL, sizeof (secflags_buf->buf)
        };
            NULL, sizeof (secflags_buf->buf) };
        int i;

        for (i = 0; vda[i].v_msg != 0; i++)
        for (i = 0; vda[i].v_msg != 0; i++) {
                vda[i].v_val = secflag_to_bit(i);
        }

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

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

        return ((const char *)secflags_buf->buf);
}
______unchanged_portion_omitted_
```

```
**********************************************************
    4812 Wed Jun 15 19:31:34 2016
new/usr/src/common/secflags/secflags.c
Code review comments from jeffpc
**********************************************************
______unchanged_portion_omitted_

void
secflags_to_str(secflagset_t flags, char *buf, size_t buflen)
{
        struct flagdesc *fd;

        if (buflen >= 1)
                buf[0] = '\0';

        if (flags == 0) {
                (void) strlcpy(buf, "none", buflen);
                return;
        }

        for (fd = flagdescs; fd->name != NULL; fd++) {
                if (secflag_isset(flags, fd->value)) {
                        if (buf[0] != '\0')
                                (void) strlcat(buf, ",", buflen);
                                (void) strlcat(buf, ", ", buflen);
                        (void) strlcat(buf, fd->name, buflen);
                }

                secflag_clear(&flags, fd->value);
        }

        if (flags != 0) {       /* unknown flags */
                char hexbuf[19]; /* 0x%16 PRIx64 */
                char hexbuf[11]; /* 0x%08x */

                (void) snprintf(hexbuf, sizeof (hexbuf), "0x%16" PRIx64, flags);
                (void) snprintf(hexbuf, sizeof (hexbuf), "0x%08x", flags);
                if (buf[0] != '\0')
                        (void) strlcat(buf, ",", buflen);
                        (void) strlcat(buf, ", ", buflen);
                (void) strlcat(buf, hexbuf, buflen);
        }
}
______unchanged_portion_omitted_
```

```
**********************************************************
    3512 Wed Jun 15 19:31:35 2016
new/usr/src/lib/libc/inc/priv_private.h
Code review comments from jeffpc
**********************************************************
/*
 * 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 2007 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

#ifndef _PRIV_PRIVATE_H
#define _PRIV_PRIVATE_H

#pragma ident   "%Z%%M% %I%     %E% SMI"

#endif /* ! codereview */
#include <sys/types.h>
#include <sys/priv.h>
#include <limits.h>

/*
 * Libc private privilege data.
 */

#ifdef __cplusplus
extern "C" {
#endif

#define LOADPRIVDATA(d)         d = __priv_getdata()
#define GETPRIVDATA()           __priv_getdata()
#define LOCKPRIVDATA()          { \
                                        /* Data already allocated */ \
                                        (void) lock_data(); \
                                        (void) refresh_data(); \
                                }
#define UNLOCKPRIVDATA()        unlock_data()
#define WITHPRIVLOCKED(t, b, x) { \
                                        t __result; \
                                        if (lock_data() != 0) \
                                                return (b); \
                                        __result = (x); \
                                        if (__result == (b) && refresh_data()) \
                                                __result = (x); \
                                        unlock_data(); \
                                        return (__result); \
```

```
                                }

/*
 * Privilege mask macros.
 */
#define __NBWRD         (CHAR_BIT * sizeof (priv_chunk_t))
#define privmask(n)     (1 << ((__NBWRD - 1) - ((n) % __NBWRD)))
#define privword(n)     ((n)/__NBWRD)

/*
 * Same as the functions, but for numeric privileges.
 */
#define PRIV_ADDSET(a, p)       ((priv_chunk_t *)(a))[privword(p)] |= \
                                                        privmask(p)
#define PRIV_DELSET(a, p)       ((priv_chunk_t *)(a))[privword(p)] &= \
                                                        ~privmask(p)
#define PRIV_ISMEMBER(a, p)     ((((priv_chunk_t *)(a))[privword(p)] & \
                                                        privmask(p)) != 0)

/*
 * The structure is static except for the setsort, privnames and nprivs
 * field.  The pinfo structure initially has sufficient room and the kernel
 * guarantees no offset changes so we can copy a new structure on top of it.
 * The locking stratgegy is this: we lock it when we need to reference any
 * of the volatile fields.
 */
typedef struct priv_data {
        size_t                  pd_setsize;             /* In bytes */
        int                     pd_nsets, pd_nprivs;
        uint32_t                pd_ucredsize;
        char                    **pd_setnames;
        char                    **pd_privnames;
        int                     *pd_setsort;
        priv_impl_info_t        *pd_pinfo;
        priv_set_t              *pd_basicset;
        priv_set_t              *pd_zoneset;
} priv_data_t;

extern priv_data_t *__priv_getdata(void);
extern priv_data_t *__priv_parse_info(priv_impl_info_t *);
extern void __priv_free_info(priv_data_t *);
extern priv_data_t *privdata;

extern int lock_data(void);
extern boolean_t refresh_data(void);
extern void unlock_data(void);

extern boolean_t __priv_isemptyset(priv_data_t *, const priv_set_t *);
extern boolean_t __priv_isfullset(priv_data_t *, const priv_set_t *);
extern boolean_t __priv_issubset(priv_data_t *, const priv_set_t *,
                                const priv_set_t *);
extern const char *__priv_getbynum(const priv_data_t *, int);

extern int getprivinfo(priv_impl_info_t *, size_t);

extern priv_set_t *priv_basic(void);

#ifdef __cplusplus
}
#endif

#endif /* _PRIV_PRIVATE_H */
```

```
**********************************************************
    4697 Wed Jun 15 19:31:35 2016
new/usr/src/lib/libc/port/sys/sbrk.c
Code review comments from jeffpc
**********************************************************
______unchanged_portion_omitted_

/*
 * _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)
{
        uintptr_t old_brk;
        uintptr_t ret_brk;
        uintptr_t high_brk;
        uintptr_t new_brk;
        intptr_t brk_result;
        int brk_result;

        if (!primary_link_map) {
                errno = ENOTSUP;
                return ((void *)-1);
        }
        if ((low_align & (low_align - 1)) != 0 ||
            (high_align & (high_align - 1)) != 0) {
                errno = EINVAL;
                return ((void *)-1);
        }
        low_align = MAX(low_align, ALIGNSZ);
        high_align = MAX(high_align, ALIGNSZ);

        lmutex_lock(&__sbrk_lock);

        if (_nd == NULL)
        if (_nd == NULL) {
                _nd = (void *)_brk_unlocked(0);
        }

        old_brk = (uintptr_t)BRKALIGN(_nd);
        ret_brk = P2ROUNDUP(old_brk, low_align);
        high_brk = ret_brk + min_size;
        new_brk = P2ROUNDUP(high_brk, high_align);

        /*
         * Check for overflow
         */
        if (ret_brk < old_brk || high_brk < ret_brk || new_brk < high_brk) {
                lmutex_unlock(&__sbrk_lock);
                errno = ENOMEM;
                return ((void *)-1);
        }

        if ((brk_result = _brk_unlocked((void *)new_brk)) == 0)
        if ((brk_result = (int)_brk_unlocked((void *)new_brk)) == 0)
                _nd = (void *)new_brk;
        lmutex_unlock(&__sbrk_lock);

        if (brk_result != 0)
```

```
                return ((void *)-1);

        if (actual_size != NULL)
                *actual_size = (new_brk - ret_brk);
        return ((void *)ret_brk);
}

int
brk(void *new_brk)
{
        intptr_t result;
        int result;

        /*
         * brk(2) will return the current brk if given an argument of 0, so we
         * need to fail it here
         */
        if (new_brk == 0) {
                errno = ENOMEM;
                return (-1);
        }

        if (!primary_link_map) {
                errno = ENOTSUP;
                return (-1);
        }
        /*
         * Need to align this here;  _brk_unlocked won't do it for us.
         */
        new_brk = BRKALIGN(new_brk);

        lmutex_lock(&__sbrk_lock);
        if ((result = _brk_unlocked(new_brk)) == 0)
        if ((result = (int)_brk_unlocked(new_brk)) == 0)
                _nd = new_brk;
        lmutex_unlock(&__sbrk_lock);

        return (result);
}
______unchanged_portion_omitted_
```

```
**********************************************************
   10652 Wed Jun 15 19:31:36 2016
new/usr/src/lib/libscf/common/highlevel.c
Code review comments from jeffpc
**********************************************************
______unchanged_portion_omitted_

int
scf_default_secflags(scf_handle_t *hndl, psecflags_t *flags)
{
#if !defined(NATIVE_BUILD)
        scf_property_t *prop;
        scf_value_t *val;
        const char *flagname;
        int flag;
        struct group_desc *g;
        struct group_desc groups[] = {
                {NULL, "svc:/system/process-security/"
                    ":properties/default"},
                {NULL, "svc:/system/process-security/"
                    ":properties/lower"},
                {NULL, "svc:/system/process-security/"
                    ":properties/upper"},
                {NULL, NULL}
        };

        groups[0].set = &flags->psf_inherit;
        groups[1].set = &flags->psf_lower;
        groups[2].set = &flags->psf_upper;

        /* Ensure sane defaults */
        psecflags_default(flags);

        for (g = groups; g->set != NULL; g++) {
                for (flag = 0; (flagname = secflag_to_str(flag)) != NULL;
                    flag++) {
                        char *pfmri;
                        uint8_t flagval = 0;

                        if ((val = scf_value_create(hndl)) == NULL)
                                return (-1);

                        if ((prop = scf_property_create(hndl)) == NULL) {
                                scf_value_destroy(val);
                                return (-1);
                        }

                        if ((pfmri = uu_msprintf("%s/%s", g->fmri,
                            flagname)) == NULL)
                                uu_die("Allocation failure\n");

                        if (scf_handle_decode_fmri(hndl, pfmri,
                            NULL, NULL, NULL, NULL, prop, NULL) != 0)
                                goto next;

                        if (scf_property_get_value(prop, val) != 0)
                                goto next;

                        (void) scf_value_get_boolean(val, &flagval);

                        if (flagval != 0)
                                secflag_set(g->set, flag);
                        else
                                secflag_clear(g->set, flag);

next:
```

```
                        uu_free(pfmri);
                        scf_value_destroy(val);
                        scf_property_destroy(prop);
                }
        }

        if (!psecflags_validate(flags))
                return (-1);

        return (0);
#else
        assert(0);
        abort();
#endif /* ! codereview */
#endif /* !NATIVE_BUILD */
        return (0);
}
______unchanged_portion_omitted_
```

```
**********************************************************
   59567 Wed Jun 15 19:31:37 2016
new/usr/src/man/man1/ld.1
Code review comments from jeffpc
**********************************************************
'\" te
.\" Copyright 1989 AT&T
.\" Copyright (c) 2009, Sun Microsystems, Inc. All Rights Reserved
.\" Copyright (c) 2012, Joyent, Inc. All Rights Reserved
.\" The contents of this file are subject to the terms of the Common Development
.\"  See the License for the specific language governing permissions and limitat
.\" the fields enclosed by brackets "[]" replaced with your own identifying info
.TH LD 1 "Jun 6, 2016"
.TH LD 1 "Sep 10, 2013"
.SH NAME
ld \- link-editor for object files
.SH SYNOPSIS
.LP
.nf
\fBld\fR [\fB-32\fR | \fB-64\fR] [\fB-a\fR | \fB-r\fR] [\fB-b\fR] [\fB-B\fRdirec
[\fB-B\fR dynamic | static] [\fB-B\fR eliminate] [\fB-B\fR group] [\fB-B\fR loca
[\fB-B\fR reduce] [\fB-B\fR symbolic] [\fB-c\fR \fIname\fR] [\fB-C\fR] [\fB-d\fR
[\fB-D\fR \fItoken\fR,...] [\fB-e\fR \fIepsym\fR] [\fB-f\fR \fIname\fR | \fB-F\f
[\fB-i\fR] [\fB-I\fR \fIname\fR] [\fB-l\fR \fIx\fR] [\fB-L\fR \fIpath\fR] [\fB-m
[\fB-N\fR \fIstring\fR] [\fB-o\fR \fIoutfile\fR] [\fB-p\fR \fIauditlib\fR] [\fB-
[\fB-Q\fR y | n] [\fB-R\fR \fIpath\fR] [\fB-s\fR] [\fB-S\fR \fIsupportlib\fR] [\
[\fB-u\fR \fIsymname\fR] [\fB-V\fR] [\fB-Y P\fR\fI,dirlist\fR] [\fB-z\fR absexec
[\fB-z\fR allextract | defaultextract | weakextract ] [\fB-z\fR altexec64]
[\fB-z\fR aslr[=\fIstate\fR]] [\fB-z\fR assert-deflib] [ \fB-z\fR assert-deflib=
[\fB-z\fR combreloc | nocombreloc ] [\fB-z\fR defs | nodefs]
[\fB-z\fR direct | nodirect] [\fB-z\fR endfiltee]
[\fB-z\fR fatal-warnings | nofatal-warnings ] [\fB-z\fR finiarray=\fIfunction\fR
[\fB-z\fR globalaudit] [\fB-z\fR groupperm | nogroupperm]
[\fB-z\fR guidance[=\fIid1\fR,\fIid2\fR...] [\fB-z\fR help ]
[\fB-z\fR ignore | record] [\fB-z\fR initarray=\fIfunction\fR] [\fB-z\fR initfir
[\fB-z\fR interpose] [\fB-z\fR lazyload | nolazyload]
[\fB-z\fR ld32=\fIarg1\fR,\fIarg2\fR,...] [\fB-z\fR ld64=\fIarg1\fR,\fIarg2\fR,.
[\fB-z\fR loadfltr] [\fB-z\fR muldefs] [\fB-z\fR nocompstrtab] [\fB-z\fR nodefau
[\fB-z\fR nodelete] [\fB-z\fR nodlopen] [\fB-z\fR nodump] [\fB-z\fR noldynsym]
[\fB-z\fR nopartial] [\fB-z\fR noversion] [\fB-z\fR now] [\fB-z\fR origin]
[\fB-z\fR preinitarray=\fIfunction\fR] [\fB-z\fR redlocsym] [\fB-z\fR relaxreloc
[\fB-z\fR rescan-now] [\fB-z\fR recan] [\fB-z\fR rescan-start \fI\&...\fR \fB-z\
[\fB-z\fR target=sparc|x86] [\fB-z\fR text | textwarn | textoff]
[\fB-z\fR verbose] [\fB-z\fR wrap=\fIsymbol\fR] \fIfilename\fR...
.fi

.SH DESCRIPTION
.LP
The link-editor, \fBld\fR, combines relocatable object files by resolving
symbol references to symbol definitions, together with performing relocations.
\fBld\fR operates in two modes, static or dynamic, as governed by the \fB-d\fR
option. In all cases, the output of \fBld\fR is left in the file \fBa.out\fR by
default. See NOTES.
.sp
.LP
In dynamic mode, \fB-dy\fR, the default, relocatable object files that are
provided as arguments are combined to produce an executable object file. This
file is linked at execution with any shared object files that are provided as
arguments. If the \fB-G\fR option is specified, relocatable object files are
combined to produce a shared object. Without the \fB-G\fR option, a dynamic
executable is created.
.sp
.LP
In static mode, \fB-dn\fR, relocatable object files that are provided as
arguments are combined to produce a static executable file. If the \fB-r\fR
option is specified, relocatable object files are combined to produce one
```

```
relocatable object file. See \fBStatic Executables\fR.
.sp
.LP
Dynamic linking is the most common model for combining relocatable objects, and
the eventual creation of processes within Solaris. This environment tightly
couples the work of the link-editor and the runtime linker, \fBld.so.1\fR(1).
Both of these utilities, together with their related technologies and
utilities, are extensively documented in the \fILinker and Libraries Guide\fR.
.sp
.LP
If any argument is a library, \fBld\fR by default searches the library exactly
once at the point the library is encountered on the argument list. The library
can be either a shared object or relocatable archive. See \fBar.h\fR(3HEAD)).
.sp
.LP
A shared object consists of an indivisible, whole unit that has been generated
by a previous link-edit of one or more input files. When the link-editor
processes a shared object, the entire contents of the shared object become a
logical part of the resulting output file image. The shared object is not
physically copied during the link-edit as its actual inclusion is deferred
until process execution. This logical inclusion means that all symbol entries
defined in the shared object are made available to the link-editing process.
See Chapter 4, \fIShared Objects,\fR in \fILinker and Libraries Guide\fR
.sp
.LP
For an archive library, \fBld\fR loads only those routines that define an
unresolved external reference. \fBld\fR searches the symbol table of the
archive library sequentially to resolve external references that can be
satisfied by library members. This search is repeated until no external
references can be resolved by the archive. Thus, the order of members in the
library is functionally unimportant, unless multiple library members exist that
define the same external symbol. Archive libraries that have interdependencies
can require multiple command line definitions, or the use of one of the
\fB-z\fR \fBrescan\fR options. See \fIArchive Processing\fR in \fILinker and
Libraries Guide\fR.
.sp
.LP
\fBld\fR is a cross link-editor, able to link 32-bit objects or 64-bit objects,
for Sparc or x86 targets. \fBld\fR uses the \fBELF\fR class and machine type of
the first relocatable object on the command line to govern the mode in which to
operate. The mixing of 32-bit objects and 64-bit objects is not permitted.
Similarly, only objects of a single machine type are allowed. See the
\fB-32\fR, \fB-64\fR and \fB-z target\fR options, and the \fBLD_NOEXEC_64\fR
environment variable.
.SS "Static Executables"
.LP
The creation of static executables has been discouraged for many releases. In
fact, 64-bit system archive libraries have never been provided. Because a
static executable is built against system archive libraries, the executable
contains system implementation details. This self-containment has a number of
drawbacks.
.RS +4
.TP
.ie t \(bu
.el o
The executable is immune to the benefits of system patches delivered as shared
objects. The executable therefore, must be rebuilt to take advantage of many
system improvements.
.RE
.RS +4
.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
.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
.RS 4n
Causes any global symbols, not assigned to a version definition, to be
eliminated from the symbol table. Version definitions can be supplied by means
of a \fBmapfile\fR to indicate the global symbols that should remain visible in
the generated object. This option achieves the same symbol elimination as the
\fIauto-elimination\fR directive that is available as part of a \fBmapfile\fR
```

```
version definition. This option can be useful when combining versioned and
non-versioned relocatable objects. See also the \fB-B\fR \fBlocal\fR option and
the \fB-B\fR \fBreduce\fR option. See \fIDefining Additional Symbols with a
mapfile\fR in \fILinker and Libraries Guide\fR.
.RE

.sp
.ne 2
.na
\fB\fB-B\fR \fBgroup\fR\fR
.ad
.sp .6
.RS 4n
Establishes a shared object and its dependencies as a group. Objects within the
group are bound to other members of the group at runtime. This mode is similar
to adding the object to the process by using \fBdlopen\fR(3C) with the
\fBRTLD_GROUP\fR mode. An object that has an explicit dependency on a object
identified as a group, becomes a member of the group.
.sp
As the group must be self contained, use of the \fB-B\fR \fBgroup\fR option
also asserts the \fB-z\fR \fBdefs\fR option.
.RE

.sp
.ne 2
.na
\fB\fB-B\fR \fBlocal\fR\fR
.ad
.sp .6
.RS 4n
Causes any global symbols, not assigned to a version definition, to be reduced
to local. Version definitions can be supplied by means of a \fBmapfile\fR to
indicate the global symbols that should remain visible in the generated object.
This option achieves the same symbol reduction as the \fIauto-reduction\fR
directive that is available as part of a \fBmapfile\fR version definition. This
option can be useful when combining versioned and non-versioned relocatable
objects. See also the \fB-B\fR \fBeliminate\fR option and the \fB-B\fR
\fBreduce\fR option. See \fIDefining Additional Symbols with a mapfile\fR in
\fILinker and Libraries Guide\fR.
.RE

.sp
.ne 2
.na
\fB\fB-B\fR \fBreduce\fR\fR
.ad
.sp .6
.RS 4n
When generating a relocatable object, causes the reduction of symbolic
information defined by any version definitions. Version definitions can be
supplied by means of a \fBmapfile\fR to indicate the global symbols that should
remain visible in the generated object. By default, when a relocatable object
is generated, version definitions are only recorded in the output image. The
actual reduction of symbolic information is carried out when the object is used
in the construction of a dynamic executable or shared object. The \fB-B\fR
\fBreduce\fR option is applied automatically when a dynamic executable or
shared object is created.
.RE

.sp
.ne 2
.na
\fB\fB-B\fR \fBsymbolic\fR\fR
.ad
.sp .6
.RS 4n
```

```
In dynamic mode only. When building a shared object, binds references to global
symbols to their definitions, if available, within the object. Normally,
references to global symbols within shared objects are not bound until runtime,
even if definitions are available. This model allows definitions of the same
symbol in an executable or other shared object to override the object’s own
definition. \fBld\fR issues warnings for undefined symbols unless \fB-z\fR
\fBdefs\fR overrides.
.sp
The \fB-B\fR \fBsymbolic\fR option is intended for specialized dynamic objects
and is not recommended for general use. To reduce the runtime relocation
processing that is required an object, the creation of a version definition is
recommended.
.RE

.sp
.ne 2
.na
\fB\fB-c\fR \fIname\fR\fR
.ad
.sp .6
.RS 4n
Records the configuration file \fIname\fR for use at runtime. Configuration
files can be employed to alter default search paths, provide a directory cache,
together with providing alternative object dependencies. See \fBcrle\fR(1).
.RE

.sp
.ne 2
.na
\fB\fB-C\fR\fR
.ad
.sp .6
.RS 4n
Demangles C++ symbol names displayed in diagnostic messages.
.RE

.sp
.ne 2
.na
\fB\fB-d\fR \fBy\fR | \fBn\fR\fR
.ad
.sp .6
.RS 4n
When \fB-d\fR \fBy\fR, the default, is specified, \fBld\fR uses dynamic
linking. When \fB-d\fR \fBn\fR is specified, \fBld\fR uses static linking. See
\fBStatic Executables\fR under DESCRIPTION, and \fB-B\fR
\fBdynamic\fR|\fBstatic\fR.
.RE

.sp
.ne 2
.na
\fB\fB-D\fR \fItoken\fR,...\fR
.ad
.sp .6
.RS 4n
Prints debugging information as specified by each \fItoken\fR, to the standard
error. The special token \fBhelp\fR indicates the full list of tokens
available. See \fIDebugging Aids\fR in \fILinker and Libraries Guide\fR.
.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
command. \fB-Q\fR \fBn\fR suppresses version identification. \fB\&.comment\fR
sections can be manipulated by the \fBmcs\fR(1) utility.
.RE

.sp
.ne 2
```

```
.na
\fB\fB-r\fR\fR
.ad
.br
.na
\fB\fB--relocatable\fR\fR
.ad
.sp .6
.RS 4n
Combines relocatable object files to produce one relocatable object file.
\fBld\fR does not complain about unresolved references. This option cannot be
used with the \fB-a\fR option.
.RE

.sp
.ne 2
.na
\fB\fB-R\fR \fIpath\fR\fR
.ad
.br
.na
\fB\fB-rpath\fR \fIpath\fR\fR
.ad
.sp .6
.RS 4n
A colon-separated list of directories used to specify library search
directories to the runtime linker. If present and not NULL, the path is
recorded in the output object file and passed to the runtime linker. Multiple
instances of this option are concatenated together with each \fIpath\fR
separated by a colon. See \fIDirectories Searched by the Runtime Linker\fR in
\fILinker and Libraries Guide\fR.
.sp
The use of a runpath within an associated object is preferable to setting
global search paths such as through the \fBLD_LIBRARY_PATH\fR environment
variable. Only the runpaths that are necessary to find the objects dependencies
should be recorded. \fBldd\fR(1) can also be used to discover unused runpaths
in dynamic objects, when used with the \fB-U\fR option.
.sp
Various tokens can also be supplied with a runpath that provide a flexible
means of identifying system capabilities or an objects location. See Appendix
C, \fIEstablishing Dependencies with Dynamic String Tokens,\fR in \fILinker and
Libraries Guide\fR. The \fB$ORIGIN\fR token is especially useful in allowing
dynamic objects to be relocated to different locations in the file system.
.RE

.sp
.ne 2
.na
\fB\fB-s\fR\fR
.ad
.br
.na
\fB\fB--strip-all\fR\fR
.ad
.sp .6
.RS 4n
Strips symbolic information from the output file. Any debugging information,
that is, \fB\&.line\fR, \fB\&.debug*\fR, and \fB\&.stab*\fR sections, and their
associated relocation entries are removed. Except for relocatable files, a
symbol table \fBSHT_SYMTAB\fR and its associated string table section are not
created in the output object file. The elimination of a \fBSHT_SYMTAB\fR symbol
table can reduce the \fB\&.stab*\fR debugging information that is generated
using the compiler drivers \fB-g\fR option. See the \fB-z\fR \fBredlocsym\fR
and \fB-z\fR \fBnoldynsym\fR options.
.RE
```

```
.sp
.ne 2
.na
\fB\fB-S\fR \fIsupportlib\fR\fR
.ad
.sp .6
.RS 4n
The shared object \fIsupportlib\fR is loaded with \fBld\fR and given
information regarding the linking process. Shared objects that are defined by
using the \fB-S\fR option can also be supplied using the \fBSGS_SUPPORT\fR
environment variable. See \fILink-Editor Support Interface\fR in \fILinker and
Libraries Guide\fR.
.RE

.sp
.ne 2
.na
\fB\fB-t\fR\fR
.ad
.sp .6
.RS 4n
Turns off the warning for multiply-defined symbols that have different sizes or
different alignments.
.RE

.sp
.ne 2
.na
\fB\fB-u\fR \fIsymname\fR\fR
.ad
.br
.na
\fB\fB--undefined\fR \fIsymname\fR\fR
.ad
.sp .6
.RS 4n
Enters \fIsymname\fR as an undefined symbol in the symbol table. This option is
useful for loading entirely from an archive library. In this instance, an
unresolved reference is needed to force the loading of the first routine. The
placement of this option on the command line is significant. This option must
be placed before the library that defines the symbol. See \fIDefining
Additional Symbols with the u option\fR in \fILinker and Libraries Guide\fR.
.RE

.sp
.ne 2
.na
\fB\fB-V\fR\fR
.ad
.br
.na
\fB\fB--version\fR\fR
.ad
.sp .6
.RS 4n
Outputs a message giving information about the version of \fBld\fR being used.
.RE

.sp
.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 simply use the settings inherited from its
environment should not use this flag at all.
.RE

.sp
.ne 2
.na
\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));
}
```

______***unchanged_portion_omitted_***

```
**********************************************************
    5791 Wed Jun 15 19:31:39 2016
new/usr/src/man/man1/psecflags.1
Code review comments from jeffpc
**********************************************************
'\" 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.
.\"
.\" Copyright 2015, Richard Lowe.
.\"
.TH "PSECFLAGS" "1" "June 6, 2016"
.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 \fIspec\fR \fI-e\fR \fIcommand\fR \
[\fIarg\fR]...
\fB/usr/bin/psecflags\fR \fI-s\fR \fIspec\fR \fI-e\fR \fIcommand\fR
        [\fIarg\fR]...
.fi
.LP
.nf
\fB/usr/bin/psecflags\fR \fI-s\fR \fIspec\fR [\fI-i\fR \fIidtype\fR] \
\fIid\fR ...
\fB/usr/bin/psecflags\fR \fI-s\fR \fIspec\fR [\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', the inheritable
set by '\fBI\fR', the lower set by '\fBL\fR', and the upper set by '\fBU\fR'.
.P
The fourth invocation lists the supported process security-flags, documented
in \fBsecurity-flags\fR(5).

