```
**********************************************************
    2854 Sat Jun  8 20:37:50 2013
new/usr/src/cmd/initpkg/Makefile
3788 /etc/bootrc is defunct and should be removed
**********************************************************
```

```
   1 #
   2 # CDDL HEADER START
   3 #
   4 # The contents of this file are subject to the terms of the
   5 # Common Development and Distribution License (the "License").
   6 # You may not use this file except in compliance with the License.
   7 #
   8 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
   9 # or http://www.opensolaris.org/os/licensing.
  10 # See the License for the specific language governing permissions
  11 # and limitations under the License.
  12 #
  13 # When distributing Covered Code, include this CDDL HEADER in each
  14 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  15 # If applicable, add the following below this CDDL HEADER, with the
  16 # fields enclosed by brackets "[]" replaced with your own identifying
  17 # information: Portions Copyright [yyyy] [name of copyright owner]
  18 #
  19 # CDDL HEADER END
  20 #
  21 #
  22 # Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
  23 # Use is subject to license terms.
  24 #

  26 SHFILES=        dfstab vfstab
  27 CPFILES=        rcS rc0 rc1 rc2 rc3 mountall shutdown swapadd umountall
  28 ALL=            $(SHFILES) $(CPFILES)
  29 TXTS=           rc2.d/mk.rc2.d.sh
  30 CLOBBERFILES=   $(ALL)
  31 RCDIRS=         rc2.d

  33 include ../Makefile.cmd

  35 ETCDFSD=        $(ROOTETC)/dfs

  37 SBINF=          rcS mountall rc0 rc1 rc2 rc3 swapadd umountall
  38 SBINL=          rc5 rc6
  39 USRSBINF=       mountall shutdown umountall

  41 ETCTABS=        vfstab inittab nscd.conf security/crypt.conf
  41 sparc_ETCTABS=
  42 i386_ETCTABS=   bootrc
  43 ETCTABS=        vfstab inittab nscd.conf security/crypt.conf $($(MACH)_ETCTABS)

  43 DFSTAB=         dfstab
  44 SBINETC=        rcS mountall rc0 rc1 rc2 rc3 rc5 rc6 swapadd umountall
  45 USRSBINETC=     shutdown

  47 FILEMODE=       0744

  49 ROOTSBINF=      $(SBINF:%=$(ROOTSBIN)/%)
  50 ROOTSBINL=      $(SBINL:%=$(ROOTSBIN)/%)
  51 ROOTUSRSBINF=   $(USRSBINF:%=$(ROOTUSRSBIN)/%)
  52 ROOTETCTABS=    $(ETCTABS:%=$(ROOTETC)/%)
  53 ROOTDFSTAB=     $(DFSTAB:%=$(ETCDFSD)/%)
  54 SYMSBINF=       $(SBINETC:%=$(ROOTETC)/%)
  55 SYMUSRSBINF=    $(USRSBINETC:%=$(ROOTETC)/%)

  57 $(ROOTETC)/inittab              := FILEMODE =   0644
  58 $(ROOTETC)/vfstab               := FILEMODE =   0644
```



```
  59 $(ROOTETC)/nscd.conf            := FILEMODE =   0644
  60 $(ROOTETC)/security/crypt.conf  := FILEMODE =   0644
  63 $(ROOTETC)/bootrc               := FILEMODE =   0755
  61 $(ROOTDFSTAB)                   := FILEMODE =   0644
  62 $(ROOTSBIN)/mountall            := FILEMODE =   0555
  63 $(ROOTUSRSBIN)/mountall         := FILEMODE =   0555
  64 $(ROOTSBIN)/umountall           := FILEMODE =   0555
  65 $(ROOTUSRSBIN)/umountall        := FILEMODE =   0555
  66 $(ROOTUSRSBIN)/shutdown         := FILEMODE =   0755

  68 $(ETCDFSD)/% : %
  69         $(INS.file)

  71 .KEEP_STATE:

  73 all: $(ALL) all_init.d $(TXTS)

  75 $(SYMSBINF):
  76         $(RM) $@; $(SYMLINK) ../sbin/$(@F) $@

  78 $(SYMUSRSBINF):
  79         $(RM) $@; $(SYMLINK) ../usr/sbin/$(@F) $@

  81 $(ROOTSBINL):   $(ROOTSBIN)/rc0
  82         $(RM) $@; $(LN) $(ROOTSBIN)/rc0 $@

  84 all_init.d: FRC
  85         @cd init.d; pwd; $(MAKE) $(MFLAGS) all

  87 ins_init.d: FRC
  88         @cd init.d; pwd; $(MAKE) $(MFLAGS) install

  90 $(SHFILES):
  91         sh $@.sh $(ROOT)

  93 install: $(ALL) ins_all ins_init.d $(RCDIRS)

  95 ins_all : $(ROOTSBINF) $(ROOTSBINL) $(ROOTUSRSBINF) $(ROOTETCTABS) \
  96         $(ROOTDFSTAB)  $(SYMSBINF) $(SYMUSRSBINF)

  98 # Don't re-install directories already installed by Targetdirs
  99 #$(DIRS):
 100 #       $(INS.dir)

 102 $(RCDIRS):      FRC
 103         @cd $@; pwd; ROOT=$(ROOT) sh mk.$@.sh

 105 FRC:

 107 clean lint:

 109 include ../Makefile.targ
```

****
8214 Sat Jun 8 20:37:52 2013
new/usr/src/cmd/prtconf/prtconf.c
3788 /etc/bootrc is defunct and should be removed
****
_____unchanged_portion_omitted_

```
 162 /*
 163  * debug version has two more flags:
 164  *      -L force load driver
 165  *      -M: print per driver list
 166  */

 168 #ifdef  DEBUG
 169 static const char *optstring = "abcdDvVxpPFf:M:dLuC";
 170 #else
 171 static const char *optstring = "abcdDvVxpPFf:uC";
 172 #endif  /* DEBUG */

 174 int
 175 main(int argc, char *argv[])
 176 {
 177         long pagesize, npages;
 178         int c, ret;
 179         char hw_provider[SYS_NMLN];

 181         setpname(argv[0]);
 182         opts.o_promdev = "/dev/openprom";

 184         while ((c = getopt(argc, argv, optstring)) != -1)  {