.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, described in
\fBsecurity-flags\fR(5).
.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 of a comma-separated list of
flags, optionally preceded by a '-' or '!'. Where '-' and '!' indicate that the
given flag should be removed from the specification.  The pseudo-flags "all",
"none" and "current" are supported, to indicate that all flags, no flags, or
the current set of flags (respectively) are to be included.
.P
By default, the inheritable flags are changed.  You may optionally specify the
set to change using their single-letter identifiers and an equals sign.
.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
        L:      none
        U:      aslr,forbidnullmap,noexecstack
        U:      aslr, forbidnullmap, noexecstack
.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 current,aslr -e /bin/sh\fR
$ psecflags $$
100724: -sh
        E:      none
        I:      aslr
        L:      none
        U:      aslr,forbidnullmap,noexecstack
        U:      aslr, forbidnullmap, noexecstack
.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 current,-aslr -i uid bob\fR
.fi
```

```
.in -2

.LP
\fBExample 4\fR Add the aslr flag to the lower set, so that all future
child processes must have this flag set.
.sp
.in +2
.nf
example# \fBpsecflags -s L=current,aslr $$\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"
.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)
```

```
**********************************************************
   44021 Wed Jun 15 19:31:40 2016
new/usr/src/man/man1m/zonecfg.1m
Code review comments from jeffpc
**********************************************************
'\" te
.\" Copyright (c) 2004, 2009 Sun Microsystems, Inc. All Rights Reserved.
.\" Copyright 2013 Joyent, Inc. All Rights Reserved.
.\" The contents of this file are subject to the terms of the Common Development
.\" See the License for the specific language governing permissions and limitati
.\" fields enclosed by brackets "[]" replaced with your own identifying informat
.TH ZONECFG 1M "Jun 6, 2016"
.TH ZONECFG 1M "Feb 28, 2014"
.SH NAME
zonecfg \- set up zone configuration
.SH SYNOPSIS
.LP
.nf
\fBzonecfg\fR \fB-z\fR \fIzonename\fR
.fi

.LP
.nf
\fBzonecfg\fR \fB-z\fR \fIzonename\fR \fIsubcommand\fR
.fi

.LP
.nf
\fBzonecfg\fR \fB-z\fR \fIzonename\fR \fB-f\fR \fIcommand_file\fR
.fi

.LP
.nf
\fBzonecfg\fR help
.fi

.SH DESCRIPTION
.LP
The \fBzonecfg\fR utility creates and modifies the configuration of a zone.
Zone configuration consists of a number of resources and properties.
.sp
.LP
To simplify the user interface, \fBzonecfg\fR uses the concept of a scope. The
default scope is global.
.sp
.LP
The following synopsis of the \fBzonecfg\fR command is for interactive usage:
.sp
.in +2
.nf
zonecfg \fB-z\fR \fIzonename subcommand\fR
.fi
.in -2
.sp

.sp
.LP
Parameters changed through \fBzonecfg\fR do not affect a running zone. The zone
must be rebooted for the changes to take effect.
.sp
.LP
In addition to creating and modifying a zone, the \fBzonecfg\fR utility can
also be used to persistently specify the resource management settings for the
global zone.
.sp
.LP
```

```
In the following text, "rctl" is used as an abbreviation for "resource
control". See \fBresource_controls\fR(5).
.sp
.LP
Every zone is configured with an associated brand. The brand determines the
user-level environment used within the zone, as well as various behaviors for
the zone when it is installed, boots, or is shutdown. Once a zone has been
installed the brand cannot be changed. The default brand is determined by the
installed distribution in the global zone. Some brands do not support all of
the \fBzonecfg\fR properties and resources. See the brand-specific man page for
more details on each brand. For an overview of brands, see the \fBbrands\fR(5)
man page.
.SS "Resources"
.LP
The following resource types are supported:
.sp
.ne 2
.na
\fB\fBattr\fR\fR
.ad
.sp .6
.RS 4n
Generic attribute.
.RE

.sp
.ne 2
.na
\fB\fBcapped-cpu\fR\fR
.ad
.sp .6
.RS 4n
Limits for CPU usage.
.RE

.sp
.ne 2
.na
\fB\fBcapped-memory\fR\fR
.ad
.sp .6
.RS 4n
Limits for physical, swap, and locked memory.
.RE

.sp
.ne 2
.na
\fB\fBdataset\fR\fR
.ad
.sp .6
.RS 4n
\fBZFS\fR dataset.
.RE

.sp
.ne 2
.na
\fB\fBdedicated-cpu\fR\fR
.ad
.sp .6
.RS 4n
Subset of the system's processors dedicated to this zone while it is running.
.RE

.sp
```

```
.ne 2
.na
\fB\fBdevice\fR\fR
.ad
.sp .6
.RS 4n
Device.
.RE

.sp
.ne 2
.na
\fB\fBfs\fR\fR
.ad
.sp .6
.RS 4n
file-system
.RE

.sp
.ne 2
.na
\fB\fBnet\fR\fR
.ad
.sp .6
.RS 4n
Network interface.
.RE

.sp
.ne 2
.na
\fB\fBrctl\fR\fR
.ad
.sp .6
.RS 4n
Resource control.
.RE

.sp
.ne 2
.na
\fB\fBsecurity-flags\fR\fR
.ad
.sp .6
.RS 4n
Process security flag settings.
.RE

.SS "Properties"
.LP
Each resource type has one or more properties. There are also some global
properties, that is, properties of the configuration as a whole, rather than of
some particular resource.
.sp
.LP
The following properties are supported:
.sp
.ne 2
.na
\fB(global)\fR
.ad
.sp .6
.RS 4n
\fBzonename\fR
.RE
```

```
.sp
.ne 2
.na
\fB(global)\fR
.ad
.sp .6
.RS 4n
\fBzonepath\fR
.RE

.sp
.ne 2
.na
\fB(global)\fR
.ad
.sp .6
.RS 4n
\fBautoboot\fR
.RE

.sp
.ne 2
.na
\fB(global)\fR
.ad
.sp .6
.RS 4n
\fBbootargs\fR
.RE

.sp
.ne 2
.na
\fB(global)\fR
.ad
.sp .6
.RS 4n
\fBpool\fR
.RE

.sp
.ne 2
.na
\fB(global)\fR
.ad
.sp .6
.RS 4n
\fBlimitpriv\fR
.RE

.sp
.ne 2
.na
\fB(global)\fR
.ad
.sp .6
.RS 4n
\fBbrand\fR
.RE

.sp
.ne 2
.na
\fB(global)\fR
.ad
```

```
.sp .6
.RS 4n
\fBcpu-shares\fR
.RE

.sp
.ne 2
.na
\fB(global)\fR
.ad
.sp .6
.RS 4n
\fBhostid\fR
.RE

.sp
.ne 2
.na
\fB(global)\fR
.ad
.sp .6
.RS 4n
\fBmax-lwps\fR
.RE

.sp
.ne 2
.na
\fB(global)\fR
.ad
.sp .6
.RS 4n
\fBmax-msg-ids\fR
.RE

.sp
.ne 2
.na
\fB(global)\fR
.ad
.sp .6
.RS 4n
\fBmax-sem-ids\fR
.RE

.sp
.ne 2
.na
\fB(global)\fR
.ad
.sp .6
.RS 4n
\fBmax-shm-ids\fR
.RE

.sp
.ne 2
.na
\fB(global)\fR
.ad
.sp .6
.RS 4n
\fBmax-shm-memory\fR
.RE

.sp
```

```
.ne 2
.na
\fB(global)\fR
.ad
.sp .6
.RS 4n
\fBscheduling-class\fR
.RE

.sp
.ne 2
.na
.B (global)
.ad
.sp .6
.RS 4n
.B fs-allowed
.RE

.sp
.ne 2
.na
\fB\fBfs\fR\fR
.ad
.sp .6
.RS 4n
\fBdir\fR, \fBspecial\fR, \fBraw\fR, \fBtype\fR, \fBoptions\fR
.RE

.sp
.ne 2
.na
\fB\fBnet\fR\fR
.ad
.sp .6
.RS 4n
\fBaddress\fR, \fBphysical\fR, \fBdefrouter\fR
.RE

.sp
.ne 2
.na
\fB\fBdevice\fR\fR
.ad
.sp .6
.RS 4n
\fBmatch\fR
.RE

.sp
.ne 2
.na
\fB\fBrctl\fR\fR
.ad
.sp .6
.RS 4n
\fBname\fR, \fBvalue\fR
.RE

.sp
.ne 2
.na
\fB\fBattr\fR\fR
.ad
.sp .6
.RS 4n
```

```
\fBname\fR, \fBtype\fR, \fBvalue\fR
.RE

.sp
.ne 2
.na
\fB\fBdataset\fR\fR
.ad
.sp .6
.RS 4n
\fBname\fR
.RE

.sp
.ne 2
.na
\fB\fBdedicated-cpu\fR\fR
.ad
.sp .6
.RS 4n
\fBncpus\fR, \fBimportance\fR
.RE

.sp
.ne 2
.na
\fB\fBcapped-memory\fR\fR
.ad
.sp .6
.RS 4n
\fBphysical\fR, \fBswap\fR, \fBlocked\fR
.RE

.sp
.ne 2
.na
\fB\fBcapped-cpu\fR\fR
.ad
.sp .6
.RS 4n
\fBncpus\fR
.RE

.sp
.ne 2
.na
\fB\fBsecurity-flags\fB\fB
.ad
.sp .6
.RS 4n
\fBlower\fR, \fBdefault\fR, \fBupper\fR.
.RE

.sp
.LP
As for the property values which are paired with these names, they are either
simple, complex, or lists. The type allowed is property-specific. Simple values
are strings, optionally enclosed within quotation marks. Complex values have
the syntax:
.sp
.in +2
.nf
(<\fIname\fR>=<\fIvalue\fR>,<\fIname\fR>=<\fIvalue\fR>,...)
.fi
.in -2
.sp
```

```
.sp
.LP
where each <\fIvalue\fR> is simple, and the <\fIname\fR> strings are unique
within a given property. Lists have the syntax:
.sp
.in +2
.nf
[<\fIvalue\fR>,...]
.fi
.in -2
.sp

.sp
.LP
where each <\fIvalue\fR> is either simple or complex. A list of a single value
(either simple or complex) is equivalent to specifying that value without the
list syntax. That is, "foo" is equivalent to "[foo]". A list can be empty
(denoted by "[]").
.sp
.LP
In interpreting property values, \fBzonecfg\fR accepts regular expressions as
specified in \fBfnmatch\fR(5). See \fBEXAMPLES\fR.
.sp
.LP
The property types are described as follows:
.sp
.ne 2
.na
\fBglobal: \fBzonename\fR\fR
.ad
.sp .6
.RS 4n
The name of the zone.
.RE

.sp
.ne 2
.na
\fBglobal: \fBzonepath\fR\fR
.ad
.sp .6
.RS 4n
Path to zone’s file system.
.RE

.sp
.ne 2
.na
\fBglobal: \fBautoboot\fR\fR
.ad
.sp .6
.RS 4n
Boolean indicating that a zone should be booted automatically at system boot.
Note that if the zones service is disabled, the zone will not autoboot,
regardless of the setting of this property. You enable the zones service with a
\fBsvcadm\fR command, such as:
.sp
.in +2
.nf
# \fBsvcadm enable svc:/system/zones:default\fR
.fi
.in -2
.sp

Replace \fBenable\fR with \fBdisable\fR to disable the zones service. See
```

```
\fBsvcadm\fR(1M).
.RE

.sp
.ne 2
.na
\fBglobal: \fBbootargs\fR\fR
.ad
.sp .6
.RS 4n
Arguments (options) to be passed to the zone bootup, unless options are
supplied to the "\fBzoneadm boot\fR" command, in which case those take
precedence. The valid arguments are described in \fBzoneadm\fR(1M).
.RE

.sp
.ne 2
.na
\fBglobal: \fBpool\fR\fR
.ad
.sp .6
.RS 4n
Name of the resource pool that this zone must be bound to when booted. This
property is incompatible with the \fBdedicated-cpu\fR resource.
.RE

.sp
.ne 2
.na
\fBglobal: \fBlimitpriv\fR\fR
.ad
.sp .6
.RS 4n
The maximum set of privileges any process in this zone can obtain. The property
should consist of a comma-separated privilege set specification as described in
\fBpriv_str_to_set\fR(3C). Privileges can be excluded from the resulting set by
preceding their names with a dash (-) or an exclamation point (!). The special
privilege string "zone" is not supported in this context. If the special string
"default" occurs as the first token in the property, it expands into a safe set
of privileges that preserve the resource and security isolation described in
\fBzones\fR(5). A missing or empty property is equivalent to this same set of
safe privileges.
.sp
The system administrator must take extreme care when configuring privileges for
a zone. Some privileges cannot be excluded through this mechanism as they are
required in order to boot a zone. In addition, there are certain privileges
which cannot be given to a zone as doing so would allow processes inside a zone
to unduly affect processes in other zones. \fBzoneadm\fR(1M) indicates when an
invalid privilege has been added or removed from a zone’s privilege set when an
attempt is made to either "boot" or "ready" the zone.
.sp
See \fBprivileges\fR(5) for a description of privileges. The command "\fBppriv
-l\fR" (see \fBppriv\fR(1)) produces a list of all Solaris privileges. You can
specify privileges as they are displayed by \fBppriv\fR. In
\fBprivileges\fR(5), privileges are listed in the form
PRIV_\fIprivilege_name\fR. For example, the privilege \fIsys_time\fR, as you
would specify it in this property, is listed in \fBprivileges\fR(5) as
\fBPRIV_SYS_TIME\fR.
.RE

.sp
.ne 2
.na
\fBglobal: \fBbrand\fR\fR
.ad
.sp .6
```

```
.RS 4n
The zone’s brand type.
.RE

.sp
.ne 2
.na
\fBglobal: \fBip-type\fR\fR
.ad
.sp .6
.RS 4n
A zone can either share the IP instance with the global zone, which is the
default, or have its own exclusive instance of IP.
.sp
This property takes the values \fBshared\fR and \fBexclusive\fR.
.RE

.sp
.ne 2
.na
\fBglobal: \fBhostid\fR\fR
.ad
.sp .6
.RS 4n
A zone can emulate a 32-bit host identifier to ease system consolidation. A
zone’s \fBhostid\fR property is empty by default, meaning that the zone does
not emulate a host identifier. Zone host identifiers must be hexadecimal values
between 0 and FFFFFFFE. A \fB0x\fR or \fB0X\fR prefix is optional. Both
uppercase and lowercase hexadecimal digits are acceptable.
.RE

.sp
.ne 2
.na
\fB\fBfs\fR: dir, special, raw, type, options\fR
.ad
.sp .6
.RS 4n
Values needed to determine how, where, and so forth to mount file systems. See
\fBmount\fR(1M), \fBmount\fR(2), \fBfsck\fR(1M), and \fBvfstab\fR(4).
.RE

.sp
.ne 2
.na
\fB\fBnet\fR: address, physical, defrouter\fR
.ad
.sp .6
.RS 4n
The network address and physical interface name of the network interface. The
network address is one of:
.RS +4
.TP
.ie t \(bu
.el o
a valid IPv4 address, optionally followed by "\fB/\fR" and a prefix length;
.RE
.RS +4
.TP
.ie t \(bu
.el o
a valid IPv6 address, which must be followed by "\fB/\fR" and a prefix length;
.RE
.RS +4
.TP
.ie t \(bu
.el o
a host name which resolves to an IPv4 address.
.RE
Note that host names that resolve to IPv6 addresses are not supported.
.sp
The physical interface name is the network interface name.
.sp
The default router is specified similarly to the network address except that it
must not be followed by a \fB/\fR (slash) and a network prefix length.
.sp
A zone can be configured to be either exclusive-IP or shared-IP. For a
shared-IP zone, you must set both the physical and address properties; setting
the default router is optional. The interface specified in the physical
property must be plumbed in the global zone prior to booting the non-global
zone. However, if the interface is not used by the global zone, it should be
configured \fBdown\fR in the global zone, and the default router for the
interface should be specified here.
.sp
For an exclusive-IP zone, the physical property must be set and the address and
default router properties cannot be set.
.RE

.sp
.ne 2
.na
\fB\fBdevice\fR: match\fR
.ad
.sp .6
.RS 4n
Device name to match.
.RE

.sp
.ne 2
.na
\fB\fBrctl\fR: name, value\fR
.ad
.sp .6
.RS 4n
The name and \fIpriv\fR/\fIlimit\fR/\fIaction\fR triple of a resource control.
See \fBprctl\fR(1) and \fBrctladm\fR(1M). The preferred way to set rctl values
is to use the global property name associated with a specific rctl.
.RE

.sp
.ne 2
.na
\fB\fBattr\fR: name, type, value\fR
.ad
.sp .6
.RS 4n
The name, type and value of a generic attribute. The \fBtype\fR must be one of
\fBint\fR, \fBuint\fR, \fBboolean\fR or \fBstring\fR, and the value must be of
that type. \fBuint\fR means unsigned , that is, a non-negative integer.
.RE

.sp
.ne 2
.na
\fB\fBdataset\fR: name\fR
.ad
.sp .6
.RS 4n
The name of a \fBZFS\fR dataset to be accessed from within the zone. See
\fBzfs\fR(1M).
.RE
```

```
.sp
.ne 2
.na
\fBglobal: \fBcpu-shares\fR\fR
.ad
.sp .6
.RS 4n
The number of Fair Share Scheduler (FSS) shares to allocate to this zone. This
property is incompatible with the \fBdedicated-cpu\fR resource. This property
is the preferred way to set the \fBzone.cpu-shares\fR rctl.
.RE

.sp
.ne 2
.na
\fBglobal: \fBmax-lwps\fR\fR
.ad
.sp .6
.RS 4n
The maximum number of LWPs simultaneously available to this zone. This property
is the preferred way to set the \fBzone.max-lwps\fR rctl.
.RE

.sp
.ne 2
.na
\fBglobal: \fBmax-msg-ids\fR\fR
.ad
.sp .6
.RS 4n
The maximum number of message queue IDs allowed for this zone. This property is
the preferred way to set the \fBzone.max-msg-ids\fR rctl.
.RE

.sp
.ne 2
.na
\fBglobal: \fBmax-sem-ids\fR\fR
.ad
.sp .6
.RS 4n
The maximum number of semaphore IDs allowed for this zone. This property is the
preferred way to set the \fBzone.max-sem-ids\fR rctl.
.RE

.sp
.ne 2
.na
\fBglobal: \fBmax-shm-ids\fR\fR
.ad
.sp .6
.RS 4n
The maximum number of shared memory IDs allowed for this zone. This property is
the preferred way to set the \fBzone.max-shm-ids\fR rctl.
.RE

.sp
.ne 2
.na
\fBglobal: \fBmax-shm-memory\fR\fR
.ad
.sp .6
.RS 4n
The maximum amount of shared memory allowed for this zone. This property is the
preferred way to set the \fBzone.max-shm-memory\fR rctl. A scale (K, M, G, T)
```

```
can be applied to the value for this number (for example, 1M is one megabyte).
.RE

.sp
.ne 2
.na
\fBglobal: \fBscheduling-class\fR\fR
.ad
.sp .6
.RS 4n
Specifies the scheduling class used for processes running in a zone. When this
property is not specified, the scheduling class is established as follows:
.RS +4
.TP
.ie t \(bu
.el o
If the \fBcpu-shares\fR property or equivalent rctl is set, the scheduling
class FSS is used.
.RE
.RS +4
.TP
.ie t \(bu
.el o
If neither \fBcpu-shares\fR nor the equivalent rctl is set and the zone’s pool
property references a pool that has a default scheduling class, that class is
used.
.RE
.RS +4
.TP
.ie t \(bu
.el o
Under any other conditions, the system default scheduling class is used.
.RE
.RE



.sp
.ne 2
.na
\fB\fBdedicated-cpu\fR: ncpus, importance\fR
.ad
.sp .6
.RS 4n
The number of CPUs that should be assigned for this zone’s exclusive use. The
zone will create a pool and processor set when it boots. See \fBpooladm\fR(1M)
and \fBpoolcfg\fR(1M) for more information on resource pools. The \fBncpu\fR
property can specify a single value or a range (for example, 1-4) of
processors. The \fBimportance\fR property is optional; if set, it will specify
the \fBpset.importance\fR value for use by \fBpoold\fR(1M). If this resource is
used, there must be enough free processors to allocate to this zone when it
boots or the zone will not boot. The processors assigned to this zone will not
be available for the use of the global zone or other zones. This resource is
incompatible with both the \fBpool\fR and \fBcpu-shares\fR properties. Only a
single instance of this resource can be added to the zone.
.RE

.sp
.ne 2
.na
\fB\fBcapped-memory\fR: physical, swap, locked\fR
.ad
.sp .6
.RS 4n
The caps on the memory that can be used by this zone. A scale (K, M, G, T) can
be applied to the value for each of these numbers (for example, 1M is one
```

```
megabyte). Each of these properties is optional but at least one property must
be set when adding this resource. Only a single instance of this resource can
be added to the zone. The \fBphysical\fR property sets the \fBmax-rss\fR for
this zone. This will be enforced by \fBrcapd\fR(1M) running in the global zone.
The \fBswap\fR property is the preferred way to set the \fBzone.max-swap\fR
rctl. The \fBlocked\fR property is the preferred way to set the
\fBzone.max-locked-memory\fR rctl.
.RE

.sp
.ne 2
.na
\fB\fBcapped-cpu\fR: ncpus\fR
.ad
.sp .6
.RS 4n
Sets a limit on the amount of CPU time that can be used by a zone. The unit
used translates to the percentage of a single CPU that can be used by all user
threads in a zone, expressed as a fraction (for example, \fB\&.75\fR) or a
mixed number (whole number and fraction, for example, \fB1.25\fR). An
\fBncpu\fR value of \fB1\fR means 100% of a CPU, a value of \fB1.25\fR means
125%, \fB\&.75\fR mean 75%, and so forth. When projects within a capped zone
have their own caps, the minimum value takes precedence.
.sp
The \fBcapped-cpu\fR property is an alias for \fBzone.cpu-cap\fR resource
control and is related to the \fBzone.cpu-cap\fR resource control. See
\fBresource_controls\fR(5).
.RE

.sp
.ne 2
.na
\fB\fBsecurity-flags\fR: lower, default, upper\fR
.ad
.sp .6
.RS 4n
Set the process security flags associated with the zone.  The \fBlower\fR and
\fBupper\fR fields set the limits, the \fBdefault\fR field is set of flags all
zone processes inherit.
.RE

.sp
.ne 2
.na
\fBglobal: \fBfs-allowed\fR\fR
.ad
.sp .6
.RS 4n
A comma-separated list of additional filesystems that may be mounted within
the zone; for example "ufs,pcfs". By default, only hsfs(7fs) and network
filesystems can be mounted. If the first entry in the list is "-" then
that disables all of the default filesystems. If any filesystems are listed
after "-" then only those filesystems can be mounted.

This property does not apply to filesystems mounted into the zone via "add fs"
or "add dataset".

WARNING: allowing filesystem mounts other than the default may allow the zone
administrator to compromise the system with a malicious filesystem image, and
is not supported.
.RE

.sp
.LP
The following table summarizes resources, property-names, and types:
.sp
```

```
.in +2
.nf
resource          property-name   type
(global)          zonename        simple
(global)          zonepath        simple
(global)          autoboot        simple
(global)          bootargs        simple
(global)          pool            simple
(global)          limitpriv       simple
(global)          brand           simple
(global)          ip-type         simple
(global)          hostid          simple
(global)          cpu-shares      simple
(global)          max-lwps        simple
(global)          max-msg-ids     simple
(global)          max-sem-ids     simple
(global)          max-shm-ids     simple
(global)          max-shm-memory  simple
(global)          scheduling-class simple
fs                dir             simple
                   special         simple
                   raw             simple
                   type            simple
                   options         list of simple
net               address         simple
                   physical        simple
device            match           simple
rctl              name            simple
                   value           list of complex
attr              name            simple
                   type            simple
                   value           simple
dataset           name            simple
dedicated-cpu     ncpus           simple or range
                   importance      simple

capped-memory     physical        simple with scale
                   swap            simple with scale
                   locked          simple with scale

capped-cpu        ncpus           simple
security-flags    lower           simple
                   default        simple
                   upper          simple
.fi
.in -2
.sp

.sp
.LP
To further specify things, the breakdown of the complex property "value" of the
"rctl" resource type, it consists of three name/value pairs, the names being
"priv", "limit" and "action", each of which takes a simple value. The "name"
property of an "attr" resource is syntactically restricted in a fashion similar
but not identical to zone names: it must begin with an alphanumeric, and can
contain alphanumerics plus the hyphen (\fB-\fR), underscore (\fB_\fR), and dot
(\fB\&.\fR) characters. Attribute names beginning with "zone" are reserved for
use by the system. Finally, the "autoboot" global property must have a value of
"true" or "false".
.SS "Using Kernel Statistics to Monitor CPU Caps"
.LP
Using the kernel statistics (\fBkstat\fR(3KSTAT)) module \fBcaps\fR, the system
maintains information for all capped projects and zones. You can access this
information by reading kernel statistics (\fBkstat\fR(3KSTAT)), specifying
\fBcaps\fR as the \fBkstat\fR module name. The following command displays
kernel statistics for all active CPU caps:
```

```
.sp
.in +2
.nf
# \fBkstat caps::'/cpucaps/'\fR
.fi
.in -2
.sp

.sp
.LP
A \fBkstat\fR(1M) command running in a zone displays only CPU caps relevant for
that zone and for projects in that zone. See \fBEXAMPLES\fR.
.sp
.LP
The following are cap-related arguments for use with \fBkstat\fR(1M):
.sp
.ne 2
.na
\fB\fBcaps\fR\fR
.ad
.sp .6
.RS 4n
The \fBkstat\fR module.
.RE

.sp
.ne 2
.na
\fB\fBproject_caps\fR or \fBzone_caps\fR\fR
.ad
.sp .6
.RS 4n
\fBkstat\fR class, for use with the \fBkstat\fR \fB-c\fR option.
.RE

.sp
.ne 2
.na
\fB\fBcpucaps_project_\fR\fIid\fR or \fBcpucaps_zone_\fR\fIid\fR\fR
.ad
.sp .6
.RS 4n
\fBkstat\fR name, for use with the \fBkstat\fR \fB-n\fR option. \fIid\fR is the
project or zone identifier.
.RE

.sp
.LP
The following fields are displayed in response to a \fBkstat\fR(1M) command
requesting statistics for all CPU caps.
.sp
.ne 2
.na
\fB\fBmodule\fR\fR
.ad
.sp .6
.RS 4n
In this usage of \fBkstat\fR, this field will have the value \fBcaps\fR.
.RE

.sp
.ne 2
.na
\fB\fBname\fR\fR
.ad
.sp .6
```

```
.RS 4n
As described above, \fBcpucaps_project_\fR\fIid\fR or
\fBcpucaps_zone_\fR\fIid\fR
.RE

.sp
.ne 2
.na
\fB\fBabove_sec\fR\fR
.ad
.sp .6
.RS 4n
Total time, in seconds, spent above the cap.
.RE

.sp
.ne 2
.na
\fB\fBbelow_sec\fR\fR
.ad
.sp .6
.RS 4n
Total time, in seconds, spent below the cap.
.RE

.sp
.ne 2
.na
\fB\fBmaxusage\fR\fR
.ad
.sp .6
.RS 4n
Maximum observed CPU usage.
.RE

.sp
.ne 2
.na
\fB\fBnwait\fR\fR
.ad
.sp .6
.RS 4n
Number of threads on cap wait queue.
.RE

.sp
.ne 2
.na
\fB\fBusage\fR\fR
.ad
.sp .6
.RS 4n
Current aggregated CPU usage for all threads belonging to a capped project or
zone, in terms of a percentage of a single CPU.
.RE

.sp
.ne 2
.na
\fB\fBvalue\fR\fR
.ad
.sp .6
.RS 4n
The cap value, in terms of a percentage of a single CPU.
.RE
```

```
.sp
.ne 2
.na
\fB\fBzonename\fR\fR
.ad
.sp .6
.RS 4n
Name of the zone for which statistics are displayed.
.RE

.sp
.LP
See \fBEXAMPLES\fR for sample output from a \fBkstat\fR command.
.SH OPTIONS
.LP
The following options are supported:
.sp
.ne 2
.na
\fB\fB-f\fR \fIcommand_file\fR\fR
.ad
.sp .6
.RS 4n
Specify the name of \fBzonecfg\fR command file. \fIcommand_file\fR is a text
file of \fBzonecfg\fR subcommands, one per line.
.RE

.sp
.ne 2
.na
\fB\fB-z\fR \fIzonename\fR\fR
.ad
.sp .6
.RS 4n
Specify the name of a zone. Zone names are case sensitive. Zone names must
begin with an alphanumeric character and can contain alphanumeric characters,
the underscore (\fB_\fR) the hyphen (\fB-\fR), and the dot (\fB\&.\fR). The
name \fBglobal\fR and all names beginning with \fBSUNW\fR are reserved and
cannot be used.
.RE

.SH SUBCOMMANDS
.LP
You can use the \fBadd\fR and \fBselect\fR subcommands to select a specific
resource, at which point the scope changes to that resource. The \fBend\fR and
\fBcancel\fR subcommands are used to complete the resource specification, at
which time the scope is reverted back to global. Certain subcommands, such as
\fBadd\fR, \fBremove\fR and \fBset\fR, have different semantics in each scope.
.sp
.LP
\fBzonecfg\fR supports a semicolon-separated list of subcommands. For example:
.sp
.in +2
.nf
# \fBzonecfg -z myzone "add net; set physical=myvnic; end"\fR
.fi
.in -2
.sp

.sp
.LP
Subcommands which can result in destructive actions or loss of work have an
\fB-F\fR option to force the action. If input is from a terminal device, the
user is prompted when appropriate if such a command is given without the
\fB-F\fR option otherwise, if such a command is given without the \fB-F\fR
option, the action is disallowed, with a diagnostic message written to standard
```

```
error.
.sp
.LP
The following subcommands are supported:
.sp
.ne 2
.na
\fB\fBadd\fR \fIresource-type\fR (global scope)\fR
.ad
.br
.na
\fB\fBadd\fR \fIproperty-name property-value\fR (resource scope)\fR
.ad
.sp .6
.RS 4n
In the global scope, begin the specification for a given resource type. The
scope is changed to that resource type.
.sp
In the resource scope, add a property of the given name with the given value.
The syntax for property values varies with different property types. In
general, it is a simple value or a list of simple values enclosed in square
brackets, separated by commas (\fB[foo,bar,baz]\fR). See \fBPROPERTIES\fR.
.RE

.sp
.ne 2
.na
\fB\fBcancel\fR\fR
.ad
.sp .6
.RS 4n
End the resource specification and reset scope to global. Abandons any
partially specified resources. \fBcancel\fR is only applicable in the resource
scope.
.RE

.sp
.ne 2
.na
\fB\fBclear\fR \fIproperty-name\fR\fR
.ad
.sp .6
.RS 4n
Clear the value for the property.
.RE

.sp
.ne 2
.na
\fB\fBcommit\fR\fR
.ad
.sp .6
.RS 4n
Commit the current configuration from memory to stable storage. The
configuration must be committed to be used by \fBzoneadm\fR. Until the
in-memory configuration is committed, you can remove changes with the
\fBrevert\fR subcommand. The \fBcommit\fR operation is attempted automatically
upon completion of a \fBzonecfg\fR session. Since a configuration must be
correct to be committed, this operation automatically does a verify.
.RE

.sp
.ne 2
.na
\fB\fBcreate [\fR\fB-F\fR\fB] [\fR \fB-a\fR \fIpath\fR |\fB-b\fR \fB|\fR
\fB-t\fR \fItemplate\fR\fB]\fR\fR
```

```
.ad
.sp .6
.RS 4n
Create an in-memory configuration for the specified zone. Use \fBcreate\fR to
begin to configure a new zone. See \fBcommit\fR for saving this to stable
storage.
.sp
If you are overwriting an existing configuration, specify the \fB-F\fR option
to force the action. Specify the \fB-t\fR \fItemplate\fR option to create a
configuration identical to \fItemplate\fR, where \fItemplate\fR is the name of
a configured zone.
.sp
Use the \fB-a\fR \fIpath\fR option to facilitate configuring a detached zone on
a new host. The \fIpath\fR parameter is the zonepath location of a detached
zone that has been moved on to this new host. Once the detached zone is
configured, it should be installed using the "\fBzoneadm attach\fR" command
(see \fBzoneadm\fR(1M)). All validation of the new zone happens during the
\fBattach\fR process, not during zone configuration.
.sp
Use the \fB-b\fR option to create a blank configuration. Without arguments,
\fBcreate\fR applies the Sun default settings.
.RE

.sp
.ne 2
.na
\fB\fBdelete [\fR\fB-F\fR\fB]\fR\fR
.ad
.sp .6
.RS 4n
Delete the specified configuration from memory and stable storage. This action
is instantaneous, no commit is necessary. A deleted configuration cannot be
reverted.
.sp
Specify the \fB-F\fR option to force the action.
.RE

.sp
.ne 2
.na
\fB\fBend\fR\fR
.ad
.sp .6
.RS 4n
End the resource specification. This subcommand is only applicable in the
resource scope. \fBzonecfg\fR checks to make sure the current resource is
completely specified. If so, it is added to the in-memory configuration (see
\fBcommit\fR for saving this to stable storage) and the scope reverts to
global. If the specification is incomplete, it issues an appropriate error
message.
.RE

.sp
.ne 2
.na
\fB\fBexport [\fR\fB-f\fR \fIoutput-file\fR\fB]\fR\fR
.ad
.sp .6
.RS 4n
Print configuration to standard output. Use the \fB-f\fR option to print the
configuration to \fIoutput-file\fR. This option produces output in a form
suitable for use in a command file.
.RE

.sp
.ne 2
```

```
.na
\fB\fBhelp [usage] [\fIsubcommand\fR] [syntax] [\fR\fIcommand-name\fR\fB]\fR\fR
.ad
.sp .6
.RS 4n
Print general help or help about given topic.
.RE

.sp
.ne 2
.na
\fB\fBinfo zonename | zonepath | autoboot | brand | pool | limitpriv\fR\fR
.ad
.br
.na
\fB\fBinfo [\fR\fIresource-type\fR
\fB[\fR\fIproperty-name\fR\fB=\fR\fIproperty-value\fR\fB]*]\fR\fR
.ad
.sp .6
.RS 4n
Display information about the current configuration. If \fIresource-type\fR is
specified, displays only information about resources of the relevant type. If
any \fIproperty-name\fR value pairs are specified, displays only information
about resources meeting the given criteria. In the resource scope, any
arguments are ignored, and \fBinfo\fR displays information about the resource
which is currently being added or modified.
.RE

.sp
.ne 2
.na
\fB\fBremove\fR \fIresource-type\fR\fB{\fR\fIproperty-name\fR\fB=\fR\fIproperty
-value\fR\fB}\fR(global scope)\fR
.ad
.sp .6
.RS 4n
In the global scope, removes the specified resource. The \fB[]\fR syntax means
0 or more of whatever is inside the square braces. If you want only to remove a
single instance of the resource, you must specify enough property name-value
pairs for the resource to be uniquely identified. If no property name-value
pairs are specified, all instances will be removed. If there is more than one
pair is specified, a confirmation is required, unless you use the \fB-F\fR
option.
.RE

.sp
.ne 2
.na
\fB\fBselect\fR \fIresource-type\fR
\fB{\fR\fIproperty-name\fR\fB=\fR\fIproperty-value\fR\fB}\fR\fR
.ad
.sp .6
.RS 4n
Select the resource of the given type which matches the given
\fIproperty-name\fR \fIproperty-value\fR pair criteria, for modification. This
subcommand is applicable only in the global scope. The scope is changed to that
resource type. The \fB{}\fR syntax means 1 or more of whatever is inside the
curly braces. You must specify enough \fIproperty -name property-value\fR pairs
for the resource to be uniquely identified.
.RE

.sp
.ne 2
.na
\fB\fBset\fR \fIproperty-name\fR\fB=\fR\fIproperty\fR\fB-\fR\fIvalue\fR\fR
.ad
```

```
.sp .6
.RS 4n
Set a given property name to the given value. Some properties (for example,
\fBzonename\fR and \fBzonepath\fR) are global while others are
resource-specific. This subcommand is applicable in both the global and
resource scopes.
.RE

.sp
.ne 2
.na
\fB\fBverify\fR\fR
.ad
.sp .6
.RS 4n
Verify the current configuration for correctness:
.RS +4
.TP
.ie t \(bu
.el o
All resources have all of their required properties specified.
.RE
.RS +4
.TP
.ie t \(bu
.el o
A \fBzonepath\fR is specified.
.RE
.RE

.sp
.ne 2
.na
\fB\fBrevert\fR \fB[\fR\fB-F\fR\fB]\fR\fR
.ad
.sp .6
.RS 4n
Revert the configuration back to the last committed state. The \fB-F\fR option
can be used to force the action.
.RE

.sp
.ne 2
.na
\fB\fBexit [\fR\fB-F\fR\fB]\fR\fR
.ad
.sp .6
.RS 4n
Exit the \fBzonecfg\fR session. A commit is automatically attempted if needed.
You can also use an \fBEOF\fR character to exit \fBzonecfg\fR. The \fB-F\fR
option can be used to force the action.
.RE

.SH EXAMPLES
.LP
\fBExample 1 \fRCreating the Environment for a New Zone
.sp
.LP
In the following example, \fBzonecfg\fR creates the environment for a new zone.
\fB/usr/local\fR is loopback mounted from the global zone into
\fB/opt/local\fR. \fB/opt/sfw\fR is loopback mounted from the global zone,
three logical network interfaces are added, and a limit on the number of
fair-share scheduler (FSS) CPU shares for a zone is set using the \fBrctl\fR
resource type. The example also shows how to select a given resource for
modification.
```

```
.sp
.in +2
.nf
example# \fBzonecfg -z myzone3\fR
my-zone3: No such zone configured
Use 'create' to begin configuring a new zone.
zonecfg:myzone3> \fBcreate\fR
zonecfg:myzone3> \fBset zonepath=/export/home/my-zone3\fR
zonecfg:myzone3> \fBset autoboot=true\fR
zonecfg:myzone3> \fBadd fs\fR
zonecfg:myzone3:fs> \fBset dir=/usr/local\fR
zonecfg:myzone3:fs> \fBset special=/opt/local\fR
zonecfg:myzone3:fs> \fBset type=lofs\fR
zonecfg:myzone3:fs> \fBadd options [ro,nodevices]\fR
zonecfg:myzone3:fs> \fBend\fR
zonecfg:myzone3> \fBadd fs\fR
zonecfg:myzone3:fs> \fBset dir=/mnt\fR
zonecfg:myzone3:fs> \fBset special=/dev/dsk/c0t0d0s7\fR
zonecfg:myzone3:fs> \fBset raw=/dev/rdsk/c0t0d0s7\fR
zonecfg:myzone3:fs> \fBset type=ufs\fR
zonecfg:myzone3:fs> \fBend\fR
zonecfg:myzone3> \fBadd net\fR
zonecfg:myzone3:net> \fBset address=192.168.0.1/24\fR
zonecfg:myzone3:net> \fBset physical=eri0\fR
zonecfg:myzone3:net> \fBend\fR
zonecfg:myzone3> \fBadd net\fR
zonecfg:myzone3:net> \fBset address=192.168.1.2/24\fR
zonecfg:myzone3:net> \fBset physical=eri0\fR
zonecfg:myzone3:net> \fBend\fR
zonecfg:myzone3> \fBadd net\fR
zonecfg:myzone3:net> \fBset address=192.168.2.3/24\fR
zonecfg:myzone3:net> \fBset physical=eri0\fR
zonecfg:myzone3:net> \fBend\fR
zonecfg:my-zone3> \fBset cpu-shares=5\fR
zonecfg:my-zone3> \fBadd capped-memory\fR
zonecfg:my-zone3:capped-memory> \fBset physical=50m\fR
zonecfg:my-zone3:capped-memory> \fBset swap=100m\fR
zonecfg:my-zone3:capped-memory> \fBend\fR
zonecfg:myzone3> \fBexit\fR
.fi
.in -2
.sp

.LP
\fBExample 2 \fRCreating a Non-Native Zone
.sp
.LP
The following example creates a new Linux zone:

.sp
.in +2
.nf
example# \fBzonecfg -z lxzone\fR
lxzone: No such zone configured
Use 'create' to begin configuring a new zone
zonecfg:lxzone> \fBcreate -t SUNWlx\fR
zonecfg:lxzone> \fBset zonepath=/export/zones/lxzone\fR
zonecfg:lxzone> \fBset autoboot=true\fR
zonecfg:lxzone> \fBexit\fR
.fi
.in -2
.sp

.LP
\fBExample 3 \fRCreating an Exclusive-IP Zone
.sp
```

```
.LP
The following example creates a zone that is granted exclusive access to
\fBbge1\fR and \fBbge33000\fR and that is isolated at the IP layer from the
other zones configured on the system.

.sp
.LP
The IP addresses and routing is configured inside the new zone using
\fBsysidtool\fR(1M).

.sp
.in +2
.nf
example# \fBzonecfg -z excl\fR
excl: No such zone configured
Use 'create' to begin configuring a new zone
zonecfg:excl> \fBcreate\fR
zonecfg:excl> \fBset zonepath=/export/zones/excl\fR
zonecfg:excl> \fBset ip-type=exclusive\fR
zonecfg:excl> \fBadd net\fR
zonecfg:excl:net> \fBset physical=bge1\fR
zonecfg:excl:net> \fBend\fR
zonecfg:excl> \fBadd net\fR
zonecfg:excl:net> \fBset physical=bge33000\fR
zonecfg:excl:net> \fBend\fR
zonecfg:excl> \fBexit\fR
.fi
.in -2
.sp

.LP
\fBExample 4 \fRAssociating a Zone with a Resource Pool
.sp
.LP
The following example shows how to associate an existing zone with an existing
resource pool:

.sp
.in +2
.nf
example# \fBzonecfg -z myzone\fR
zonecfg:myzone> \fBset pool=mypool\fR
zonecfg:myzone> \fBexit\fR
.fi
.in -2
.sp

.sp
.LP
For more information about resource pools, see \fBpooladm\fR(1M) and
\fBpoolcfg\fR(1M).

.LP
\fBExample 5 \fRChanging the Name of a Zone
.sp
.LP
The following example shows how to change the name of an existing zone:

.sp
.in +2
.nf
example# \fBzonecfg -z myzone\fR
zonecfg:myzone> \fBset zonename=myzone2\fR
zonecfg:myzone2> \fBexit\fR
.fi
.in -2
```

```
.sp

.LP
\fBExample 6 \fRChanging the Privilege Set of a Zone
.sp
.LP
The following example shows how to change the set of privileges an existing
zone’s processes will be limited to the next time the zone is booted. In this
particular case, the privilege set will be the standard safe set of privileges
a zone normally has along with the privilege to change the system date and
time:

.sp
.in +2
.nf
example# \fBzonecfg -z myzone\fR
zonecfg:myzone> \fBset limitpriv="default,sys_time"\fR
zonecfg:myzone2> \fBexit\fR
.fi
.in -2
.sp

.LP
\fBExample 7 \fRSetting the \fBzone.cpu-shares\fR Property for the Global Zone
.sp
.LP
The following command sets the \fBzone.cpu-shares\fR property for the global
zone:

.sp
.in +2
.nf
example# \fBzonecfg -z global\fR
zonecfg:global> \fBset cpu-shares=5\fR
zonecfg:global> \fBexit\fR
.fi
.in -2
.sp

.LP
\fBExample 8 \fRUsing Pattern Matching
.sp
.LP
The following commands illustrate \fBzonecfg\fR support for pattern matching.
In the zone \fBflexlm\fR, enter:

.sp
.in +2
.nf
zonecfg:flexlm> \fBadd device\fR
zonecfg:flexlm:device> \fBset match="/dev/cua/a00[2-5]"\fR
zonecfg:flexlm:device> \fBend\fR
.fi
.in -2
.sp

.sp
.LP
In the global zone, enter:

.sp
.in +2
.nf
global# \fBls /dev/cua\fR
a     a000  a001  a002  a003  a004  a005  a006  a007  b
.fi
```

```
.in -2
.sp

.sp
.LP
In the zone \fBflexlm\fR, enter:

.sp
.in +2
.nf
flexlm# \fBls /dev/cua\fR
a002  a003  a004  a005
.fi
.in -2
.sp

.LP
\fBExample 9 \fRSetting a Cap for a Zone to Three CPUs
.sp
.LP
The following sequence uses the \fBzonecfg\fR command to set the CPU cap for a
zone to three CPUs.

.sp
.in +2
.nf
zonecfg:myzone> \fBadd capped-cpu\fR
zonecfg:myzone>capped-cpu> \fBset ncpus=3\fR
zonecfg:myzone>capped-cpu>capped-cpu> \fBend\fR
.fi
.in -2
.sp

.sp
.LP
The preceding sequence, which uses the capped-cpu property, is equivalent to
the following sequence, which makes use of the \fBzone.cpu-cap\fR resource
control.

.sp
.in +2
.nf
zonecfg:myzone> \fBadd rctl\fR
zonecfg:myzone:rctl> \fBset name=zone.cpu-cap\fR
zonecfg:myzone:rctl> \fBadd value (priv=privileged,limit=300,action=none)\fR
zonecfg:myzone:rctl> \fBend\fR
.fi
.in -2
.sp

.LP
\fBExample 10 \fRUsing \fBkstat\fR to Monitor CPU Caps
.sp
.LP
The following command displays information about all CPU caps.

.sp
.in +2
.nf
# \fBkstat -n /cpucaps/\fR
module: caps                            instance: 0
name:   cpucaps_project_0               class:    project_caps
        above_sec                       0
        below_sec                       2157
        crtime                          821.048183159
        maxusage                        2
        nwait                           0
        snaptime                        235885.637253027
        usage                           0
        value                           18446743151372347932
        zonename                        global

module: caps                            instance: 0
name:   cpucaps_project_1               class:    project_caps
        above_sec                       0
        below_sec                       0
        crtime                          225339.192787265
        maxusage                        5
        nwait                           0
        snaptime                        235885.637591677
        usage                           5
        value                           18446743151372347932
        zonename                        global

module: caps                            instance: 0
name:   cpucaps_project_201             class:    project_caps
        above_sec                       0
        below_sec                       235105
        crtime                          780.37961782
        maxusage                        100
        nwait                           0
        snaptime                        235885.637789687
        usage                           43
        value                           100
        zonename                        global

module: caps                            instance: 0
name:   cpucaps_project_202             class:    project_caps
        above_sec                       0
        below_sec                       235094
        crtime                          791.72983782
        maxusage                        100
        nwait                           0
        snaptime                        235885.637967512
        usage                           48
        value                           100
        zonename                        global

module: caps                            instance: 0
name:   cpucaps_project_203             class:    project_caps
        above_sec                       0
        below_sec                       235034
        crtime                          852.104401481
        maxusage                        75
        nwait                           0
        snaptime                        235885.638144304
        usage                           47
        value                           100
        zonename                        global

module: caps                            instance: 0
name:   cpucaps_project_86710           class:    project_caps
        above_sec                       22
        below_sec                       235166
        crtime                          698.441717859
        maxusage                        101
        nwait                           0
        snaptime                        235885.638319871
        usage                           54
        value                           100
        zonename                        global
```

```
module: caps                            instance: 0
name:   cpucaps_zone_0                  class:    zone_caps
        above_sec                       100733
        below_sec                       134332
        crtime                          821.048177123
        maxusage                        207
        nwait                           2
        snaptime                        235885.638497731
        usage                           199
        value                           200
        zonename                        global

module: caps                            instance: 1
name:   cpucaps_project_0               class:    project_caps
        above_sec                       0
        below_sec                       0
        crtime                          225360.256448422
        maxusage                        7
        nwait                           0
        snaptime                        235885.638714404
        usage                           7
        value                           18446743151372347932
        zonename                        test_001

module: caps                            instance: 1
name:   cpucaps_zone_1                  class:    zone_caps
        above_sec                       2
        below_sec                       10524
        crtime                          225360.256440278
        maxusage                        106
        nwait                           0
        snaptime                        235885.638896443
        usage                           7
        value                           100
        zonename                        test_001
.fi
.in -2
.sp

.LP
\fBExample 11 \fRDisplaying CPU Caps for a Specific Zone or Project
.sp
.LP
Using the \fBkstat\fR \fB-c\fR and \fB-i\fR options, you can display CPU caps
for a specific zone or project, as below. The first command produces a display
for a specific project, the second for the same project within zone 1.

.sp
.in +2
.nf
# \fBkstat -c project_caps\fR

# \fBkstat -c project_caps -i 1\fR
.fi
.in -2
.sp

.SH EXIT STATUS
.LP
The following exit values are returned:
.sp
.ne 2
.na
\fB\fB0\fR\fR
.ad
.sp .6
```

```
.RS 4n
Successful completion.
.RE

.sp
.ne 2
.na
\fB\fB1\fR\fR
.ad
.sp .6
.RS 4n
An error occurred.
.RE

.sp
.ne 2
.na
\fB\fB2\fR\fR
.ad
.sp .6
.RS 4n
Invalid usage.
.RE

.SH ATTRIBUTES
.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
.LP
\fBppriv\fR(1), \fBprctl\fR(1), \fBzlogin\fR(1), \fBkstat\fR(1M),
\fBmount\fR(1M), \fBpooladm\fR(1M), \fBpoolcfg\fR(1M), \fBpoold\fR(1M),
\fBrcapd\fR(1M), \fBrctladm\fR(1M), \fBsvcadm\fR(1M), \fBsysidtool\fR(1M),
\fBzfs\fR(1M), \fBzoneadm\fR(1M), \fBpriv_str_to_set\fR(3C),
\fBkstat\fR(3KSTAT), \fBvfstab\fR(4), \fBattributes\fR(5), \fBbrands\fR(5),
\fBfnmatch\fR(5), \fBlx\fR(5), \fBprivileges\fR(5), \fBresource_controls\fR(5),
\fBsecurity-flags\fR(5), \fBzones\fR(5)
.sp
.LP
\fISystem Administration Guide: Solaris Containers-Resource Management, and
Solaris Zones\fR
.SH NOTES
.LP
All character data used by \fBzonecfg\fR must be in US-ASCII encoding.
```

```
***********************************************************
   34589 Wed Jun 15 19:31:42 2016
new/usr/src/man/man3lib/libproc.3lib
Code review comments from jeffpc
***********************************************************
.\"
.\" 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 2015 Joyent, Inc.
.\"
.Dd June 06, 2016
.Dd May 08, 2016
.Dt LIBPROC 3LIB
.Os
.Sh NAME
.Nm libproc
.Nd process control library
.Sh SYNOPSIS
.Lb libproc
.In libproc.h
.Sh DESCRIPTION
The
.Nm
library provides consumers a general series of interfaces to inspect
and control both live processes and core files. It is intended for
introspection tools such as debuggers by providing a high-level
interface to the /proc file system
.Pf ( Xr proc 4 ) .
.Pp
The
.Nm
library provides interfaces that focus on:
.Bl -bullet -offset indent
.It
Creating and attaching to live process, core files, and arbitrary ELF
objects.
.It
Interrogating the state of a process or core file.
.It
Manipulating the current state of a process or thread.
.It
Interrogating the state of threads of a process or core file.
.It
Running system calls in the context of another process.
.It
Various utilities for iterating process and core file file descriptors,
mappings, symbols, and more.
.It
Various utilities to support debugging tools.
.El
.Ss Live Processes
The
.Nm
library can be used to manipulate running processes and to create new
ones. To manipulate an existing process first
.Em grab
it with the
.Em Pgrab
```

```
function. A process is generally stopped as a side effect of grabbing
it. Callers must exercise caution, as if they do not use the library
correctly, or they terminate unexpectedly, a process may remain
stopped.
.Pp
Unprivileged users may only grab their own processes. Users with the
privilege
.Sy PRIV_PROC_OWNER
may manipulate processes that they do not own; however, additional
restrictions as described in
.Xr privileges 5
apply.
.Pp
In addition, the
.Fn Pcreate
and
.Fn Pxcreate
functions may be used to create processes which are always controlled by
the library.
.Ss Core Files
The
.Nm
library has the ability to open and interpret core files produced by
processes on the system. Process core dump generation is controlled by
the
.Xr coreadm 1M
command. In addition, the library has the ability to understand and
interpret core dumps generated by Linux kernel and can provide a subset
of its functionality on such core files, provided the original binary is
also present.
.Pp
Not all functions in the
.Nm
library are valid for core files. In general, none of the commands
which manipulate the current state of a process or thread or that try
to force system calls on a victim process will work. Furthermore
several of the information and iteration interfaces are limited based
on the data that is available in the core file. For example, if the
core file is of a process that omits the frame pointer, the ability to
iterate the stack will be limited.
.Pp
Use the
.Fn Pgrab_core
or
.Fn Pfgrab_core
function to open a core file. Use the
.Fn Pgrab_file
function to open an ELF object file.
This is useful for obtaining information stored in ELF headers and
sections.
.Ss Debug Information
Many of the operations in the library rely on debug information being
present in a process and its associated libraries. The library
leverages symbol table information, CTF data
.Pf ( Xr CTF 4 )
sections, and frame unwinding information based on the use of an ABI
defined frame pointer, eg.
.Sy %ebp
and
.Sy %rbp
on x86 systems.
.Pp
Some software providers strip programs of this information or build
their executables such that the information will not be present in a
core dump. To deal with this fact, the library is able to consume
information that is not present in the core file or the running
```

```
process. It can both consume it from the underlying executable and it
also supports finding it from related ELF objects that are linked to
it via the
.Sy .gnu_debuglink
and the
.Sy .note.gnu.build-id
ELF sections.
.Ss Iteration Interfaces
The
.Nm
library provides the ability to iterate over the following aspects of a
process or core file:
.Bl -bullet -offset indent
.It
Active threads
.It
Active and zombie threads
.It
All non-system processes
.It
All process mappings
.It
All objects in a process
.It
The environment
.It
The symbol table
.It
Stack frames
.It
File Descriptors
.El
.Ss System Call Injection
The
.Nm
library allows the caller to force system calls to be executed in the
context of the running process. This can be used both as a tool for
introspection, allowing one to get information outside its current
context as well as performing modifications to a process.
.Pp
These functions run in the context of the calling process. This is
often an easier way of getting non-exported information about a
process from the system. For example, the
.Xr pfiles 1
command uses this interface to get more detailed information about a
process's open file descriptors, which it would not have access to
otherwise.
.Sh INTERFACES
The shared object
.Sy libproc.so.1
provides the public interfaces defined below. See
.Xr Intro 3
for additional information on shared object interfaces. Functions are
organized into categories that describe their purpose. Individual
functions are documented in their own manual pages.
.Ss Creation, Grabbing, and Releasing
The following routines are related to creating library handles,
grabbing cores, processes, and threads, and releasing those resources.
.Bl -column -offset indent ".Sy Pmapping_iter_resolved" ".Sy Psymbol_iter_by_add
.It Sy Lfree Ta Sy Lgrab
.It Sy Lgrab_error Ta Sy Pcreate
.It Sy Pcreate_agent Ta Sy Pcreate_callback
.It Sy Pcreate_error Ta Sy Pdestroy_agent
.It Sy Pfgrab_core Ta Sy Pfree
.It Sy Pgrab Ta Sy Pgrab_core
.It Sy Pgrab_error Ta Sy Pgrab_file
```

```
.It Sy Pgrab_ops Ta Sy Prelease
.It Sy Preopen Ta Sy Pxcreate
.El
.Ss Process interrogation and manipulation
The following routines obtain information about a process and allow
manipulation of the process itself.
.Bl -column -offset indent ".Sy Pmapping_iter_resolved" ".Sy Psymbol_iter_by_add
.It Sy Paddr_to_ctf Ta Sy Paddr_to_loadobj
.It Sy Paddr_to_map Ta Sy Paddr_to_text_map
.It Sy Pasfd Ta Sy Pclearfault
.It Sy Pclearsig Ta Sy Pcontent
.It Sy Pcred Ta Sy Pctlfd
.It Sy Pdelbkpt Ta Sy Pdelwapt
.It Sy Pdstop Ta Sy Pexecname
.It Sy Pfault Ta Sy Pfgcore
.It Sy Pgcore Ta Sy Pgetareg
.It Sy Pgetauxval Ta Sy Pgetauxvec
.It Sy Pgetenv Ta Sy Pisprocdir
.It Sy Pissyscall_prev Ta Sy Plmid
.It Sy Plmid_to_loadobj Ta Sy Plmid_to_map
.It Sy Plookup_by_addr Ta Sy Plookup_by_name
.It Sy Plwp_alt_stack Ta Sy Plwp_getfpregs
.It Sy Plwp_getpsinfo Ta Sy Plwp_getregs
.It Sy Plwp_getspymaster Ta Sy Plwp_main_stack
.It Sy Plwp_setfpregs Ta Sy Plwp_setregs
.It Sy Plwp_stack Ta Sy Pname_to_ctf
.It Sy Pname_to_loadobj Ta Sy Pname_to_map
.It Sy Pobjname Ta Sy Pobjname_resolved
.It Sy Pplatform Ta Sy Ppltdest
.It Sy Ppriv Ta Sy Ppsinfo
.It Sy Pputareg Ta Sy Prd_agent
.It Sy Pread Ta Sy Pread_string
.It Sy Preset_maps Ta Sy Psetbkpt
.It Sy Psecflags Ta Sy Psetcred
.It Sy Psetfault Ta Sy Psetflags
.It Sy Psetpriv Ta Sy Psetrun
.It Sy Psetsignal Ta Sy Psetsysentry
.It Sy Psetsysexit Ta Sy Psetwapt
.It Sy Psetzoneid Ta Sy Psignal
.It Sy Pstate Ta Sy Pstatus
.It Sy Pstop Ta Sy Pstopstatus
.It Sy Psync Ta Sy Psysentry
.It Sy Psysexit Ta Sy Puname
.It Sy Punsetflags Ta Sy Pupdate_maps
.It Sy Pupdate_syms Ta Sy Pwait
.It Sy Pwrite Ta Sy Pxecbkpt
.It Sy Pxecwapt Ta Sy Pxlookup_by_addr
.It Sy Pxlookup_by_addr_resolved Ta Sy Pxlookup_by_name
.It Sy Pzonename Ta Sy Pzonepath
.It Sy Pzoneroot Ta
.El
.Ss Thread interrogation and manipulation
The following routines obtain information about a thread and allow
manipulation of the thread itself.
.Bl -column -offset indent ".Sy Pmapping_iter_resolved" ".Sy Psymbol_iter_by_add
.It Sy Lalt_stack Ta Sy Lclearfault
.It Sy Lclearsig Ta Sy Lctlfd
.It Sy Ldstop Ta Sy Lgetareg
.It Sy Lmain_stack Ta Sy Lprochandle
.It Sy Lpsinfo Ta Sy Lputareg
.It Sy Lsetrun Ta Sy Lstack
.It Sy Lstate Ta Sy Lstatus
.It Sy Lstop Ta Sy Lsync
.It Sy Lwait Ta Sy Lxecbkpt
.It Sy Lxecwapt Ta ""
.El
```

```
.Ss System Call Injection
The following routines are used to inject specific system calls and have
them run in the context of a process.
.Bl -column -offset indent ".Sy Pmapping_iter_resolved" ".Sy Psymbol_iter_by_add
.It Sy pr_access Ta Sy pr_close
.It Sy pr_creat Ta Sy pr_door_info
.It Sy pr_exit Ta Sy pr_fcntl
.It Sy pr_fstat Ta Sy pr_fstat64
.It Sy pr_fstatvfs Ta Sy pr_getitimer
.It Sy pr_getpeername Ta Sy pr_getpeerucred
.It Sy pr_getprojid Ta Sy pr_getrctl
.It Sy pr_getrlimit Ta Sy pr_getrlimit64
.It Sy pr_getsockname Ta Sy pr_getsockopt
.It Sy pr_gettaskid Ta Sy pr_getzoneid
.It Sy pr_ioctl Ta Sy pr_link
.It Sy pr_llseek Ta Sy pr_lseek
.It Sy pr_lstat Ta Sy pr_lstat64
.It Sy pr_memcntl Ta Sy pr_meminfo
.It Sy pr_mmap Ta Sy pr_munmap
.It Sy pr_open Ta Sy pr_processor_bind
.It Sy pr_rename Ta Sy pr_setitimer
.It Sy pr_setrctl Ta Sy pr_setrlimit
.It Sy pr_setrlimit64 Ta Sy pr_settaskid
.It Sy pr_sigaction Ta Sy pr_stat
.It Sy pr_stat64 Ta Sy pr_statvfs
.It Sy pr_unlink Ta Sy pr_waitid
.El
.Ss Iteration routines
These routines are used to iterate over the contents of a process.
.Bl -column -offset indent ".Sy Pmapping_iter_resolved" ".Sy Psymbol_iter_by_add
.It Sy Penv_iter Ta Sy Plwp_iter
.It Sy Plwp_iter_all Ta Sy Pmapping_iter
.It Sy Pmapping_iter_resolved Ta Sy Pobject_iter
.It Sy Pobject_iter_resolved Ta Sy Pstack_iter
.It Sy Psymbol_iter Ta Sy Psymbol_iter_by_addr
.It Sy Psymbol_iter_by_lmid Ta Sy Psymbol_iter_by_name
.It Sy Pxsymbol_iter Ta Sy Pfdinfo_iter
.El
.Ss Utility routines
The following routines are utilities that are useful to consumers of the
library.
.Bl -column -offset indent ".Sy Pmapping_iter_resolved" ".Sy Psymbol_iter_by_add
.It Sy Perror_printf Ta Sy proc_arg_grab
.It Sy proc_arg_psinfo Ta Sy proc_arg_xgrab
.It Sy proc_arg_xpsinfo Ta Sy proc_content2str
.It Sy proc_finistdio Ta Sy proc_fltname
.It Sy proc_fltset2str Ta Sy proc_flushstdio
.It Sy proc_get_auxv Ta Sy proc_get_cred
.It Sy proc_get_priv Ta Sy proc_get_psinfo
.It Sy proc_get_status Ta Sy proc_initstdio
.It Sy proc_lwp_in_set Ta Sy proc_lwp_range_valid
.It Sy proc_signame Ta Sy proc_sigset2str
.It Sy proc_str2content Ta Sy proc_str2flt
.It Sy proc_str2fltset Ta Sy proc_str2sig
.It Sy proc_str2sigset Ta Sy proc_str2sys
.It Sy proc_str2sysset Ta Sy proc_sysname
.It Sy proc_sysset2str Ta Sy proc_unctrl_psinfo
.It Sy proc_walk Ta ""
.El
.Ss x86 Specific Routines
The following routines are specific to the x86, 32-bit and 64-bit,
versions of the
.Nm
library.
.Bl -column -offset indent ".Sy Pmapping_iter_resolved" ".Sy Psymbol_iter_by_add
.It Sy Pldt Ta Sy proc_get_ldt
```

```
.El
.Ss SPARC specific Routines
The following functions are specific to the SPARC, 32-bit and 64-bit,
versions of the
.Nm
library.
.Bl -column -offset indent ".Sy Pmapping_iter_resolved" ".Sy Psymbol_iter_by_add
.It Sy Plwp_getgwindows Ta Sy Plwp_getxregs
.It Sy Plwp_setxregs Ta Sy ""
.El
.Pp
The following functions are specific to the 64-bit SPARC version of the
.Nm
library.
.Bl -column -offset indent ".Sy Pmapping_iter_resolved" ".Sy Psymbol_iter_by_add
.It Sy Plwp_getasrs Ta Sy Plwp_setasrs
.El
.Sh PROCESS STATES
Every process handle that exists in
.Nm
has a state. In some cases, such as for core files, these states are
static. In other cases, such as handles that correspond to a
running process or a created process, these states are dynamic and
change based on actions taken in the library. The state can be obtained
with the
.Xr Pstate 3PROC
function.
.Pp
The various states are:
.Bl -tag -width Dv -offset indent
.It Dv PS_RUN
An actively running process. This may be a process that was obtained
by creating it with functions such as
.Xr Pcreate 3PROC
or by grabbing an existing process such as
.Xr Pgrab 3PROC .
.It Dv PS_STOP
An active process that is no longer executing. A process may stop for
many reasons such as an explicit stop request (through
.Xr pstop 1
for example) or if a tracing event is hit.
.Pp
The reason a process is stopped may be obtained through the thread’s
.Sy lwpstatus_t
structure read directly from /proc or obtained through the
.Xr Lstatus 3PROC
function.
.It Dv PS_LOST
Control over the process has been lost. This may happen when the
process executes a new image requiring a different set of privileges.
To resume control call
.Xr Preopen 3PROC . For more information on losing control of a process,
see
.Xr proc 4 .
.It DV PS_UNDEAD
A zombie process. It has terminated, but it has not been cleaned up
yet by its parent. For more on the conditions of becoming a zombie,
see
.Xr exec 2 .
.It DV_PS_DEAD
Processes in this state are always core files. See the earlier section
.Sx Core Files
for more information on working with core files.
.It Dv PS_IDLE
A process that has never been run. This is always the case for handles
that refer to files as the files cannot be executed. Those process
```

```
handles are obtained through calling
.Xr Pgrab_file 3PROC .
.El
.Pp
Many functions relating to tracing processes, for example
.Xr Psignal 3PROC ,
.Xr Psetsignal 3PROC ,
.Xr Psetfault 3PROC ,
.Xr Psetentry 3PROC ,
and others, mention that they only act upon
.Em Active Processes .
This specifically refers to processes whose state are in
.Dv PS_RUN
and
.Dv PS_STOP .
Process handles in the other states have no notion of settable tracing
flags, though core files
.Pf ( type Dv PS_DEAD ) ,
=======
may have a read-only snapshot of their tracing settings available.
.Sh TYPES
The
.Nm
library uses many types that come from the /proc file system
.Pf ( Xr proc 4 )
and the ELF format
.Pf ( Xr elf 3ELF ) .
However, it also defines the following types:
.Pp
.Sy struct ps_prochandle
.Pp
The
.Sy struct ps_prochandle
is an opaque handle to the library and the core element of control for a
process. Consumers obtain pointers to a handle through the use of the
.Fn Pcreate ,
.Fn Pgrab ,
and related functions. When a caller is done with a handle, then it
should call one of the
.Fn Pfree
and
.Fn Prelease
functions to relinquish the handle, release associated resources, and
potentially set the process to run again.
.Pp
.Sy struct ps_lwphandle
.Pp
The
.Sy struct ps_lwphandle
is analogous to the
.Sy struct ps_prochandle ,
but it represents the control of an individual thread, rather than a
process. Consumers obtain pointers to a handle through the
.Fn Lgrab
function and relinquish it with the
.Fn Lfree
function.
.Pp
.Sy core_content_t
.Pp
The
.Sy core_content_t
is a value which describes the various content types of core files.
These are used in functions such as
.Xr Pcontent 3PROC
and
```

```
.Xr Pgcore 3PROC
to describe and control the types of content that get included. Various
content types may be included together through a bitwise-inclusive-OR.
The default system core contents are controlled with the
.Xr coreadm 1M
tool. The following table lists the current set of core contents in the
system, though the set may increase over time. The string after the
macro is the human readable string that corresponds with the constant
and is used by
.Xr coreadm 1M ,
.Xr proc_content2str 3PROC ,
and
.Xr proc_str2content 3PROC .
.Bl -tag -offset indent -width indent
.It Dv CC_CONTENT_STACK ("stack")
The contents include the process stack. Note, this only covers the main
thread’s stack. The stack of other threads is covered by
.Dv CC_CONTENT_ANON .
.It Dv CC_CONTENT_HEAP ("heap")
The contents include the process heap.
.It Dv CC_CONTENT_SHFILE ("shfile")
The contents include shared mappings that are backed by files (e.g.
mapped through
.Xr mmap 2
with the
.Dv MAP_SHARED
flag).
.It Dv CC_CONTENT_SHANNON ("shannon")
The contents include shared mappings that are backed by anonymous memory
(e.g. mapped through
.Xr mmap 2
with the
.Dv MAP_SHARED
and
.Dv MAP_ANON
flags).
.It Dv CC_CONTENT_RODATA ("rodata")
The contents include private read-only file mappings, such as shared
library text.
.It Dv CC_CONTENT_ANON ("anon")
The contents include private anonymous mappings. This includes the
stacks of threads which are not the main thread.
.It Dv CC_CONTENT_SHM ("shm")
The contents include system V shared memory.
.It Dv CC_CONTENT_ISM ("ism")
The contents include ISM (intimate shared memory) mappings.
.It Dv CC_CONTENT_DISM ("dism")
The contents include DISM (dynamic shared memory) mappings.
.It Dv CC_CONTENT_CTF ("ctf")
The contents include
.Xr ctf 4
(Compact C Type Format) information. Note, not all objects in the
process may have CTF information available.
.It Dv CC_CONTENT_SYMTAB ("symtab")
The contents include the symbol table. Note, not all objects in the
process may have a symbol table available.
.It Dv CC_CONTENT_ALL ("all")
This value indicates that all of the above content values are present.
Note that additional values may be added in the future, in which case
the value of the symbol will be updated to include them. Comparisons
with
.Dv CC_CONTENT_ALL
should validate all the expected bits are set by an expression such as
.Li (c & CC_CONTENT_ALL) == CC_CONTENT_ALL .
.It Dv CC_CONTENT_NONE ("none")
This value indicates that there is no content present.
```

```
.It Dv CC_CONTENT_DEFAULT ("default")
The content includes the following set of default values:
.Dv CC_CONTENT_STACK ,
.Dv CC_CONTENT_HEAP ,
.Dv CC_CONTENT_ISM ,
.Dv CC_CONTENT_DISM ,
.Dv CC_CONTENT_SHM ,
.Dv CC_CONTENT_SHANON ,
.Dv CC_CONTENT_TEXT ,
.Dv CC_CONTENT_DATA ,
.Dv CC_CONTENT_RODATA ,
.Dv CC_CONTENT_ANON ,
.Dv CC_CONTENT_CTF ,
and
.Dv CC_CONTENT_SYMTAB.
Note that the default may change. Comparisons with CC_CONTENT_DEFAULT
should validate that all of the expected bits are set with an expression
such as
.Li (c\ &\ CC_CONTENT_DEFAULT)\ ==\ CC_CONTENT_DEFAULT.
.It Dv CC_CONTENT_INVALID
This indicates that the contents are invalid.
.El
.Pp
.Sy prfdinfo_t
.Pp
The
.Sy prfdinfo_t
structure is used with the
.Fn Pfdinfo_iter
function which describes information about a file descriptor. The
structure is defined as follows:
.Bd -literal
typedef struct prfdinfo {
        int             pr_fd;
        mode_t          pr_mode;
        uid_t           pr_uid;
        gid_t           pr_gid;
        major_t         pr_major;       /* think stat.st_dev */
        minor_t         pr_minor;
        major_t         pr_rmajor;      /* think stat.st_rdev */
        minor_t         pr_rminor;
        ino64_t         pr_ino;
        off64_t         pr_offset;
        off64_t         pr_size;
        int             pr_fileflags;   /* fcntl(F_GETXFL), etc */
        int             pr_fdflags;     /* fcntl(F_GETFD), etc. */
        char            pr_path[MAXPATHLEN];
} prfdinfo_t;
```
______***unchanged_portion_omitted_***

****
1637 Wed Jun 15 19:31:44 2016
new/usr/src/man/man3proc/Psecflags.3proc
Code review comments from jeffpc
****

```
.\"
.\" 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 2016, Richard Lowe.
.\"
.Dd June 06, 2016
.Dd May 14, 2016
.Dt PSECFLAGS 3PROC
.Os
.Sh NAME
.Nm Psecflags ,
.Nm Psecflags_free
.Nd get and free process security flags
.Sh SYNOPSIS
.Lb libproc
.In libproc.h
.Ft int
.Fo Psecflags
.Fo Ppriv
.Fa "struct ps_prochandle *P"
.Fa "prsecflags_t **psf"
.Fc
.Ft void
.Fo Psecflags_free
.Fo Ppriv_free
.Fa "struct ps_prochandle *P"
.Fa "prsecflags_t *psf"
.Fc
.Sh DESCRIPTION
The
.Fn Psecflags
function obtains the security flags of the process handle
.Fa P .
The security flags structure will be dynamically allocated and a pointer to it
will be placed in
.Fa psf .
It must be released with a call to
.Fn Psecflags_free .
The definition of the
.Sy prsecflags_t
structure is documented in
.Xr proc 4 .
.Pp
The
.Fn Psecflags_free
function releases the storage in
.Fa psf
that was allocated as a result of calling
.Fn Psecflags .
.Sh RETURN VALUES
Upon successful completion, the
.Fn Psecflags
function returns
```

```
.Sy 0
and
.Fa psf
is updated with a pointer to the allocated security flags. Otherwise,
is updated with a pointer to the allocated privilege set. Otherwise,
.Sy -1
is returned and
.Fa psf
is not updated.
.Sh INTERFACE STABILITY
.Sy Uncommitted
.Sh MT-LEVEL
See
.Sy LOCKING
in
.Xr libproc 3LIB .
.Sh SEE ALSO
.Xr libproc 3LIB ,
.Xr proc 4 ,
.Xr security-flags 5
```

****
   16868 Wed Jun 15 19:31:45 2016
new/usr/src/man/man4/core.4
Code review comments from jeffpc
****

```
'\" te
.\" Copyright (C) 2008, Sun Microsystems, Inc. All Rights Reserved.
.\" Copyright 2012 DEY Storage Systems, Inc.  All rights reserved.
.\" Copyright (c) 2013, Joyent, Inc. All rights reserved.
.\" Copyright 1989 AT&T
.\" The contents of this file are subject to the terms of the Common Development
.\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
.\" When distributing Covered Code, include this CDDL HEADER in each file and in
.TH CORE 4 "Jun 6, 2016"
.TH CORE 4 "Jul 23, 2015"
.SH NAME
core \- process core file
.SH DESCRIPTION
.LP
The operating system writes out a core file for a process when the process is
terminated due to receiving certain signals. A core file is a disk copy of the
contents of the process address space at the time the process received the
signal, along with additional information about the state of the process. This
information can be consumed by a debugger. Core files can also be generated by
applying the \fBgcore\fR(1) utility to a running process.
.sp
.LP
Typically, core files are produced following abnormal termination of a process
resulting from a bug in the corresponding application. Whatever the cause, the
core file itself provides invaluable information to the programmer or support
engineer to aid in diagnosing the problem. The core file can be inspected using
a debugger such as \fBdbx\fR(1) or \fBmdb\fR(1) or by applying one of the
\fBproc\fR(1) tools.
.sp
.LP
The operating system attempts to create up to two core files for each
abnormally terminating process, using a global core file name pattern and a
per-process core file name pattern. These patterns are expanded to determine
the pathname of the resulting core files, and can be configured by
\fBcoreadm\fR(1M). By default, the global core file pattern is disabled and not
used, and the per-process core file pattern is set to \fBcore\fR. Therefore, by
default, the operating system attempts to create a core file named \fBcore\fR
in the process's current working directory.
.sp
.LP
A process terminates and produces a core file whenever it receives one of the
signals whose default disposition is to cause a core dump. The list of signals
that result in generating a core file is shown in \fBsignal.h\fR(3HEAD).
Therefore, a process might not produce a core file if it has blocked or
modified the behavior of the corresponding signal. Additionally, no core dump
can be created under the following conditions:
.RS +4
.TP
.ie t \(bu
.el o
If normal file and directory access permissions prevent the creation or
modification of the per-process core file pathname by the current process user
and group ID. This test does not apply to the global core file pathname
because, regardless of the UID of the process dumping core, the attempt to
write the global core file is made as the superuser.
.RE
.RS +4
.TP
.ie t \(bu
.el o
Core files owned by the user \fBnobody\fR will not be produced. For example,
```

```
core files generated for the superuser on an NFS directory are owned by
\fBnobody\fR and are, therefore, not written.
.RE
.RS +4
.TP
.ie t \(bu
.el o
If the core file pattern expands to a pathname that contains intermediate
directory components that do not exist. For example, if the global pattern is
set to \fB/var/core/%n/core.%p\fR, and no directory \fB/var/core/`uname -n`\fR
has been created, no global core files are produced.
.RE
.RS +4
.TP
.ie t \(bu
.el o
If the destination directory is part of a filesystem that is mounted read-only.
.RE
.RS +4
.TP
.ie t \(bu
.el o
If the resource limit \fBRLIMIT_CORE\fR has been set to \fB0\fR for the
process, no per-process core file is produced. Refer to \fBsetrlimit\fR(2) and
\fBulimit\fR(1) for more information on resource limits.
.RE
.RS +4
.TP
.ie t \(bu
.el o
If the core file name already exists in the destination directory and is not a
regular file (that is, is a symlink, block or character special-file, and so
forth).
.RE
.RS +4
.TP
.ie t \(bu
.el o
If the kernel cannot open the destination file \fBO_EXCL\fR, which can occur if
same file is being created by another process simultaneously.
.RE
.RS +4
.TP
.ie t \(bu
.el o
If the process's effective user ID is different from its real user ID or if its
effective group ID is different from its real group ID. Similarly, set-user-ID
and set-group-ID programs do not produce core files as this could potentially
compromise system security. These processes can be explicitly granted
permission to produce core files using \fBcoreadm\fR(1M), at the risk of
exposing secure information.
.RE
.sp
.LP
The core file contains all the process information pertinent to debugging:
contents of hardware registers, process status, and process data. The format of
a core file is object file specific.
.sp
.LP
For ELF executable programs (see \fBa.out\fR(4)), the core file generated is
also an ELF file, containing ELF program and file headers. The \fBe_type\fR
field in the file header has type \fBET_CORE\fR. The program header contains an
entry for every segment that was part of the process address space, including
shared library segments. The contents of the mappings specified by
\fBcoreadm\fR(1M) are also part of the core image. Each program header has its
\fBp_memsz\fR field set to the size of the mapping. The program headers that
```

```
represent mappings whose data is included in the core file have their
\fBp_filesz\fR field set the same as \fBp_memsz\fR, otherwise \fBp_filesz\fR is
\fBzero\fR.
.sp
.LP
A mapping’s data can be excluded due to the core file content settings (see
\fBcoreadm\fR(1M)), due to a failure, or due to a signal received after
core dump initiation but before its completion. If the data is excluded
because of a failure, the program header entry will have the
\fBPF_SUNW_FAILURE\fR flag
set in its \fBp_flags\fR field; if the data is excluded because of a signal,
the segment’s \fBp_flags\fR field will have the \fBPF_SUNW_KILLED\fR
flag set.
.sp
.LP
The program headers of an \fBELF\fR core file also contain entries for two
\fBNOTE\fR segments, each containing several note entries as described below.
The note entry header and core file note type (\fBn_type\fR) definitions are
contained in <\fBsys/elf.h\fR>. The first \fBNOTE\fR segment exists for binary
compatibility with old programs that deal with core files. It contains
structures defined in <\fBsys/old_procfs.h\fR>. New programs should recognize
and skip this \fBNOTE\fR segment, advancing instead to the new \fBNOTE\fR
segment. The old \fBNOTE\fR segment is deleted from core files in a future
release.
.sp
.LP
The old \fBNOTE\fR segment contains the following entries. Each has entry name
\fB"CORE"\fR and presents the contents of a system structure:
.sp
.ne 2
.na
\fB\fBprpsinfo_t\fR\fR
.ad
.RS 16n
\fBn_type\fR: \fBNT_PRPSINFO\fR. This entry contains information of interest to
the \fBps\fR(1) command, such as process status, \fBCPU\fR usage, \fBnice\fR
value, controlling terminal, user-ID, process-ID, the name of the executable,
and so forth. The \fBprpsinfo_t\fR structure is defined in
<\fBsys/old_procfs.h\fR>.
.RE

.sp
.ne 2
.na
\fB\fBchar\fR array\fR
.ad
.RS 16n
\fBn_type\fR: \fBNT_PLATFORM\fR. This entry contains a string describing the
specific model of the hardware platform on which this core file was created.
This information is the same as provided by \fBsysinfo\fR(2) when invoked with
the command \fBSI_PLATFORM\fR.
.RE

.sp
.ne 2
.na
\fB\fBauxv_t\fR array\fR
.ad
.RS 16n
\fBn_type\fR: \fBNT_AUXV\fR. This entry contains the array of \fBauxv_t\fR
structures that was passed by the operating system as startup information to
the dynamic linker. Auxiliary vector information is defined in
<\fBsys/auxv.h\fR>.
.RE

.sp
```

```
.LP
Following these entries, for each active (non-zombie) light-weight process
(LWP) in the process, the old \fBNOTE\fR segment contains an entry with a
\fBprstatus_t\fR structure, plus other optionally-present entries describing
the LWP, as follows:
.sp
.ne 2
.na
\fB\fBprstatus_t\fR\fR
.ad
.RS 16n
\fBn_type\fR: \fBNT_PRSTATUS\fR. This structure contains things of interest to
a debugger from the operating system, such as the general registers, signal
dispositions, state, reason for stopping, process-ID, and so forth. The
\fBprstatus_t\fR structure is defined in <\fBsys/old_procfs.h\fR>.
.RE

.sp
.ne 2
.na
\fB\fBprfpregset_t\fR\fR
.ad
.RS 16n
\fBn_type\fR: \fBNT_PRFPREG\fR. This entry is present only if the \fBLWP\fR
used the floating-point hardware. It contains the floating-point registers. The
\fBprfpregset_t\fR structure is defined in <\fBsys/procfs_isa.h\fR>.
.RE

.sp
.ne 2
.na
\fB\fBgwindows_t\fR\fR
.ad
.RS 16n
\fBn_type\fR: \fBNT_GWINDOWS\fR. This entry is present only on a SPARC machine
and only if the system was unable to flush all of the register windows to the
stack. It contains all of the unspilled register windows. The \fBgwindows_t\fR
structure is defined in <\fBsys/regset.h\fR>.
.RE

.sp
.ne 2
.na
\fB\fBprxregset_t\fR\fR
.ad
.RS 16n
\fBn_type\fR: \fBNT_PRXREG\fR. This entry is present only if the machine has
extra register state associated with it. It contains the extra register state.
The \fBprxregset_t\fR structure is defined in <\fBsys/procfs_isa.h\fR>.
.RE

.sp
.LP
The new \fBNOTE\fR segment contains the following entries. Each has entry name
"\fBCORE\fR" and presents the contents of a system structure:
.sp
.ne 2
.na
\fB\fBpsinfo_t\fR\fR
.ad
.RS 20n
\fBn_type\fR: \fBNT_PSINFO\fR. This structure contains information of interest
to the \fBps\fR(1) command, such as process status, \fBCPU\fR usage, \fBnice\fR
value, controlling terminal, user-ID, process-ID, the name of the executable,
and so forth. The \fBpsinfo_t\fR structure is defined in <\fBsys/procfs.h\fR>.
.RE
```

```
.sp
.ne 2
.na
\fB\fBpstatus_t\fR\fR
.ad
.RS 20n
\fBn_type\fR: \fBNT_PSTATUS\fR. This structure contains things of interest to a
debugger from the operating system, such as pending signals, state, process-ID,
and so forth. The \fBpstatus_t\fR structure is defined in <\fBsys/procfs.h\fR>.
.RE

.sp
.ne 2
.na
\fB\fBchar\fR array\fR
.ad
.RS 20n
\fBn_type\fR: \fBNT_PLATFORM\fR. This entry contains a string describing the
specific model of the hardware platform on which this core file was created.
This information is the same as provided by \fBsysinfo\fR(2) when invoked with
the command \fBSI_PLATFORM\fR.
.RE

.sp
.ne 2
.na
\fB\fBauxv_t\fR array\fR
.ad
.RS 20n
\fBn_type\fR: \fBNT_AUXV\fR. This entry contains the array of \fBauxv_t\fR
structures that was passed by the operating system as startup information to
the dynamic linker. Auxiliary vector information is defined in
<\fBsys/auxv.h\fR>.
.RE

.sp
.ne 2
.na
\fB\fBstruct utsname\fR\fR
.ad
.RS 20n
\fBn_type\fR: \fBNT_UTSNAME\fR. This structure contains the system information
that would have been returned to the process if it had performed a
\fBuname\fR(2) system call prior to dumping core. The \fButsname\fR structure
is defined in <\fBsys/utsname.h\fR>.
.RE

.sp
.ne 2
.na
\fB\fBprcred_t\fR\fR
.ad
.RS 20n
\fBn_type\fR: \fBNT_PRCRED\fR. This structure contains the process credentials,
including the real, saved, and effective user and group IDs. The \fBprcred_t\fR
structure is defined in <\fBsys/procfs.h\fR>. Following the structure is an
optional array of supplementary group IDs. The total number of supplementary
group IDs is given by the \fBpr_ngroups\fR member of the \fBprcred_t\fR
structure, and the structure includes space for one supplementary group. If
\fBpr_ngroups\fR is greater than 1, there is \fBpr_ngroups - 1\fR \fBgid_t\fR
items following the structure; otherwise, there is no additional data.
.RE

.sp
.ne 2
```

```
.na
\fB\fBchar array\fR\fR
.ad
.RS 20n
\fBn_type\fR: \fBNT_ZONENAME\fR. This entry contains a string which describes
the name of the zone in which the process was running. See \fBzones\fR(5). The
information is the same as provided by \fBgetzonenamebyid\fR(3C) when invoked
with the numerical ID returned by \fBgetzoneid\fR(3C).
.RE

.sp
.ne 2
.na
\fB\fBprfdinfo_t\fR\fR
.ad
.RS 20n
\fBn_type\fR: \fBNT_FDINFO\fR. This structure contains information about
any open file descriptors, including the path, flags, and
\fBstat\fR(2) information.  The \fBprfdinfo_t\fR structure is defined in
<\fBsys/procfs.h\fR>.
.RE

.sp
.ne 2
.na
\fB\fBstruct ssd\fR array\fR
.ad
.RS 20n
\fBn_type\fR: \fBNT_LDT\fR. This entry is present only on an 32-bit x86 machine
and only if the process has set up a Local Descriptor Table (LDT). It contains
an array of structures of type \fBstruct ssd\fR, each of which was typically
used to set up the \fB%gs\fR segment register to be used to fetch the address
of the current thread information structure in a multithreaded process. The
\fBssd\fR structure is defined in <\fBsys/sysi86.h\fR>.
.RE

.sp
.ne 2
.na
\fB\fBcore_content_t\fR\fR
.ad
.RS 20n
\fBn_type\fR: \fBNT_CONTENT\fR. This optional entry indicates which parts of
the process image are specified to be included in the core file. See
\fBcoreadm\fR(1M).
.RE

.sp
.LP
Following these entries, for each active and zombie \fBLWP\fR in the process,
the new \fBNOTE\fR segment contains an entry with an \fBlwpsinfo_t\fR structure
plus, for a non-zombie LWP, an entry with an \fBlwpstatus_t\fR structure, plus
other optionally-present entries describing the LWP, as follows. A zombie LWP
is a non-detached LWP that has terminated but has not yet been reaped by
another LWP in the same process.
.sp
.ne 2
.na
\fB\fBlwpsinfo_t\fR\fR
.ad
.RS 15n
\fBn_type\fR: \fBNT_LWPSINFO\fR. This structure contains information of
interest to the \fBps\fR(1) command, such as \fBLWP\fR status, \fBCPU\fR usage,
\fBnice\fR value, \fBLWP-ID\fR, and so forth. The \fBlwpsinfo_t\fR structure is
defined in <\fBsys/procfs.h\fR>. This is the only entry present for a zombie
LWP.
```

```
.RE

.sp
.ne 2
.na
\fB\fBlwpstatus_t\fR\fR
.ad
.RS 15n
\fBn_type\fR: \fBNT_LWPSTATUS\fR. This structure contains things of interest to
a debugger from the operating system, such as the general registers, the
floating point registers, state, reason for stopping, \fBLWP-ID\fR, and so
forth. The \fBlwpstatus_t\fR structure is defined in <\fBsys/procfs.h>\fR>.
.RE

.sp
.ne 2
.na
\fB\fBgwindows_t\fR\fR
.ad
.RS 15n
\fBn_type\fR: \fBNT_GWINDOWS\fR. This entry is present only on a SPARC machine
and only if the system was unable to flush all of the register windows to the
stack. It contains all of the unspilled register windows. The \fBgwindows_t\fR
structure is defined in \fB<sys/regset.h>\fR\&.
.RE

.sp
.ne 2
.na
\fB\fBprxregset_t\fR\fR
.ad
.RS 15n
\fBn_type\fR: \fBNT_PRXREG\fR. This entry is present only if the machine has
extra register state associated with it. It contains the extra register state.
The \fBprxregset_t\fR structure is defined in \fB<sys/procfs_isa.h>\fR\&.
.RE

.sp
.ne 2
.na
\fB\fBasrset_t\fR\fR
.ad
.RS 15n
\fBn_type\fR: \fBNT_ASRS\fR. This entry is present only on a SPARC V9 machine
and only if the process is a 64-bit process. It contains the ancillary state
registers for the \fBLWP.\fR The \fBasrset_t\fR structure is defined in
\fB<sys/regset.h>\fR\&.
.RE

.sp
.ne 2
.na
\fB\fBpsinfo_t\fR\fR
.ad
.RS 15n
\fBn_type\fR: \fBNT_SPYMASTER\fR. This entry is present only for an agent
LWP and contains the \fBpsinfo_t\fR of the process that created the agent
LWP. See the \fBproc\fR(4) description of the \fBspymaster\fR entry for
more details.
.RE

.sp
.ne 2
.na
\fB\fBprsecflags_t\fR\fR
.ad
```

```
.RS 15n
\fBn_type\fR: \fbNT_SECFLAGS\fR.  This entry contains the process
security-flags, see \fBsecurity-flags\fR(5), \fBproc\fR(4), and
\fBpsecflags\fR(1M) for more information.
.RE

.sp
.LP
Depending on the \fBcoreadm\fR(1M) settings, the section header of an ELF core
file can contain entries for CTF, symbol table, and string table sections. The
\fBsh_addr\fR fields are set to the base address of the first mapping of the
load object that they came from to. This can be used to match those sections
with the corresponding load object.
.sp
.LP
The size of the core file created by a process can be controlled by the user
(see \fBgetrlimit\fR(2)).
.SH SEE ALSO
.LP
\fBelfdump\fR(1), \fBgcore\fR(1), \fBmdb\fR(1), \fBproc\fR(1), \fBps\fR(1),
\fBcoreadm\fR(1M), \fBgetrlimit\fR(2), \fBsetrlimit\fR(2), \fBsetuid\fR(2),
\fBsysinfo\fR(2), \fBuname\fR(2), \fBgetzonenamebyid\fR(3C),
\fBgetzoneid\fR(3C), \fBelf\fR(3ELF), \fBsignal.h\fR(3HEAD), \fBa.out\fR(4),
\fBproc\fR(4), \fBzones\fR(5), \fBsecurity-flags\fR(5)
.sp
.LP
\fIANSI C Programmer’s Guide\fR
```

```
**********************************************************
   99687 Wed Jun 15 19:31:46 2016
new/usr/src/man/man4/proc.4
Code review comments from jeffpc
**********************************************************
'\" te
.\" Copyright 1989 AT&T
.\" Copyright (c) 2006, Sun Microsystems, Inc. All Rights Reserved.
.\" Copyright (c) 2013, Joyent, Inc. All rights reserved.
.\" The contents of this file are subject to the terms of the Common Development
.\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
.\" When distributing Covered Code, include this CDDL HEADER in each file and in
.TH PROC 4 "Jun 6, 2016"
.TH PROC 4 "Jul 23, 2015"
.SH NAME
proc \- /proc, the process file system
.SH DESCRIPTION
.LP
\fB/proc\fR is a file system that provides access to the state of each process
and light-weight process (lwp) in the system. The name of each entry in the
\fB/proc\fR directory is a decimal number corresponding to a process-ID. These
entries are themselves subdirectories. Access to process state is provided by
additional files contained within each subdirectory; the hierarchy is described
more completely below. In this document, ``\fB/proc\fR file'' refers to a
non-directory file within the hierarchy rooted at \fB/proc\fR. The owner of
each \fB/proc\fR file and subdirectory is determined by the user-ID of the
process.
.sp
.LP
\fB/proc\fR can be mounted on any mount point, in addition to the standard
\fB/proc\fR mount point, and can be mounted several places at once. Such
additional mounts are allowed in order to facilitate the confinement of
processes to subtrees of the file system via \fBchroot\fR(1M) and yet allow
such processes access to commands like \fBps\fR(1).
.sp
.LP
Standard system calls are used to access \fB/proc\fR files: \fBopen\fR(2),
\fBclose\fR(2), \fBread\fR(2), and \fBwrite\fR(2) (including \fBreadv\fR(2),
\fBwritev\fR(2), \fBpread\fR(2), and \fBpwrite\fR(2)). Most files describe
process state and can only be opened for reading. \fBctl\fR and \fBlwpctl\fR
(control) files permit manipulation of process state and can only be opened for
writing. \fBas\fR (address space) files contain the image of the running
process and can be opened for both reading and writing. An open for writing
allows process control; a read-only open allows inspection but not control. In
this document, we refer to the process as open for reading or writing if any of
its associated \fB/proc\fR files is open for reading or writing.
.sp
.LP
In general, more than one process can open the same \fB/proc\fR file at the
same time. \fIExclusive\fR \fIopen\fR is an advisory mechanism provided to
allow controlling processes to avoid collisions with each other. A process can
obtain exclusive control of a target process, with respect to other cooperating
processes, if it successfully opens any \fB/proc\fR file in the target process
for writing (the \fBas\fR or \fBctl\fR files, or the \fBlwpctl\fR file of any
lwp) while specifying \fBO_EXCL\fR in the \fBopen\fR(2). Such an open will fail
if the target process is already open for writing (that is, if an \fBas\fR,
\fBctl\fR, or \fBlwpctl\fR file is already open for writing). There can be any
number of concurrent read-only opens; \fBO_EXCL\fR is ignored on opens for
reading. It is recommended that the first open for writing by a controlling
process use the \fBO_EXCL\fR flag; multiple controlling processes usually
result in chaos.
.sp
.LP
If a process opens one of its own \fB/proc\fR files for writing, the open
succeeds regardless of \fBO_EXCL\fR and regardless of whether some other
process has the process open for writing. Self-opens do not count when another
```

```
process attempts an exclusive open. (A process cannot exclude a debugger by
opening itself for writing and the application of a debugger cannot prevent a
process from opening itself.) All self-opens for writing are forced to be
close-on-exec (see the \fBF_SETFD\fR operation of \fBfcntl\fR(2)).
.sp
.LP
Data may be transferred from or to any locations in the address space of the
traced process by applying \fBlseek\fR(2) to position the \fBas\fR file at the
virtual address of interest followed by \fBread\fR(2) or \fBwrite\fR(2) (or by
using \fBpread\fR(2) or \fBpwrite\fR(2) for the combined operation). The
address-map files \fB/proc/\fR\fIpid\fR\fB/map\fR and
\fB/proc/\fR\fIpid\fR\fB/xmap\fR can be read to determine the accessible areas
(mappings) of the address space. \fBI/O\fR transfers may span contiguous
mappings. An \fBI/O\fR request extending into an unmapped area is truncated at
the boundary. A write request beginning at an unmapped virtual address fails
with \fBEIO\fR; a read request beginning at an unmapped virtual address returns
zero (an end-of-file indication).
.sp
.LP
Information and control operations are provided through additional files.
\fB<procfs.h>\fR contains definitions of data structures and message formats
used with these files. Some of these definitions involve the use of sets of
flags. The set types \fBsigset_t\fR, \fBfltset_t\fR, and \fBsysset_t\fR
correspond, respectively, to signal, fault, and system call enumerations
defined in \fB<sys/signal.h>\fR, \fB<sys/fault.h>\fR, and
\fB<sys/syscall.h>\fR\&. Each set type is large enough to hold flags for its
own enumeration. Although they are of different sizes, they have a common
structure and can be manipulated by these macros:
.sp
.in +2
.nf
prfillset(&set);             /* turn on all flags in set */
premptyset(&set);            /* turn off all flags in set */
praddset(&set, flag);        /* turn on the specified flag */
prdelset(&set, flag);        /* turn off the specified flag */
r = prismember(&set, flag);  /* != 0 iff flag is turned on */
.fi
.in -2

.sp
.LP
One of \fBprfillset()\fR or \fBpremptyset()\fR must be used to initialize
\fBset\fR before it is used in any other operation. \fBflag\fR must be a member
of the enumeration corresponding to \fBset\fR.
.sp
.LP
Every process contains at least one \fIlight-weight process\fR, or \fIlwp\fR.
Each lwp represents a flow of execution that is independently scheduled by the
operating system. All lwps in a process share its address space as well as many
other attributes. Through the use of \fBlwpctl\fR and \fBctl\fR files as
described below, it is possible to affect individual lwps in a process or to
affect all of them at once, depending on the operation.
.sp
.LP
When the process has more than one lwp, a representative lwp is chosen by the
system for certain process status files and control operations. The
representative lwp is a stopped lwp only if all of the process's lwps are
stopped; is stopped on an event of interest only if all of the lwps are so
stopped (excluding \fBPR_SUSPENDED\fR lwps); is in a \fBPR_REQUESTED\fR stop
only if there are no other events of interest to be found; or, failing
everything else, is in a \fBPR_SUSPENDED\fR stop (implying that the process is
deadlocked). See the description of the \fBstatus\fR file for definitions of
stopped states. See the \fBPCSTOP\fR control operation for the definition of
``event of interest''.
.sp
.LP
```

```
The representative lwp remains fixed (it will be chosen again on the next
operation) as long as all of the lwps are stopped on events of interest or are
in a \fBPR_SUSPENDED\fR stop and the \fBPCRUN\fR control operation is not
applied to any of them.
.sp
.LP
When applied to the process control file, every \fB/proc\fR control operation
that must act on an lwp uses the same algorithm to choose which lwp to act
upon. Together with synchronous stopping (see \fBPCSET\fR), this enables a
debugger to control a multiple-lwp process using only the process-level status
and control files if it so chooses. More fine-grained control can be achieved
using the lwp-specific files.
.sp
.LP
The system supports two process data models, the traditional 32-bit data model
in which ints, longs and pointers are all 32 bits wide (the ILP32 data model),
and on some platforms the 64-bit data model in which longs and pointers, but
not ints, are 64 bits in width (the LP64 data model). In the LP64 data model
some system data types, notably \fBsize_t\fR, \fBoff_t\fR, \fBtime_t\fR and
\fBdev_t\fR, grow from 32 bits to 64 bits as well.
.sp
.LP
The \fB/proc\fR interfaces described here are available to both 32-bit and
64-bit controlling processes. However, many operations attempted by a 32-bit
controlling process on a 64-bit target process will fail with \fBEOVERFLOW\fR
because the address space range of a 32-bit process cannot encompass a 64-bit
process or because the data in some 64-bit system data type cannot be
compressed to fit into the corresponding 32-bit type without loss of
information. Operations that fail in this circumstance include reading and
writing the address space, reading the address-map files, and setting the
target process’s registers. There is no restriction on operations applied by a
64-bit process to either a 32-bit or a 64-bit target processes.
.sp
.LP
The format of the contents of any \fB/proc\fR file depends on the data model of
the observer (the controlling process), not on the data model of the target
process. A 64-bit debugger does not have to translate the information it reads
from a \fB/proc\fR file for a 32-bit process from 32-bit format to 64-bit
format. However, it usually has to be aware of the data model of the target
process. The \fBpr_dmodel\fR field of the \fBstatus\fR files indicates the
target process’s data model.
.sp
.LP
To help deal with system data structures that are read from 32-bit processes, a
64-bit controlling program can be compiled with the C preprocessor symbol
\fB_SYSCALL32\fR defined before system header files are included. This makes
explicit 32-bit fixed-width data structures (like \fBcstruct stat32\fR) visible
to the 64-bit program. See \fBtypes32.h\fR(3HEAD).
.SH DIRECTORY STRUCTURE
.LP
At the top level, the directory \fB/proc\fR contains entries each of which
names an existing process in the system. These entries are themselves
directories. Except where otherwise noted, the files described below can be
opened for reading only. In addition, if a process becomes a \fIzombie\fR (one
that has exited but whose parent has not yet performed a \fBwait\fR(3C) upon
it), most of its associated \fB/proc\fR files disappear from the hierarchy;
subsequent attempts to open them, or to read or write files opened before the
process exited, will elicit the error \fBENOENT\fR.
.sp
.LP
Although process state and consequently the contents of \fB/proc\fR files can
change from instant to instant, a single \fBread\fR(2) of a \fB/proc\fR file is
guaranteed to return a sane representation of state; that is, the read will be
atomic with respect to the state of the process. No such guarantee applies to
successive reads applied to a \fB/proc\fR file for a running process. In
addition, atomicity is not guaranteed for \fBI/O\fR applied to the \fBas\fR
```

```
(address-space) file for a running process or for a process whose address space
contains memory shared by another running process.
.sp
.LP
A number of structure definitions are used to describe the files. These
structures may grow by the addition of elements at the end in future releases
of the system and it is not legitimate for a program to assume that they will
not.
.SH STRUCTURE OF \fB/proc/\fR\fIpid\fR
.LP
A given directory \fB/proc/\fR\fIpid\fR contains the following entries. A
process can use the invisible alias \fB/proc/self\fR if it wishes to open one
of its own \fB/proc\fR files (invisible in the sense that the name ‘‘self’’
does not appear in a directory listing of \fB/proc\fR obtained from
\fBls\fR(1), \fBgetdents\fR(2), or \fBreaddir\fR(3C)).
.SS "contracts"
.LP
A directory containing references to the contracts held by the process. Each
entry is a symlink to the contract’s directory under \fB/system/contract\fR.
See \fBcontract\fR(4).
.SS "as"
.LP
Contains the address-space image of the process; it can be opened for both
reading and writing. \fBlseek\fR(2) is used to position the file at the virtual
address of interest and then the address space can be examined or changed
through \fBread\fR(2) or \fBwrite\fR(2) (or by using \fBpread\fR(2) or
\fBpwrite\fR(2) for the combined operation).
.SS "ctl"
.LP
A write-only file to which structured messages are written directing the system
to change some aspect of the process’s state or control its behavior in some
way. The seek offset is not relevant when writing to this file. Individual lwps
also have associated \fBlwpctl\fR files in the lwp subdirectories. A control
message may be written either to the process’s \fBctl\fR file or to a specific
\fBlwpctl\fR file with operation-specific effects. The effect of a control
message is immediately reflected in the state of the process visible through
appropriate status and information files. The types of control messages are
described in detail later. See \fBCONTROL MESSAGES\fR.
.SS "status"
.LP
Contains state information about the process and the representative lwp. The
file contains a \fBpstatus\fR structure which contains an embedded
\fBlwpstatus\fR structure for the representative lwp, as follows:
.sp
.in +2
.nf
typedef struct pstatus {
     int pr_flags;            /* flags (see below) */
     int pr_nlwp;             /* number of active lwps in the process */
     int pr_nzomb;            /* number of zombie lwps in the process */
     pid_tpr_pid;             /* process id */
     pid_tpr_ppid;            /* parent process id */
     pid_tpr_pgid;            /* process group id */
     pid_tpr_sid;             /* session id */
     id_t pr_aslwpid;         /* obsolete */
     id_t pr_agentid;         /* lwp-id of the agent lwp, if any */
     sigset_t pr_sigpend;     /* set of process pending signals */
     uintptr_t pr_brkbase;    /* virtual address of the process heap */
     size_t pr_brksize;       /* size of the process heap, in bytes */
     uintptr_t pr_stkbase;    /* virtual address of the process stack */
     size_tpr_stksize;        /* size of the process stack, in bytes */
     timestruc_t pr_utime;    /* process user cpu time */
     timestruc_t pr_stime;    /* process system cpu time */
     timestruc_t pr_cutime;   /* sum of children’s user times */
     timestruc_t pr_cstime;   /* sum of children’s system times */
     sigset_t pr_sigtrace;    /* set of traced signals */
```

```
        fltset_t pr_flttrace;    /* set of traced faults */
        sysset_t pr_sysentry;    /* set of system calls traced on entry */
        sysset_t pr_sysexit;     /* set of system calls traced on exit */
        char pr_dmodel;          /* data model of the process */
        taskid_t pr_taskid;      /* task id */
        projid_t pr_projid;      /* project id */
        zoneid_t pr_zoneid;      /* zone id */
        lwpstatus_t pr_lwp;      /* status of the representative lwp */
    } pstatus_t;
```
______***unchanged_portion_omitted_***

```
**********************************************************
   34007 Wed Jun 15 19:31:49 2016
new/usr/src/man/man5/privileges.5
Code review comments from jeffpc
**********************************************************
'\" te
.\" Copyright (c) 2009, Sun Microsystems, Inc. All Rights Reserved.
.\" Copyright 2015, Joyent, Inc. All Rights Reserved.
.\" The contents of this file are subject to the terms of the Common Development
.\"  See the License for the specific language governing permissions and limitat
.\" the fields enclosed by brackets "[]" replaced with your own identifying info
.TH PRIVILEGES 5 "Jun 6, 2016"
.TH PRIVILEGES 5 "Oct 30, 2015"
.SH NAME
privileges \- process privilege model
.SH DESCRIPTION
.LP
Solaris software implements a set of privileges that provide fine-grained
control over the actions of processes. The possession of a certain privilege
allows a process to perform a specific set of restricted operations.
.sp
.LP
The change to a primarily privilege-based security model in the Solaris
operating system gives developers an opportunity to restrict processes to those
privileged operations actually needed instead of all (super-user) or no
privileges (non-zero UIDs). Additionally, a set of previously unrestricted
operations now requires a privilege; these privileges are dubbed the "basic"
privileges and are by default given to all processes.
.sp
.LP
Taken together, all defined privileges with the exception of the "basic"
privileges compose the set of privileges that are traditionally associated with
the root user. The "basic" privileges are "privileges" unprivileged processes
were accustomed to having.
.sp
.LP
The defined privileges are:
.sp
.ne 2
.na
\fB\fBPRIV_CONTRACT_EVENT\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to request reliable delivery of events to an event endpoint.
.sp
Allow a process to include events in the critical event set term of a template
which could be generated in volume by the user.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_CONTRACT_IDENTITY\fR\fR
.ad
.sp .6
.RS 4n
Allows a process to set the service FMRI value of a process contract template.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_CONTRACT_OBSERVER\fR\fR
.ad
.sp .6
```

```
.RS 4n
Allow a process to observe contract events generated by contracts created and
owned by users other than the process's effective user ID.
.sp
Allow a process to open contract event endpoints belonging to contracts created
and owned by users other than the process's effective user ID.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_CPC_CPU\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to access per-CPU hardware performance counters.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_DTRACE_KERNEL\fR\fR
.ad
.sp .6
.RS 4n
Allow DTrace kernel-level tracing.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_DTRACE_PROC\fR\fR
.ad
.sp .6
.RS 4n
Allow DTrace process-level tracing. Allow process-level tracing probes to be
placed and enabled in processes to which the user has permissions.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_DTRACE_USER\fR\fR
.ad
.sp .6
.RS 4n
Allow DTrace user-level tracing. Allow use of the syscall and profile DTrace
providers to examine processes to which the user has permissions.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_FILE_CHOWN\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to change a file's owner user ID. Allow a process to change a
file's group ID to one other than the process's effective group ID or one of
the process's supplemental group IDs.
.RE

.sp
.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 open objects in the filesystem for reading. This
privilege is not necessary to read from an already open file which was opened
before dropping the \fBPRIV_FILE_READ\fR privilege.
.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 open objects in the filesytem for writing, or otherwise
modify them. This privilege is not necessary to write to an already open file
which was opened before dropping the \fBPRIV_FILE_WRITE\fR privilege.
.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. This
privilege is not necessary to communicate using an existing endpoint already
opened before dropping the \fBPRIV_NET_ACCESS\fR privilege.
.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_MEMINFO\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to access physical memory information.
.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
\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
Allow a process to perform various system configuration tasks. Allow
filesystem-specific administrative procedures, such as filesystem configuration
ioctls, quota calls, creation and deletion of snapshots, and manipulating the
PCFS bootsector.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_SYS_DEVICES\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to create device special files. Allow a process to successfully
call a kernel module that calls the kernel \fBdrv_priv\fR(9F) function to check
for allowed access. Allow a process to open the real console device directly.
Allow a process to open devices that have been exclusively opened.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_SYS_DL_CONFIG\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to configure a system’s datalink interfaces.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_SYS_IP_CONFIG\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to configure a system’s IP interfaces and routes. Allow a
process to configure network parameters for \fBTCP/IP\fR using \fBndd\fR. Allow
a process access to otherwise restricted \fBTCP/IP\fR information using
\fBndd\fR. Allow a process to configure \fBIPsec\fR. Allow a process to pop
anchored \fBSTREAM\fRs modules with matching \fBzoneid\fR.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_SYS_IPC_CONFIG\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to increase the size of a System V IPC Message Queue buffer.
.RE
```

```
.sp
.ne 2
.na
\fB\fBPRIV_SYS_IPTUN_CONFIG\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to configure IP tunnel links.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_SYS_LINKDIR\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to unlink and link directories.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_SYS_MOUNT\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to mount and unmount filesystems that would otherwise be
restricted (that is, most filesystems except \fBnamefs\fR). Allow a process to
add and remove swap devices.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_SYS_NET_CONFIG\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to do all that \fBPRIV_SYS_IP_CONFIG\fR,
\fBPRIV_SYS_DL_CONFIG\fR, and \fBPRIV_SYS_PPP_CONFIG\fR allow, plus the
following: use the \fBrpcmod\fR STREAMS module and insert/remove STREAMS
modules on locations other than the top of the module stack.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_SYS_NFS\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to provide NFS service: start NFS kernel threads, perform NFS
locking operations, bind to NFS reserved ports: ports 2049 (\fBnfs\fR) and port
4045 (\fBlockd\fR).
.RE

.sp
.ne 2
.na
\fB\fBPRIV_SYS_PPP_CONFIG\fR\fR
.ad
.sp .6
.RS 4n
Allow a process to create, configure, and destroy PPP instances with pppd(1M)
\fBpppd\fR(1M) and control PPPoE plumbing with \fBsppptun\fR(1M)sppptun(1M).
```

```
This privilege is granted by default to exclusive IP stack instance zones.
.RE

.sp
.ne 2
.na
\fB\fBPRIV_SYS_RES_BIND\fR\fR
.ad
.sp .6
.RS 4n
Allows a process to bind processes to processor sets.
.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,
\fBPRIV_FILE_READ\fR, \fBPRIV_FILE_WRITE\fR, \fBPRIV_NET_ACCESS\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
role. Use \fBppriv\fR(1) to enumerate the privileges supported on a system and
\fBtruss\fR(1) to determine which privileges a program requires.
.SH SEE ALSO
.LP
\fBmdb\fR(1), \fBppriv\fR(1), \fBadd_drv\fR(1M), \fBifconfig\fR(1M),
\fBlockd\fR(1M), \fBnfsd\fR(1M), \fBpppd\fR(1M), \fBrem_drv\fR(1M),
\fBsmbd\fR(1M), \fBsppptun\fR(1M), \fBupdate_drv\fR(1M), \fBIntro\fR(2),
\fBaccess\fR(2), \fBacct\fR(2), \fBacl\fR(2), \fBadjtime\fR(2), \fBaudit\fR(2),
\fBauditon\fR(2), \fBchmod\fR(2), \fBchown\fR(2), \fBchroot\fR(2),
\fBcreat\fR(2), \fBexec\fR(2), \fBfcntl\fR(2), \fBfork\fR(2),
\fBfpathconf\fR(2), \fBgetacct\fR(2), \fBgetpflags\fR(2), \fBgetppriv\fR(2),
\fBgetsid\fR(2), \fBkill\fR(2), \fBlink\fR(2), \fBmemcntl\fR(2),
\fBmknod\fR(2), \fBmount\fR(2), \fBmsgctl\fR(2), \fBnice\fR(2),
\fBntp_adjtime\fR(2), \fBopen\fR(2), \fBp_online\fR(2), \fBpriocntl\fR(2),
\fBpriocntlset\fR(2), \fBprocessor_bind\fR(2), \fBpset_bind\fR(2),
\fBpset_create\fR(2), \fBreadlink\fR(2), \fBresolvepath\fR(2), \fBrmdir\fR(2),
\fBsemctl\fR(2), \fBsetauid\fR(2), \fBsetegid\fR(2), \fBseteuid\fR(2),
\fBsetgid\fR(2), \fBsetgroups\fR(2), \fBsetpflags\fR(2), \fBsetppriv\fR(2),
\fBsetrctl\fR(2), \fBsetregid\fR(2), \fBsetreuid\fR(2), \fBsetrlimit\fR(2),
\fBsettaskid\fR(2), \fBsetuid\fR(2), \fBshmctl\fR(2), \fBshmget\fR(2),
\fBshmop\fR(2), \fBsigsend\fR(2), \fBstat\fR(2), \fBstatvfs\fR(2),
\fBstime\fR(2), \fBswapctl\fR(2), \fBsysinfo\fR(2), \fBuadmin\fR(2),
\fBulimit\fR(2), \fBumount\fR(2), \fBunlink\fR(2), \fButime\fR(2),
\fButimes\fR(2), \fBbind\fR(3SOCKET), \fBdoor_ucred\fR(3C),
\fBpriv_addset\fR(3C), \fBpriv_set\fR(3C), \fBpriv_getbyname\fR(3C),
\fBpriv_getbynum\fR(3C), \fBpriv_set_to_str\fR(3C), \fBpriv_str_to_set\fR(3C),
\fBsocket\fR(3SOCKET), \fBt_bind\fR(3NSL), \fBtimer_create\fR(3C),
\fBucred_get\fR(3C), \fBexec_attr\fR(4), \fBproc\fR(4), \fBsystem\fR(4),
\fBuser_attr\fR(4), \fBxVM\fR(5), \fBddi_cred\fR(9F), \fBdrv_priv\fR(9F),
\fBpriv_getbyname\fR(9F), \fBpriv_policy\fR(9F), \fBpriv_policy_choice\fR(9F),
\fBpriv_policy_only\fR(9F)
.sp
.LP
\fISystem Administration Guide: Security Services\fR
```

```
**********************************************************
    3590 Wed Jun 15 19:31:50 2016
new/usr/src/man/man5/security-flags.5
Code review comments from jeffpc
**********************************************************
.\"
.\" 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 2015, Richard Lowe.
.\"
.TH "SECURITY-FLAGS" "5" "June 6, 2016"
.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 four 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 upper set (abbreviated \fIU\fR)
specify the maximal flags that a process can have in its inheritable set.  The
lower set (abbreviated \fIL\fR) specify the minimal amount of flags that a
process must have in its inheritable set.  The inheritable set may be changed
at any time, subject to permissions and the lower and upper sets.
.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 (\fBASLR\fR)
.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

.sp
.ne 2
.na
```

```
Forbid mappings at NULL (\fBFORBIDNULLMAP\fR)
.ad
.RS 11n
Mappings with an address of 0 are forbidden, and return EINVAL rather than
being honored.
.RE

.sp
.ne 2
.na
Make the userspace stack non-executable (\fBNOEXECSTACK\fR)
.ad
.RS 11n
The stack will be mapped without executable permission, and attempts to
execute it will fault.
.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 \fBdefault\fR, \fBlower\fR and \fBupper\fR, property groups.
For example, to enable ASLR by default you would execute the following
commands:
.sp
.in +2
.nf
# svccfg -s svc:/system/process-security setprop default/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)
```

```
**********************************************************
   14783 Wed Jun 15 19:31:51 2016
new/usr/src/man/man5/smf_method.5
Code review comments from jeffpc
**********************************************************
'\" te
.\" Copyright (c) 2009, Sun Microsystems, Inc. All Rights Reserved.
.\" The contents of this file are subject to the terms of the Common Development
.\"  See the License for the specific language governing permissions and limitat
.\" the fields enclosed by brackets "[]" replaced with your own identifying info
.TH SMF_METHOD 5 "June 6, 2016"
.TH SMF_METHOD 5 "May 20, 2009"
.SH NAME
smf_method \- service management framework conventions for methods
.SH DESCRIPTION
.LP
The class of services managed by \fBsvc.startd\fR(1M) in the service management
framework, \fBsmf\fR(5), consists of applications that fit a simple
\fBfork\fR(2)-\fBexec\fR(2) model. The \fBsvc.startd\fR(1M) master daemon and
other restarters support the \fBfork\fR(2)-\fBexec\fR(2) model, potentially
with additional capabilities. The \fBsvc.startd\fR(1M) daemon and other
restarters require that the methods which activate, manipulate, or examine a
service instance follow the conventions described in this manual page.
.SS "Invocation form"
.LP
The form of a method invocation is not dictated by convention. In some cases, a
method invocation might consist of the direct invocation of the daemon or other
binary executable that provides the service. For cases in which an executable
script or other mediating executable is used, the convention recommends the
form:
.sp
.in +2
.nf
/path/to/method_executable abbr_method_name
.fi
.in -2

.sp
.LP
The \fIabbr_method_name\fR used for the recommended form is a supported method
such as \fBstart\fR or \fBstop\fR. The set of methods supported by a restarter
is given on the related restarter page. The \fBsvc.startd\fR(1M) daemon
supports \fBstart\fR, \fBstop\fR, and \fBrefresh\fR methods.
.sp
.LP
A restarter might define other kinds of methods beyond those referenced in this
page. The conventions surrounding such extensions are defined by the restarter
and might not be identical to those given here.
.SS "Environment Variables"
.LP
The restarter provides four environment variables to the method that determine
the context in which the method is invoked.
.sp
.ne 2
.na
\fB\fBSMF_FMRI\fR\fR
.ad
.sp .6
.RS 4n
The service fault management resource identifier (FMRI) of the instance for
which the method is invoked.
.RE

.sp
.ne 2
.na
```

```
\fB\fBSMF_METHOD\fR\fR
.ad
.sp .6
.RS 4n
The full name of the method being invoked, such as \fBstart\fR or \fBstop\fR.
.RE

.sp
.ne 2
.na
\fB\fBSMF_RESTARTER\fR\fR
.ad
.sp .6
.RS 4n
The service FMRI of the restarter that invokes the method
.RE

.sp
.ne 2
.na
\fB\fBSMF_ZONENAME\fR\fR
.ad
.sp .6
.RS 4n
The name of the zone in which the method is running. This can also be obtained
by using the \fBzonename\fR(1) command.
.RE

.sp
.LP
These variables should be removed from the environment prior to the invocation
of any persistent process by the method. A convenience shell function,
\fBsmf_clear_env\fR, is given for service authors who use Bourne-compatible
shell scripting to compose service methods in the include file described below.
.sp
.LP
The method context can cause other environment variables to be set as described
below.
.SS "Method Definition"
.LP
A method is defined minimally by three properties in a propertygroup of type
\fBmethod\fR.
.sp
.LP
These properties are:
.sp
.ne 2
.na
\fBexec (\fIastring\fR)\fR
.ad
.RS 27n
Method executable string.
.RE

.sp
.ne 2
.na
\fBtimeout_seconds (\fIcount\fR)\fR
.ad
.RS 27n
Number of seconds before method times out. See the \fBTimeouts\fR section for
more detail.
.RE

.sp
.ne 2
```

```
.na
\fBtype (\fIastring\fR)\fR
.ad
.RS 27n
Method type. Currently always set to \fBmethod\fR.
.RE

.sp
.LP
A Method Context can be defined to further refine the execution environment of
the method. See the \fBMethod Context\fR section for more information.
.SS "Method Tokens"
.LP
When defined in the \fBexec\fR string of the method by the restarter
\fBsvc.startd\fR, a set of tokens are parsed and expanded with appropriate
value. Other restarters might not support method tokens. The delegated
restarter for inet services, \fBinetd\fR(1M), does not support the following
method expansions.
.sp
.ne 2
.na
\fB\fB%%\fR\fR
.ad
.sp .6
.RS 4n
%
.RE

.sp
.ne 2
.na
\fB\fB%r\fR\fR
.ad
.sp .6
.RS 4n
Name of the restarter, such as \fBsvc.startd\fR
.RE

.sp
.ne 2
.na
\fB\fB%m\fR\fR
.ad
.sp .6
.RS 4n
The full name of the method being invoked, such as \fBstart\fR or \fBstop\fR.
.RE

.sp
.ne 2
.na
\fB\fB%s\fR\fR
.ad
.sp .6
.RS 4n
Name of the service
.RE

.sp
.ne 2
.na
\fB\fB%i\fR\fR
.ad
.sp .6
.RS 4n
Name of the instance
```

```
.RE

.sp
.ne 2
.na
\fB\fB\fR\fB%f\fR\fR
.ad
.sp .6
.RS 4n
FMRI of the instance
.RE

.sp
.ne 2
.na
\fB\fB%{prop[:,]}\fR\fR
.ad
.sp .6
.RS 4n
Value(s) of a property. The \fBprop\fR might be a property FMRI, a property
group name and a property name separated by a \fB/\fR, or a property name in
the \fBapplication\fR property group. These values can be followed by a \fB,\fR
(comma) or \fB:\fR (colon). If present, the separators are used to separate
multiple values. If absent, a space is used. The following shell metacharacters
encountered in string values are quoted with a \ (backslash):
.sp
.in +2
.nf
; & ( ) | ^ < > newline space tab  \  " ’
.fi
.in -2

An invalid expansion constitutes method failure.
.RE

.sp
.LP
Two explicit tokens can be used in the place of method commands.
.sp
.ne 2
.na
\fB\fB:kill [-signal]\fR\fR
.ad
.sp .6
.RS 4n
Sends the specified signal, which is \fBSIGTERM\fR by default, to all processes
in the primary instance contract. Always returns \fBSMF_EXIT_OK\fR. This token
should be used to replace common \fBpkill\fR invocations.
.RE

.sp
.ne 2
.na
\fB\fB:true\fR\fR
.ad
.sp .6
.RS 4n
Always returns \fBSMF_EXIT_OK\fR. This token should be used for methods that
are required by the restarter but which are unnecessary for the particular
service implementation.
.RE

.SS "Exiting and Exit Status"
.LP
The required behavior of a start method is to delay exiting until the service
instance is ready to answer requests or is otherwise functional.
```

```
.sp
.LP
The following exit status codes are defined in \fB<libscf.h>\fR and in the
shell support file.
.sp

.sp
.TS
l l l
l l l .
\fBSMF_EXIT_OK\fR       \fB0\fR T{
Method exited, performing its operation successfully.
T}
\fBSMF_EXIT_ERR_FATAL\fR        \fB95\fR        T{
Method failed fatally and is unrecoverable without administrative intervention.
T}
\fBSMF_EXIT_ERR_CONFIG\fR       \fB96\fR        T{
Unrecoverable configuration error. A common condition that returns this exit sta
T}
\fBSMF_EXIT_ERR_NOSMF\fR        \fB99\fR        T{
Method has been mistakenly invoked outside the \fBsmf\fR(5) facility. Services t
T}
\fBSMF_EXIT_ERR_PERM\fR \fB100\fR       T{
Method requires a form of permission such as file access, privilege, authorizati
T}
\fBSMF_EXIT_ERR_OTHER\fR        \fBnon-zero\fR  T{
Any non-zero exit status from a method is treated as an unknown error. A series
T}
.TE

.sp
.LP
Use of a precise exit code allows the responsible restarter to categorize an
error response as likely to be intermittent and worth pursuing restart or
permanent and request administrative intervention.
.SS "Timeouts"
.LP
Each method can have an independent timeout, given in seconds. The choice of a
particular timeout should be based on site expectations for detecting a method
failure due to non-responsiveness. Sites with replicated filesystems or other
failover resources can elect to lengthen method timeouts from the default.
Sites with no remote resources can elect to shorten the timeouts. Method
timeout is specified by the \fBtimeout_seconds\fR property.
.sp
.LP
If you specify \fB0 timeout_seconds\fR for a method, it declares to the
restarter that there is no timeout for the service. This setting is not
preferred, but is available for services that absolutely require it.
.sp
.LP
\fB-1 timeout_seconds\fR is also accepted, but is a deprecated specification.
.SS "Shell Programming Support"
.LP
A set of environment variables that define the above exit status values is
provided with convenience shell functions in the file
\fB/lib/svc/share/smf_include.sh\fR. This file is a Bourne shell script
suitable for inclusion via the source operator in any Bourne-compatible shell.
.sp
.LP
To assist in the composition of scripts that can serve as SMF methods as well
as \fB/etc/init.d\fR scripts, the \fBsmf_present()\fR shell function is
provided. If the \fBsmf\fR(5) facility is not available, \fBsmf_present()\fR
returns a non-zero exit status.
.sp
.LP
One possible structure for such a script follows:
```

```
.sp
.in +2
.nf
if smf_present; then
      # Shell code to run application as managed service
      ....

      smf_clear_env
else
      # Shell code to run application as /etc/init.d script
      ....
fi
.fi
.in -2

.sp
.LP
This example shows the use of both convenience functions that are provided.
.SS "Method Context"
.LP
The service management facility offers a common mechanism set the context in
which the \fBfork\fR(2)-\fBexec\fR(2) model services execute.
.sp
.LP
The desired method context should be provided by the service developer. All
service instances should run with the lowest level of privileges possible to
limit potential security compromises.
.sp
.LP
A method context can contain the following properties:
.sp
.ne 2
.na
\fB\fBuse_profile\fR\fR
.ad
.sp .6
.RS 4n
A boolean that specifies whether the profile should be used instead of the
\fBuser\fR, \fBgroup\fR, \fBprivileges\fR, and \fBlimit_privileges\fR
properties.
.RE

.sp
.ne 2
.na
\fBenvironment\fR
.ad
.sp .6
.RS 4n
Environment variables to insert into the environment of the method, in the form
of a number of \fBNAME=value\fR strings.
.RE

.sp
.ne 2
.na
\fB\fBprofile\fR\fR
.ad
.sp .6
.RS 4n
The name of an RBAC (role-based access control) profile which, along with the
method executable, identifies an entry in \fBexec_attr\fR(4).
.RE

.sp
.ne 2
```

```
.na
\fB\fBuser\fR\fR
.ad
.sp .6
.RS 4n
The user ID in numeric or text form.
.RE

.sp
.ne 2
.na
\fB\fBgroup\fR\fR
.ad
.sp .6
.RS 4n
The group ID in numeric or text form.
.RE

.sp
.ne 2
.na
\fB\fBsupp_groups\fR\fR
.ad
.sp .6
.RS 4n
An optional string that specifies the supplemental group memberships by ID, in
numeric or text form.
.RE

.sp
.ne 2
.na
\fB\fBprivileges\fR\fR
.ad
.sp .6
.RS 4n
An optional string specifying the privilege set as defined in
\fBprivileges\fR(5).
.RE

.sp
.ne 2
.na
\fB\fBlimit_privileges\fR\fR
.ad
.sp .6
.RS 4n
An optional string specifying the limit privilege set as defined in
\fBprivileges\fR(5).
.RE

.sp
.ne 2
.na
\fB\fBworking_directory\fR\fR
.ad
.sp .6
.RS 4n
The home directory from which to launch the method. \fB:home\fR can be used as
a token to indicate the home directory of the user whose \fBuid\fR is used to
launch the method. If the property is unset, \fB:home\fR is used.
.RE

.sp
.ne 2
.na
```

```
\fB\fBsecurity_flags\fR\fR
.ad
.sp .6
.RS 4n
The security flags to apply when launching the method.  See \fBsecurity-flags\fR
.sp
.LP
The "default" keyword specifies those flags specified in
\fBsvc:/system/process-security\fR.  The "all" keyword enables all flags, the
"none" keyword enables no flags.  Further flags may be added by specifying
their name, or removed by specifying their name prefixed by '-' or '!'.
.sp
.LP
Use of "all" has associated risks, as future versions of the system may
include further flags which may harm poorly implemented software.
.RE

.sp
.ne 2
.na
\fB\fBcorefile_pattern\fR\fR
.ad
.sp .6
.RS 4n
An optional string that specifies the corefile pattern to use for the service,
as per \fBcoreadm\fR(1M). Most restarters supply a default. Setting this
property overrides local customizations to the global core pattern.
.RE

.sp
.ne 2
.na
\fB\fBproject\fR\fR
.ad
.sp .6
.RS 4n
The project ID in numeric or text form. \fB:default\fR can be used as a token
to indicate a project identified by \fBgetdefaultproj\fR(3PROJECT) for the user
whose \fBuid\fR is used to launch the method.
.RE

.sp
.ne 2
.na
\fB\fBresource_pool\fR\fR
.ad
.sp .6
.RS 4n
The resource pool name on which to launch the method. \fB:default\fR can be
used as a token to indicate the pool specified in the \fBproject\fR(4) entry
given in the \fBproject\fR attribute above.
.RE

.sp
.LP
The method context can be set for the entire service instance by specifying a
\fBmethod_context\fR property group for the service or instance. A method might
override the instance method context by providing the method context properties
on the method property group.
.sp
.LP
Invalid method context settings always lead to failure of the method, with the
exception of invalid environment variables that issue warnings.
.sp
.LP
In addition to the context defined above, many \fBfork\fR(2)-\fBexec\fR(2)
```

```
model restarters also use the following conventions when invoking executables
as methods:
.sp
.ne 2
.na
\fBArgument array\fR
.ad
.sp .6
.RS 4n
The arguments in \fBargv[]\fR are set consistently with the result \fB/bin/sh
-c\fR of the \fBexec\fR string.
.RE

.sp
.ne 2
.na
\fBFile descriptors\fR
.ad
.sp .6
.RS 4n
File descriptor \fB0\fR is \fB/dev/null\fR. File descriptors \fB1\fR and
\fB2\fR are recommended to be a per-service log file.
.RE

.SH FILES
.ne 2
.na
\fB\fB/lib/svc/share/smf_include.sh\fR\fR
.ad
.sp .6
.RS 4n
Definitions of exit status values.
.RE

.sp
.ne 2
.na
\fB\fB/usr/include/libscf.h\fR\fR
.ad
.sp .6
.RS 4n
Definitions of exit status codes.
.RE

.SH SEE ALSO
.LP
\fBzonename\fR(1), \fBcoreadm\fR(1M), \fBinetd\fR(1M), \fBsvccfg\fR(1M),
\fBsvc.startd\fR(1M), \fBexec\fR(2), \fBfork\fR(2),
\fBgetdefaultproj\fR(3PROJECT), \fBexec_attr\fR(4), \fBproject\fR(4),
\fBservice_bundle\fR(4), \fBattributes\fR(5), \fBprivileges\fR(5),
\fBrbac\fR(5), \fBsmf\fR(5), \fBsmf_bootstrap\fR(5), \fBzones\fR(5),
\fBsecurity-flags\fR(5)
.SH NOTES
.LP
The present version of \fBsmf\fR(5) does not support multiple repositories.
.sp
.LP
When a service is configured to be started as root but with privileges
different from \fBlimit_privileges\fR, the resulting process is privilege
aware.  This can be surprising to developers who expect \fBseteuid(<non-zero
UID>)\fR to reduce privileges to basic or less.
```

****
1454 Wed Jun 15 19:31:53 2016
new/usr/src/test/os-tests/tests/secflags/Makefile
Code review comments from jeffpc
****

```
#
# 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 2015, Richard Lowe.


include $(SRC)/cmd/Makefile.cmd
include $(SRC)/test/Makefile.com

PROG =  secflags_aslr           \
        secflags_core           \
        secflags_dts            \
        secflags_elfdump        \
        secflags_forbidnullmap  \
        secflags_limits         \
        secflags_noexecstack    \
        secflags_proc           \
        secflags_psecflags      \
        secflags_syscall        \
        secflags_truss          \
        secflags_zonecfg

PROG += addrs-32 addrs-64 stacky

ROOTOPTPKG = $(ROOT)/opt/os-tests
TESTDIR = $(ROOTOPTPKG)/tests/secflags

CMDS = $(PROG:%=$(TESTDIR)/%)
$(CMDS) := FILEMODE = 0555

addrs-32: addrs.c
        $(LINK.c) addrs.c -o $@ $(LDLIBS)
        $(POST_PROCESS)

addrs-64: addrs.c
        $(LINK64.c) addrs.c -o $@ $(LDLIBS)
        $(POST_PROCESS)

stacky := MAPFILE.NES=                  # Will foil the test, clearly
stacky: stacky.o
        $(LINK.c) stacky.o -o $@ $(LDLIBS)
        $(LINK.c) -m32 stacky.o -o $@ $(LDLIBS)
        $(POST_PROCESS)

secflags_syscall: secflags_syscall.c
        $(LINK.c) secflags_syscall.c -o $@ $(LDLIBS)
        $(LINK.c) -m32 secflags_syscall.c -o $@ $(LDLIBS)
        $(POST_PROCESS)

all: $(PROG)

install: all $(CMDS)
```

```
lint:

clobber: clean
        -$(RM) $(PROG)

clean:

$(CMDS): $(TESTDIR) $(PROG)

$(TESTDIR):
        $(INS.dir)

$(TESTDIR)/%: %
        $(INS.file)
```

```
**********************************************************
    1381 Wed Jun 15 19:31:54 2016
new/usr/src/test/os-tests/tests/secflags/secflags_aslr.sh
Code review comments from jeffpc
**********************************************************
   1 #! /usr/bin/ksh
   2 #
   3 #
   4 # This file and its contents are supplied under the terms of the
   5 # Common Development and Distribution License ("CDDL"), version 1.0.
   6 # You may only use this file in accordance with the terms of version
   7 # 1.0 of the CDDL.
   8 #
   9 # A full copy of the text of the CDDL should have accompanied this
  10 # source.  A copy of the CDDL is also available via the Internet at
  11 # http://www.illumos.org/license/CDDL.
  12 #

  14 # Copyright 2015, Richard Lowe.
  14 # Copyright 2015, Richald Lowe.

  16 # Verify that aslr messes things up, by comparing the mappings of 2 identical
  17 # processes
  16 # Verify that aslr musses things up, by comparing the mappings of 2 identical pr

  19 LC_ALL=C                        # Collation is important

  21 /usr/bin/psecflags -s aslr $$

  23 tmpdir=/tmp/test.$$

  25 mkdir $tmpdir
  26 cd $tmpdir

  28 cleanup() {
  29     cd /
  30     rm -fr $tmpdir
  31 }
______unchanged_portion_omitted_
```

****
1754 Wed Jun 15 19:31:54 2016
new/usr/src/test/os-tests/tests/secflags/secflags_elfdump.sh
Code review comments from jeffpc
****
______*unchanged_portion_omitted_*

```
trap cleanup EXIT

## gcore-produced core
gcore $pid >/dev/null

cat > gcore-expected.$$ <<EOF
    namesz: 0x5
    descsz: 0x28
    descsz: 0x14
    type:   [ NT_SECFLAGS ]
    name:
        CORE\0
    desc: (prsecflags_t)
        pr_version:    1
        pr_effective:  [ ASLR ]
        pr_inherit:    [ ASLR ]
        pr_lower:      0
        pr_upper:      [ ASLR FORBIDNULLMAP NOEXECSTACK ]
EOF

/usr/bin/elfdump -n core.${pid} | grep -B5 -A5 prsecflags_t > gcore-output.$$

if ! diff -u gcore-expected.$$ gcore-output.$$; then
    exit 1;
fi

## kernel-produced core
kill -SEGV $pid

cat > core-expected.$$ <<EOF
    namesz: 0x5
    descsz: 0x28
    descsz: 0x14
    type:   [ NT_SECFLAGS ]
    name:
        CORE\0
    desc: (prsecflags_t)
        pr_version:    1
        pr_effective:  [ ASLR ]
        pr_inherit:    [ ASLR ]
        pr_lower:      0
        pr_upper:      [ ASLR FORBIDNULLMAP NOEXECSTACK ]
EOF

/usr/bin/elfdump -n core | grep -B5 -A5 prsecflags_t > core-output.$$

if ! diff -u core-expected.$$ core-output.$$; then
    exit 1;
fi

exit 0
```

****
    3711 Wed Jun 15 19:31:55 2016
new/usr/src/test/os-tests/tests/secflags/secflags_psecflags.sh
Code review comments from jeffpc
****

______unchanged_portion_omitted_

```
self_add() {
    echo "Add (self)"
    /usr/bin/psecflags -s current,noexecstack $$
    cat > expected <<EOF
        I:      aslr,noexecstack
        I:      aslr, noexecstack
EOF

    /usr/bin/psecflags $$ | grep I: > output
    diff -u expected output || exit 1
}
```

______unchanged_portion_omitted_

```
child_add() {
    echo "Add (child)"

    typeset pid;

    /usr/bin/psecflags -s aslr $$
    /usr/bin/psecflags -s current,noexecstack -e sleep 10000 &
    pid=$!
    cat > expected <<EOF
        E:      aslr,noexecstack
        I:      aslr,noexecstack
        E:      aslr, noexecstack
        I:      aslr, noexecstack
EOF
    /usr/bin/psecflags $pid | grep '[IE]:' > output
    kill $pid
    /usr/bin/psecflags -s none $$
    diff -u expected output || exit 1
}
```

______unchanged_portion_omitted_

****
3717 Wed Jun 15 19:31:56 2016
new/usr/src/test/os-tests/tests/secflags/secflags_zonecfg.sh
Code review comments from jeffpc
****

```
#! /usr/bin/ksh
#
#
# 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 2015, Richard Lowe.
# Copyright 2015, Richald Lowe.

# Verify that zones can be configured with security-flags
LC_ALL=C                        # Collation is important

expect_success() {
    name=$1

    (echo "create -b";
     echo "set zonepath=/$name.$$";
     cat /dev/stdin;
     echo "verify";
     echo "commit";
     echo "exit") | zonecfg -z $name.$$ > out.$$ 2>&1

    r=$?

    zonecfg -z $name.$$ delete -F

    if (($r != 0)); then
        printf "%s: FAIL\n" $name
        cat out.$$
        rm out.$$
        return 1
    else
        rm out.$$
        printf  "%s: PASS\n" $name
        return 0
    fi
}
```
______*unchanged_portion_omitted_*

```
**********************************************************
    3364 Wed Jun 15 19:31:57 2016
new/usr/src/uts/common/exec/elf/elf_impl.h
Code review comments from jeffpc
**********************************************************
/*
 * 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.
 */

#ifndef _ELF_ELF_IMPL_H
#define _ELF_ELF_IMPL_H

#pragma ident   "%Z%%M% %I%     %E% SMI"

#endif /* ! codereview */
#ifdef  __cplusplus
extern "C" {
#endif

#if     !defined(_LP64) || defined(_ELF32_COMPAT)

/*
 * Definitions for ELF32, native 32-bit or 32-bit compatibility mode.
 */
#define ELFCLASS        ELFCLASS32
typedef unsigned int    aux_val_t;
typedef auxv32_t        aux_entry_t;

#define USR_LIB_RTLD    "/usr/lib/ld.so.1"

#else   /* !_LP64 || _ELF32_COMPAT */

/*
 * Definitions for native 64-bit ELF
 */
#define ELFCLASS        ELFCLASS64
typedef unsigned long   aux_val_t;
typedef auxv_t          aux_entry_t;

/* put defines for 64-bit architectures here */
#if defined(__sparcv9)
#define USR_LIB_RTLD    "/usr/lib/sparcv9/ld.so.1"
#endif

#if defined(__amd64)
```

```
#define USR_LIB_RTLD    "/usr/lib/amd64/ld.so.1"
#endif

#endif  /* !_LP64 || _ELF32_COMPAT */

/*
 * Start of an ELF Note.
 */
typedef struct {
        Nhdr    nhdr;
        char    name[8];
} Note;

#ifdef  _ELF32_COMPAT
/*
 * These are defined only for the 32-bit compatibility
 * compilation mode of the 64-bit kernel.
 */
#define elfexec elf32exec
#define elfnote elf32note
#define elfcore elf32core
#define mapexec_brand           mapexec32_brand
#define setup_note_header       setup_note_header32
#define write_elfnotes          write_elfnotes32
#define setup_old_note_header   setup_old_note_header32
#define write_old_elfnotes      write_old_elfnotes32

#if defined(__sparc)
#define gwindows_t      gwindows32_t
#define rwindow         rwindow32
#endif

#define psinfo_t        psinfo32_t
#define pstatus_t       pstatus32_t
#define lwpsinfo_t      lwpsinfo32_t
#define lwpstatus_t     lwpstatus32_t

#define prgetpsinfo     prgetpsinfo32
#define prgetstatus     prgetstatus32
#define prgetlwpsinfo   prgetlwpsinfo32
#define prgetlwpstatus  prgetlwpstatus32
#define prgetwindows    prgetwindows32

#define prpsinfo_t      prpsinfo32_t
#define prstatus_t      prstatus32_t
#if defined(prfpregset_t)
#undef prfpregset_t
#endif
#define prfpregset_t    prfpregset32_t

#define oprgetstatus    oprgetstatus32
#define oprgetpsinfo    oprgetpsinfo32
#define prgetprfpregs   prgetprfpregs32

#endif  /*      _ELF32_COMPAT   */

extern int elfnote(vnode_t *, offset_t *, int, int, void *, rlim64_t, cred_t *);
extern void setup_old_note_header(Phdr *, proc_t *);
extern void setup_note_header(Phdr *, proc_t *);

extern int write_old_elfnotes(proc_t *, int, vnode_t *, offset_t,
    rlim64_t, cred_t *);

extern int write_elfnotes(proc_t *, int, vnode_t *, offset_t,
    rlim64_t, cred_t *, core_content_t);
```

```
#ifdef  __cplusplus
}
#endif

#endif  /* _ELF_ELF_IMPL_H */
```

```
**********************************************************
   37198 Wed Jun 15 19:31:58 2016
new/usr/src/uts/common/os/fork.c
Code review comments from jeffpc
**********************************************************
______unchanged_portion_omitted_

/*
 * create a child proc struct.
 */
static int
getproc(proc_t **cpp, pid_t pid, uint_t flags)
{
        proc_t          *pp, *cp;
        pid_t           newpid;
        struct user     *uarea;
        extern uint_t   nproc;
        struct cred     *cr;
        uid_t           ruid;
        zoneid_t        zoneid;
        task_t          *task;
        kproject_t      *proj;
        zone_t          *zone;
        int             rctlfail = 0;

        if (zone_status_get(curproc->p_zone) >= ZONE_IS_SHUTTING_DOWN)
                return (-1);    /* no point in starting new processes */

        pp = (flags & GETPROC_KERNEL) ? &p0 : curproc;
        task = pp->p_task;
        proj = task->tk_proj;
        zone = pp->p_zone;

        mutex_enter(&pp->p_lock);
        mutex_enter(&zone->zone_nlwps_lock);
        if (proj != proj0p) {
                if (task->tk_nprocs >= task->tk_nprocs_ctl)
                        if (rctl_test(rc_task_nprocs, task->tk_rctls,
                            pp, 1, 0) & RCT_DENY)
                                rctlfail = 1;

                if (proj->kpj_nprocs >= proj->kpj_nprocs_ctl)
                        if (rctl_test(rc_project_nprocs, proj->kpj_rctls,
                            pp, 1, 0) & RCT_DENY)
                                rctlfail = 1;

                if (zone->zone_nprocs >= zone->zone_nprocs_ctl)
                        if (rctl_test(rc_zone_nprocs, zone->zone_rctls,
                            pp, 1, 0) & RCT_DENY)
                                rctlfail = 1;

                if (rctlfail) {
                        mutex_exit(&zone->zone_nlwps_lock);
                        mutex_exit(&pp->p_lock);
                        atomic_inc_32(&zone->zone_ffcap);
                        goto punish;
                }
        }
        task->tk_nprocs++;
        proj->kpj_nprocs++;
        zone->zone_nprocs++;
        mutex_exit(&zone->zone_nlwps_lock);
        mutex_exit(&pp->p_lock);

        cp = kmem_cache_alloc(process_cache, KM_SLEEP);
        bzero(cp, sizeof (proc_t));
```

```
        /*
         * Make proc entry for child process
         */
        mutex_init(&cp->p_splock, NULL, MUTEX_DEFAULT, NULL);
        mutex_init(&cp->p_crlock, NULL, MUTEX_DEFAULT, NULL);
        mutex_init(&cp->p_pflock, NULL, MUTEX_DEFAULT, NULL);
#if defined(__x86)
        mutex_init(&cp->p_ldtlock, NULL, MUTEX_DEFAULT, NULL);
#endif
        mutex_init(&cp->p_maplock, NULL, MUTEX_DEFAULT, NULL);
        cp->p_stat = SIDL;
        cp->p_mstart = gethrtime();
        cp->p_as = &kas;
        /*
         * p_zone must be set before we call pid_allocate since the process
         * will be visible after that and code such as prfind_zone will
         * look at the p_zone field.
         */
        cp->p_zone = pp->p_zone;
        cp->p_t1_lgrpid = LGRP_NONE;
        cp->p_tr_lgrpid = LGRP_NONE;

        if ((newpid = pid_allocate(cp, pid, PID_ALLOC_PROC)) == -1) {
                if (nproc == v.v_proc) {
                        CPU_STATS_ADDQ(CPU, sys, procovf, 1);
                        cmn_err(CE_WARN, "out of processes");
                }
                goto bad;
        }

        mutex_enter(&pp->p_lock);
        cp->p_exec = pp->p_exec;
        cp->p_execdir = pp->p_execdir;
        mutex_exit(&pp->p_lock);

        if (cp->p_exec) {
                VN_HOLD(cp->p_exec);
                /*
                 * Each VOP_OPEN() must be paired with a corresponding
                 * VOP_CLOSE(). In this case, the executable will be
                 * closed for the child in either proc_exit() or gexec().
                 */
                if (VOP_OPEN(&cp->p_exec, FREAD, CRED(), NULL) != 0) {
                        VN_RELE(cp->p_exec);
                        cp->p_exec = NULLVP;
                        cp->p_execdir = NULLVP;
                        goto bad;
                }
        }
        if (cp->p_execdir)
                VN_HOLD(cp->p_execdir);

        /*
         * If not privileged make sure that this user hasn’t exceeded
         * v.v_maxup processes, and that users collectively haven’t
         * exceeded v.v_maxuptl processes.
         */
        mutex_enter(&pidlock);
        ASSERT(nproc < v.v_proc);       /* otherwise how’d we get our pid? */
        cr = CRED();
        ruid = crgetruid(cr);
        zoneid = crgetzoneid(cr);
        if (nproc >= v.v_maxup &&       /* short-circuit; usually false */
            (nproc >= v.v_maxuptl ||
            upcount_get(ruid, zoneid) >= v.v_maxup) &&
	    secpolicy_newproc(cr) != 0) {
		mutex_exit(&pidlock);
		zcmn_err(zoneid, CE_NOTE,
		    "out of per-user processes for uid %d", ruid);
		goto bad;
	}

	/*
	 * Everything is cool, put the new proc on the active process list.
	 * It is already on the pid list and in /proc.
	 * Increment the per uid process count (upcount).
	 */
	nproc++;
	upcount_inc(ruid, zoneid);

	cp->p_next = practive;
	practive->p_prev = cp;
	practive = cp;

	cp->p_ignore = pp->p_ignore;
	cp->p_siginfo = pp->p_siginfo;
	cp->p_flag = pp->p_flag & (SJCTL|SNOWAIT|SNOCD);
	cp->p_sessp = pp->p_sessp;
	sess_hold(pp);
	cp->p_brand = pp->p_brand;
	if (PROC_IS_BRANDED(pp))
		BROP(pp)->b_copy_procdata(cp, pp);
	cp->p_bssbase = pp->p_bssbase;
	cp->p_brkbase = pp->p_brkbase;
	cp->p_brksize = pp->p_brksize;
	cp->p_brkpageszc = pp->p_brkpageszc;
	cp->p_stksize = pp->p_stksize;
	cp->p_stkpageszc = pp->p_stkpageszc;
	cp->p_stkprot = pp->p_stkprot;
	cp->p_datprot = pp->p_datprot;
	cp->p_usrstack = pp->p_usrstack;
	cp->p_model = pp->p_model;
	cp->p_ppid = pp->p_pid;
	cp->p_ancpid = pp->p_pid;
	cp->p_portcnt = pp->p_portcnt;
	/*
	 * Security flags are preserved on fork, the inherited copy come into
	 * effect on exec
	 */
	cp->p_secflags = pp->p_secflags;
	bcopy(&pp->p_secflags, &cp->p_secflags, sizeof (psecflags_t));

	/*
	 * Initialize watchpoint structures
	 */
	avl_create(&cp->p_warea, wa_compare, sizeof (struct watched_area),
	    offsetof(struct watched_area, wa_link));

	/*
	 * Initialize immediate resource control values.
	 */
	cp->p_stk_ctl = pp->p_stk_ctl;
	cp->p_fsz_ctl = pp->p_fsz_ctl;
	cp->p_vmem_ctl = pp->p_vmem_ctl;
	cp->p_fno_ctl = pp->p_fno_ctl;

	/*
	 * Link up to parent-child-sibling chain.  No need to lock
	 * in general since only a call to freeproc() (done by the
	 * same parent as newproc()) diddles with the child chain.
	 */
```

```
	cp->p_sibling = pp->p_child;
	if (pp->p_child)
		pp->p_child->p_psibling = cp;

	cp->p_parent = pp;
	pp->p_child = cp;

	cp->p_child_ns = NULL;
	cp->p_sibling_ns = NULL;

	cp->p_nextorph = pp->p_orphan;
	cp->p_nextofkin = pp;
	pp->p_orphan = cp;

	/*
	 * Inherit profiling state; do not inherit REALPROF profiling state.
	 */
	cp->p_prof = pp->p_prof;
	cp->p_rprof_cyclic = CYCLIC_NONE;

	/*
	 * Inherit pool pointer from the parent.  Kernel processes are
	 * always bound to the default pool.
	 */
	mutex_enter(&pp->p_lock);
	if (flags & GETPROC_KERNEL) {
		cp->p_pool = pool_default;
		cp->p_flag |= SSYS;
	} else {
		cp->p_pool = pp->p_pool;
	}
	atomic_inc_32(&cp->p_pool->pool_ref);
	mutex_exit(&pp->p_lock);

	/*
	 * Add the child process to the current task.  Kernel processes
	 * are always attached to task0.
	 */
	mutex_enter(&cp->p_lock);
	if (flags & GETPROC_KERNEL)
		task_attach(task0p, cp);
	else
		task_attach(pp->p_task, cp);
	mutex_exit(&cp->p_lock);
	mutex_exit(&pidlock);

	avl_create(&cp->p_ct_held, contract_compar, sizeof (contract_t),
	    offsetof(contract_t, ct_ctlist));

	/*
	 * Duplicate any audit information kept in the process table
	 */
	if (audit_active)	/* copy audit data to cp */
		audit_newproc(cp);

	crhold(cp->p_cred = cr);

	/*
	 * Bump up the counts on the file structures pointed at by the
	 * parent's file table since the child will point at them too.
	 */
	fcnt_add(P_FINFO(pp), 1);

	if (PTOU(pp)->u_cdir) {
		VN_HOLD(PTOU(pp)->u_cdir);
	} else {
```

```
                ASSERT(pp == &p0);
                /*
                 * We must be at or before vfs_mountroot(); it will take care of
                 * assigning our current directory.
                 */
        }
        if (PTOU(pp)->u_rdir)
                VN_HOLD(PTOU(pp)->u_rdir);
        if (PTOU(pp)->u_cwd)
                refstr_hold(PTOU(pp)->u_cwd);

        /*
         * copy the parent's uarea.
         */
        uarea = PTOU(cp);
        bcopy(PTOU(pp), uarea, sizeof (*uarea));
        flist_fork(P_FINFO(pp), P_FINFO(cp));

        gethrestime(&uarea->u_start);
        uarea->u_ticks = ddi_get_lbolt();
        uarea->u_mem = rm_asrss(pp->p_as);
        uarea->u_acflag = AFORK;

        /*
         * If inherit-on-fork, copy /proc tracing flags to child.
         */
        if ((pp->p_proc_flag & P_PR_FORK) != 0) {
                cp->p_proc_flag |= pp->p_proc_flag & (P_PR_TRACE|P_PR_FORK);
                cp->p_sigmask = pp->p_sigmask;
                cp->p_fltmask = pp->p_fltmask;
        } else {
                sigemptyset(&cp->p_sigmask);
                premptyset(&cp->p_fltmask);
                uarea->u_systrap = 0;
                premptyset(&uarea->u_entrymask);
                premptyset(&uarea->u_exitmask);
        }
        /*
         * If microstate accounting is being inherited, mark child
         */
        if ((pp->p_flag & SMSFORK) != 0)
                cp->p_flag |= pp->p_flag & (SMSFORK|SMSACCT);

        /*
         * Inherit fixalignment flag from the parent
         */
        cp->p_fixalignment = pp->p_fixalignment;

        *cpp = cp;
        return (0);

bad:
        ASSERT(MUTEX_NOT_HELD(&pidlock));

        mutex_destroy(&cp->p_crlock);
        mutex_destroy(&cp->p_pflock);
#if defined(__x86)
        mutex_destroy(&cp->p_ldtlock);
#endif
        if (newpid != -1) {
                proc_entry_free(cp->p_pidp);
                (void) pid_rele(cp->p_pidp);
        }
        kmem_cache_free(process_cache, cp);

        mutex_enter(&zone->zone_nlwps_lock);
```

```
        task->tk_nprocs--;
        proj->kpj_nprocs--;
        zone->zone_nprocs--;
        mutex_exit(&zone->zone_nlwps_lock);
        atomic_inc_32(&zone->zone_ffnoproc);

punish:
        /*
         * We most likely got into this situation because some process is
         * forking out of control.  As punishment, put it to sleep for a
         * bit so it can't eat the machine alive.  Sleep interval is chosen
         * to allow no more than one fork failure per cpu per clock tick
         * on average (yes, I just made this up).  This has two desirable
         * properties: (1) it sets a constant limit on the fork failure
         * rate, and (2) the busier the system is, the harsher the penalty
         * for abusing it becomes.
         */
        INCR_COUNT(&fork_fail_pending, &pidlock);
        delay(fork_fail_pending / ncpus + 1);
        DECR_COUNT(&fork_fail_pending, &pidlock);

        return (-1); /* out of memory or proc slots */
}
```

______***unchanged_portion_omitted_***

```
**********************************************************
   35236 Wed Jun 15 19:31:59 2016
new/usr/src/uts/common/sys/procfs.h
Code review comments from jeffpc
**********************************************************
______unchanged_portion_omitted_

#define PRSECFLAGS_VERSION_1            1
#define PRSECFLAGS_VERSION_CURRENT      PRSECFLAGS_VERSION_1
typedef struct prsecflags {
        uint32_t pr_version;
        char pr_pad[4];
#endif /* ! codereview */
        secflagset_t pr_effective;
        secflagset_t pr_inherit;
        secflagset_t pr_lower;
        secflagset_t pr_upper;
} prsecflags_t;

/*
 * Watchpoint interface.  PCWATCH and /proc/<pid>/watch
 */
typedef struct prwatch {
        uintptr_t pr_vaddr;     /* virtual address of watched area */
        size_t  pr_size;        /* size of watched area in bytes */
        int     pr_wflags;      /* watch type flags */
        int     pr_pad;
} prwatch_t;

/* pr_wflags */
#define WA_READ         0x04    /* trap on read access */
#define WA_WRITE        0x02    /* trap on write access */
#define WA_EXEC         0x01    /* trap on execute access */
#define WA_TRAPAFTER    0x08    /* trap after instruction completes */

/*
 * PCREAD/PCWRITE I/O interface.
 */
typedef struct priovec {
        void    *pio_base;      /* buffer in controlling process */
        size_t  pio_len;        /* size of read/write request */
        off_t   pio_offset;     /* virtual address in target process */
} priovec_t;

/*
 * Resource usage.  /proc/<pid>/usage /proc/<pid>/lwp/<lwpid>/lwpusage
 */
typedef struct prusage {
        id_t            pr_lwpid;       /* lwp id.  0: process or defunct */
        int             pr_count;       /* number of contributing lwps */
        timestruc_t     pr_tstamp;      /* current time stamp */
        timestruc_t     pr_create;      /* process/lwp creation time stamp */
        timestruc_t     pr_term;        /* process/lwp termination time stamp */
        timestruc_t     pr_rtime;       /* total lwp real (elapsed) time */
        timestruc_t     pr_utime;       /* user level cpu time */
        timestruc_t     pr_stime;       /* system call cpu time */
        timestruc_t     pr_ttime;       /* other system trap cpu time */
        timestruc_t     pr_tftime;      /* text page fault sleep time */
        timestruc_t     pr_dftime;      /* data page fault sleep time */
        timestruc_t     pr_kftime;      /* kernel page fault sleep time */
        timestruc_t     pr_ltime;       /* user lock wait sleep time */
        timestruc_t     pr_slptime;     /* all other sleep time */
        timestruc_t     pr_wtime;       /* wait-cpu (latency) time */
        timestruc_t     pr_stoptime;    /* stopped time */
        timestruc_t     filltime[6];    /* filler for future expansion */
        ulong_t         pr_minf;        /* minor page faults */
```

```
        ulong_t         pr_majf;        /* major page faults */
        ulong_t         pr_nswap;       /* swaps */
        ulong_t         pr_inblk;       /* input blocks */
        ulong_t         pr_oublk;       /* output blocks */
        ulong_t         pr_msnd;        /* messages sent */
        ulong_t         pr_mrcv;        /* messages received */
        ulong_t         pr_sigs;        /* signals received */
        ulong_t         pr_vctx;        /* voluntary context switches */
        ulong_t         pr_ictx;        /* involuntary context switches */
        ulong_t         pr_sysc;        /* system calls */
        ulong_t         pr_ioch;        /* chars read and written */
        ulong_t         filler[10];     /* filler for future expansion */
} prusage_t;

/*
 * Page data file.  /proc/<pid>/pagedata
 */

/* page data file header */
typedef struct prpageheader {
        timestruc_t     pr_tstamp;      /* real time stamp */
        long            pr_nmap;        /* number of address space mappings */
        long            pr_npage;       /* total number of pages */
} prpageheader_t;

/* page data mapping header */
typedef struct prasmap {
        uintptr_t pr_vaddr;     /* virtual address of mapping */
        size_t  pr_npage;       /* number of pages in mapping */
        char    pr_mapname[PRMAPSZ];    /* name in /proc/<pid>/object */
        offset_t pr_offset;     /* offset into mapped object, if any */
        int     pr_mflags;      /* protection and attribute flags */
        int     pr_pagesize;    /* pagesize (bytes) for this mapping */
        int     pr_shmid;       /* SysV shmid, -1 if not SysV shared memory */
        int     pr_filler[1];   /* filler for future expansion */
} prasmap_t;

/*
 * pr_npage bytes (plus 0-7 null bytes to round up to an 8-byte boundary)
 * follow each mapping header, each containing zero or more of these flags.
 */
#define PG_REFERENCED   0x02            /* page referenced since last read */
#define PG_MODIFIED     0x01            /* page modified since last read */
#define PG_HWMAPPED     0x04            /* page is present and mapped */

/*
 * Open files.  Only in core files (for now).  Note that we'd like to use
 * the stat or stat64 structure, but both of these structures are unfortunately
 * not consistent between 32 and 64 bit modes.  To keep our lives simpler, we
 * just define our own structure with types that are not sensitive to this
 * difference.  Also, it turns out that pfiles omits a lot of info from the
 * struct stat (e.g. times, device sizes, etc.) so we don't bother adding those
 * here.
 */
typedef struct prfdinfo {
        int             pr_fd;
        mode_t          pr_mode;

        uid_t           pr_uid;
        gid_t           pr_gid;

        major_t         pr_major;       /* think stat.st_dev */
        minor_t         pr_minor;

        major_t         pr_rmajor;      /* think stat.st_rdev */
        minor_t         pr_rminor;
        ino64_t         pr_ino;
        off64_t         pr_offset;
        off64_t         pr_size;

        int             pr_fileflags;   /* fcntl(F_GETXFL), etc */
        int             pr_fdflags;     /* fcntl(F_GETFD), etc. */

        char            pr_path[MAXPATHLEN];
} prfdinfo_t;

/*
 * Header for /proc/<pid>/lstatus /proc/<pid>/lpsinfo /proc/<pid>/lusage
 */
typedef struct prheader {
        long    pr_nent;        /* number of entries */
        long    pr_entsize;     /* size of each entry, in bytes */
} prheader_t;

/*
 * Macros for manipulating sets of flags.
 * sp must be a pointer to one of sigset_t, fltset_t, or sysset_t.
 * flag must be a member of the enumeration corresponding to *sp.
 */

/* turn on all flags in set */
#define prfillset(sp) \
        { register int _i_ = sizeof (*(sp))/sizeof (uint32_t); \
                while (_i_) ((uint32_t *)(sp))[--_i_] = (uint32_t)0xFFFFFFFF; }

/* turn off all flags in set */
#define premptyset(sp) \
        { register int _i_ = sizeof (*(sp))/sizeof (uint32_t); \
                while (_i_) ((uint32_t *)(sp))[--_i_] = (uint32_t)0; }

/* turn on specified flag in set */
#define praddset(sp, flag) \
        ((void)(((unsigned)((flag)-1) < 32*sizeof (*(sp))/sizeof (uint32_t)) ? \
        (((uint32_t *)(sp))[((flag)-1)/32] |= (1U<<(((flag)-1)%32))) : 0))

/* turn off specified flag in set */
#define prdelset(sp, flag) \
        ((void)(((unsigned)((flag)-1) < 32*sizeof (*(sp))/sizeof (uint32_t)) ? \
            (((uint32_t *)(sp))[((flag)-1)/32] &= ~(1U<<(((flag)-1)%32))) : 0))

/* query: != 0 iff flag is turned on in set */
#define prismember(sp, flag) \
        (((unsigned)((flag)-1) < 32*sizeof (*(sp))/sizeof (uint32_t)) && \
            (((uint32_t *)(sp))[((flag)-1)/32] & (1U<<(((flag)-1)%32))))

#if defined(_SYSCALL32)

/*
 * dev32_t version of PRNODEV
 */
#define PRNODEV32 (dev32_t)(-1)

/*
 * Kernel view of /proc structures for _ILP32 programs.
 */

/*
 * _ILP32 lwp status file.  /proc/<pid>/lwp/<lwpid>/lwpstatus
 */
typedef struct lwpstatus32 {
        int     pr_flags;       /* flags */
```

```
        id32_t  pr_lwpid;       /* specific lwp identifier */
        short   pr_why;         /* reason for lwp stop, if stopped */
        short   pr_what;        /* more detailed reason */
        short   pr_cursig;      /* current signal, if any */
        short   pr_pad1;
        siginfo32_t pr_info;    /* info associated with signal or fault */
        sigset_t pr_lwppend;    /* set of signals pending to the lwp */
        sigset_t pr_lwphold;    /* set of signals blocked by the lwp */
        struct sigaction32 pr_action;   /* signal action for current signal */
        stack32_t pr_altstack;  /* alternate signal stack info */
        caddr32_t pr_oldcontext;        /* address of previous ucontext */
        short   pr_syscall;     /* system call number (if in syscall) */
        short   pr_nsysarg;     /* number of arguments to this syscall */
        int     pr_errno;       /* errno for failed syscall, 0 if successful */
        int32_t pr_sysarg[PRSYSARGS];   /* arguments to this syscall */
        int32_t pr_rval1;       /* primary syscall return value */
        int32_t pr_rval2;       /* second syscall return value, if any */
        char    pr_clname[PRCLSZ];      /* scheduling class name */
        timestruc32_t pr_tstamp;        /* real-time time stamp of stop */
        timestruc32_t pr_utime; /* lwp user cpu time */
        timestruc32_t pr_stime; /* lwp system cpu time */
        int     pr_filler[11 - 2 * sizeof (timestruc32_t) / sizeof (int)];
        int     pr_errpriv;     /* missing privilege */
        caddr32_t pr_ustack;    /* address of stack boundary data (stack32_t) */
        uint32_t pr_instr;      /* current instruction */
        prgregset32_t pr_reg;   /* general registers */
        prfpregset32_t pr_fpreg; /* floating-point registers */
} lwpstatus32_t;

/*
 * _ILP32 process status file.  /proc/<pid>/status
 */
typedef struct pstatus32 {
        int     pr_flags;       /* flags */
        int     pr_nlwp;        /* number of active lwps in the process */
        pid32_t pr_pid;         /* process id */
        pid32_t pr_ppid;        /* parent process id */
        pid32_t pr_pgid;        /* process group id */
        pid32_t pr_sid;         /* session id */
        id32_t  pr_aslwpid;     /* historical; now always zero */
        id32_t  pr_agentid;     /* lwp id of the /proc agent lwp, if any */
        sigset_t pr_sigpend;    /* set of process pending signals */
        caddr32_t pr_brkbase;   /* address of the process heap */
        size32_t pr_brksize;    /* size of the process heap, in bytes */
        caddr32_t pr_stkbase;   /* address of the process stack */
        size32_t pr_stksize;    /* size of the process stack, in bytes */
        timestruc32_t pr_utime; /* process user cpu time */
        timestruc32_t pr_stime; /* process system cpu time */
        timestruc32_t pr_cutime;        /* sum of children's user times */
        timestruc32_t pr_cstime;        /* sum of children's system times */
        sigset_t pr_sigtrace;   /* set of traced signals */
        fltset_t pr_flttrace;   /* set of traced faults */
        sysset_t pr_sysentry;   /* set of system calls traced on entry */
        sysset_t pr_sysexit;    /* set of system calls traced on exit */
        char    pr_dmodel;      /* data model of the process */
        char    pr_pad[3];
        id32_t  pr_taskid;      /* task id */
        id32_t  pr_projid;      /* project id */
        int     pr_nzomb;       /* number of zombie lwps in the process */
        id32_t  pr_zoneid;      /* zone id */
        int     pr_filler[15];  /* reserved for future use */
        lwpstatus32_t pr_lwp;   /* status of the representative lwp */
} pstatus32_t;

/*
 * _ILP32 lwp ps(1) information file.  /proc/<pid>/lwp/<lwpid>/lwpsinfo
 */
typedef struct lwpsinfo32 {
        int     pr_flag;        /* lwp flags */
        id32_t  pr_lwpid;       /* lwp id */
        caddr32_t pr_addr;      /* internal address of lwp */
        caddr32_t pr_wchan;     /* wait addr for sleeping lwp */
        char    pr_stype;       /* synchronization event type */
        char    pr_state;       /* numeric lwp state */
        char    pr_sname;       /* printable character for pr_state */
        char    pr_nice;        /* nice for cpu usage */
        short   pr_syscall;     /* system call number (if in syscall) */
        char    pr_oldpri;      /* pre-SVR4, low value is high priority */
        char    pr_cpu;         /* pre-SVR4, cpu usage for scheduling */
        int     pr_pri;         /* priority, high value is high priority */
                        /* The following percent number is a 16-bit binary */
                        /* fraction [0 .. 1] with the binary point to the */
                        /* right of the high-order bit (1.0 == 0x8000) */
        ushort_t pr_pctcpu;     /* % of recent cpu time used by this lwp */
        ushort_t pr_pad;
        timestruc32_t pr_start; /* lwp start time, from the epoch */
        timestruc32_t pr_time;  /* usr+sys cpu time for this lwp */
        char    pr_clname[PRCLSZ];      /* scheduling class name */
        char    pr_name[PRFNSZ];        /* name of system lwp */
        processorid_t pr_onpro;         /* processor which last ran this lwp */
        processorid_t pr_bindpro;       /* processor to which lwp is bound */
        psetid_t pr_bindpset;   /* processor set to which lwp is bound */
        int     pr_lgrp;        /* lwp home lgroup */
        int     pr_filler[4];   /* reserved for future use */
} lwpsinfo32_t;

/*
 * _ILP32 process ps(1) information file.  /proc/<pid>/psinfo
 */
typedef struct psinfo32 {
        int     pr_flag;        /* process flags */
        int     pr_nlwp;        /* number of active lwps in the process */
        pid32_t pr_pid;         /* unique process id */
        pid32_t pr_ppid;        /* process id of parent */
        pid32_t pr_pgid;        /* pid of process group leader */
        pid32_t pr_sid;         /* session id */
        uid32_t pr_uid;         /* real user id */
        uid32_t pr_euid;        /* effective user id */
        gid32_t pr_gid;         /* real group id */
        gid32_t pr_egid;        /* effective group id */
        caddr32_t pr_addr;      /* address of process */
        size32_t pr_size;       /* size of process image in Kbytes */
        size32_t pr_rssize;     /* resident set size in Kbytes */
        size32_t pr_pad1;
        dev32_t pr_ttydev;      /* controlling tty device (or PRNODEV) */
        ushort_t pr_pctcpu;     /* % of recent cpu time used by all lwps */
        ushort_t pr_pctmem;     /* % of system memory used by process */
        timestruc32_t pr_start; /* process start time, from the epoch */
        timestruc32_t pr_time;  /* usr+sys cpu time for this process */
        timestruc32_t pr_ctime; /* usr+sys cpu time for reaped children */
        char    pr_fname[PRFNSZ];       /* name of execed file */
        char    pr_psargs[PRARGSZ];     /* initial characters of arg list */
        int     pr_wstat;       /* if zombie, the wait() status */
        int     pr_argc;        /* initial argument count */
        caddr32_t pr_argv;      /* address of initial argument vector */
        caddr32_t pr_envp;      /* address of initial environment vector */
        char    pr_dmodel;      /* data model of the process */
        char    pr_pad2[3];
        id32_t  pr_taskid;      /* task id */
        id32_t  pr_projid;      /* project id */
        int     pr_nzomb;       /* number of zombie lwps in the process */
        id32_t  pr_poolid;      /* pool id */
```

```
        id32_t  pr_zoneid;      /* zone id */
        id32_t  pr_contract;    /* process contract */
        int     pr_filler[1];   /* reserved for future use */
        lwpsinfo32_t pr_lwp;    /* information for representative lwp */
} psinfo32_t;

/*
 * _ILP32 Memory-management interface.  /proc/<pid>/map /proc/<pid>/rmap
 */
typedef struct prmap32 {
        caddr32_t pr_vaddr;     /* virtual address of mapping */
        size32_t pr_size;       /* size of mapping in bytes */
        char    pr_mapname[64]; /* name in /proc/<pid>/object */
        offset_t pr_offset;     /* offset into mapped object, if any */
        int     pr_mflags;      /* protection and attribute flags */
        int     pr_pagesize;    /* pagesize (bytes) for this mapping */
        int     pr_shmid;       /* SysV shmid, -1 if not SysV shared memory */
        int     pr_filler[1];   /* filler for future expansion */
} prmap32_t;

/*
 * _ILP32 HAT memory-map interface.  /proc/<pid>/xmap
 */
typedef struct prxmap32 {
        caddr32_t pr_vaddr;     /* virtual address of mapping */
        size32_t pr_size;       /* size of mapping in bytes */
        char    pr_mapname[PRMAPSZ];    /* name in /proc/<pid>/object */
        offset_t pr_offset;     /* offset into mapped object, if any */
        int     pr_mflags;      /* protection and attribute flags (see below) */
        int     pr_pagesize;    /* pagesize (bytes) for this mapping */
        int     pr_shmid;       /* SysV shmid, -1 if not SysV shared memory */
        dev32_t pr_dev;  /* st_dev from stat64() of mapped object, or PRNODEV */
        uint64_t pr_ino; /* st_ino from stat64() of mapped object, if any */
        uint32_t pr_rss;        /* pages of resident memory */
        uint32_t pr_anon;       /* pages of resident anonymous memory */
        uint32_t pr_locked;     /* pages of locked memory */
        uint32_t pr_pad;        /* currently unused */
        uint64_t pr_hatpagesize; /* pagesize of the hat mapping */
        uint32_t pr_filler[6];  /* filler for future expansion */
} prxmap32_t;

/*
 * _ILP32 Process credentials.  PCSCRED and /proc/<pid>/cred
 */
typedef struct prcred32 {
        uid32_t pr_euid;        /* effective user id */
        uid32_t pr_ruid;        /* real user id */
        uid32_t pr_suid;        /* saved user id (from exec) */
        gid32_t pr_egid;        /* effective group id */
        gid32_t pr_rgid;        /* real group id */
        gid32_t pr_sgid;        /* saved group id (from exec) */
        int     pr_ngroups;     /* number of supplementary groups */
        gid32_t pr_groups[1];   /* array of supplementary groups */
} prcred32_t;

/*
 * _ILP32 Watchpoint interface.  PCWATCH and /proc/<pid>/watch
 */
typedef struct prwatch32 {
        caddr32_t pr_vaddr;     /* virtual address of watched area */
        size32_t pr_size;       /* size of watched area in bytes */
        int     pr_wflags;      /* watch type flags */
        int     pr_pad;
} prwatch32_t;

/*
```

```
 * _ILP32 PCREAD/PCWRITE I/O interface.
 */
typedef struct priovec32 {
        caddr32_t pio_base;     /* buffer in controlling process */
        size32_t pio_len;       /* size of read/write request */
        off32_t pio_offset;     /* virtual address in target process */
} priovec32_t;

/*
 * _ILP32 Resource usage.  /proc/<pid>/usage /proc/<pid>/lwp/<lwpid>/lwpusage
 */
typedef struct prusage32 {
        id32_t          pr_lwpid;       /* lwp id.  0: process or defunct */
        int32_t         pr_count;       /* number of contributing lwps */
        timestruc32_t   pr_tstamp;      /* current time stamp */
        timestruc32_t   pr_create;      /* process/lwp creation time stamp */
        timestruc32_t   pr_term;        /* process/lwp termination time stamp */
        timestruc32_t   pr_rtime;       /* total lwp real (elapsed) time */
        timestruc32_t   pr_utime;       /* user level cpu time */
        timestruc32_t   pr_stime;       /* system call cpu time */
        timestruc32_t   pr_ttime;       /* other system trap cpu time */
        timestruc32_t   pr_tftime;      /* text page fault sleep time */
        timestruc32_t   pr_dftime;      /* data page fault sleep time */
        timestruc32_t   pr_kftime;      /* kernel page fault sleep time */
        timestruc32_t   pr_ltime;       /* user lock wait sleep time */
        timestruc32_t   pr_slptime;     /* all other sleep time */
        timestruc32_t   pr_wtime;       /* wait-cpu (latency) time */
        timestruc32_t   pr_stoptime;    /* stopped time */
        timestruc32_t   filltime[6];    /* filler for future expansion */
        uint32_t        pr_minf;        /* minor page faults */
        uint32_t        pr_majf;        /* major page faults */
        uint32_t        pr_nswap;       /* swaps */
        uint32_t        pr_inblk;       /* input blocks */
        uint32_t        pr_oublk;       /* output blocks */
        uint32_t        pr_msnd;        /* messages sent */
        uint32_t        pr_mrcv;        /* messages received */
        uint32_t        pr_sigs;        /* signals received */
        uint32_t        pr_vctx;        /* voluntary context switches */
        uint32_t        pr_ictx;        /* involuntary context switches */
        uint32_t        pr_sysc;        /* system calls */
        uint32_t        pr_ioch;        /* chars read and written */
        uint32_t        filler[10];     /* filler for future expansion */
} prusage32_t;

/*
 * _ILP32 Page data file.  /proc/<pid>/pagedata
 */

/* _ILP32 page data file header */
typedef struct prpageheader32 {
        timestruc32_t   pr_tstamp;      /* real time stamp */
        int32_t         pr_nmap;        /* number of address space mappings */
        int32_t         pr_npage;       /* total number of pages */
} prpageheader32_t;

/* _ILP32 page data mapping header */
typedef struct prasmap32 {
        caddr32_t pr_vaddr;     /* virtual address of mapping */
        size32_t pr_npage;      /* number of pages in mapping */
        char    pr_mapname[64]; /* name in /proc/<pid>/object */
        offset_t pr_offset;     /* offset into mapped object, if any */
        int     pr_mflags;      /* protection and attribute flags */
        int     pr_pagesize;    /* pagesize (bytes) for this mapping */
        int     pr_shmid;       /* SysV shmid, -1 if not SysV shared memory */
        int     pr_filler[1];   /* filler for future expansion */
} prasmap32_t;
```

```
/*
 * _ILP32 Header for /proc/<pid>/lstatus /proc/<pid>/lpsinfo /proc/<pid>/lusage
 */
typedef struct prheader32 {
        int32_t pr_nent;        /* number of entries */
        int32_t pr_entsize;     /* size of each entry, in bytes */
} prheader32_t;

#endif  /* _SYSCALL32 */

#endif  /* !_KERNEL && _STRUCTURED_PROC == 0 */

#ifdef  __cplusplus
}
#endif

#endif  /* _SYS_PROCFS_H */
```

****
3060 Wed Jun 15 19:32:00 2016
new/usr/src/uts/common/sys/secflags.h
Code review comments from jeffpc
****

```
/*
 * 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>
#include <sys/procset.h>

struct proc;
typedef uint64_t secflagset_t;
typedef uint32_t secflagset_t;

typedef struct psecflags {
        secflagset_t psf_effective;
        secflagset_t psf_inherit;
        secflagset_t psf_lower;
        secflagset_t psf_upper;
} psecflags_t;
```
______***unchanged_portion_omitted_***