 185                 switch (c)  {
 186                 case 'a':
 187                         ++opts.o_ancestors;
 188                         break;
 189                 case 'b':
 190                         ++opts.o_productinfo;
 191                         break;
 192                 case 'c':
 193                         ++opts.o_children;
 194                         break;
 195                 case 'd':
 196                         ++opts.o_pciid;
 197                         break;
 198                 case 'D':
 199                         ++opts.o_drv_name;
 200                         break;
 201                 case 'v':
 202                         ++opts.o_verbose;
 203                         break;
 204                 case 'p':
 205                         ++opts.o_prominfo;
 206                         break;
 207                 case 'f':
 208                         opts.o_promdev = optarg;
 209                         break;
 210                 case 'V':
 211                         ++opts.o_promversion;
 212                         break;
 213                 case 'x':
 214                         ++opts.o_prom_ready64;
 215                         break;
 216                 case 'F':
 217                         ++opts.o_fbname;
 218                         ++opts.o_noheader;
 219                         break;
```



```
 220                 case 'P':
 221                         ++opts.o_pseudodevs;
 222                         break;
 223                 case 'C':
 224                         ++opts.o_forcecache;
 225                         break;
 226 #ifdef  DEBUG
 227                 case 'M':
 228                         dbg.d_drivername = optarg;
 229                         ++dbg.d_bydriver;
 230                         break;
 231                 case 'L':
 232                         ++dbg.d_forceload;
 233                         break;
 234 #endif  /* DEBUG */

 236                 default:
 237                         (void) fprintf(stderr, usage, opts.o_progname);
 238                         return (1);
 239                 }
 240         }

 242         (void) uname(&opts.o_uts);

 244         if (opts.o_fbname)
 245                 return (do_fbname());

 247         if (opts.o_promversion)
 248                 return (do_promversion());

 250         if (opts.o_prom_ready64)
 251                 return (do_prom_version64());

 253         if (opts.o_productinfo)
 254                 return (do_productinfo());

 256         opts.o_devices_path = NULL;
 257         opts.o_devt = DDI_DEV_T_NONE;
 258         opts.o_target = 0;
 259         if (optind < argc) {
 260                 struct stat     sinfo;
 261                 char            *path = argv[optind];
 262                 int             error;

 264                 if (opts.o_prominfo) {
 265                         /* PROM tree cannot be used with path */
 266                         (void) fprintf(stderr, "%s: path and -p option are "
 267                             "mutually exclusive\n", opts.o_progname);
 268                         return (1);
 269                 }

 271                 if (strlen(path) >= MAXPATHLEN) {
 272                         (void) fprintf(stderr, "%s: "
 273                             "path specified is too long\n", opts.o_progname);
 274                         return (1);
 275                 }

 277                 if (error = stat(path, &sinfo)) {

 279                         /* an invalid path was specified */
 280                         (void) fprintf(stderr, "%s: invalid path specified\n",
 281                             opts.o_progname);
 282                         return (1);

 284                 } else if (((sinfo.st_mode & S_IFMT) == S_IFCHR) ||
 285                     ((sinfo.st_mode & S_IFMT) == S_IFBLK)) {
```

```
287                         opts.o_devt = sinfo.st_rdev;
288                         error = 0;

290                 } else if ((sinfo.st_mode & S_IFMT) == S_IFDIR) {
291                         size_t  len, plen;

293                         /* clean up the path */
294                         cleanup_path(path, new_path);

296                         len = strlen(new_path);
297                         plen = strlen("/devices");
298                         if (len < plen) {
299                                 /* This is not a valid /devices path */
300                                 error = 1;
301                         } else if ((len == plen) &&
302                             (strcmp(new_path, "/devices") == 0)) {
303                                 /* /devices is the root nexus */
304                                 opts.o_devices_path = "/";
305                                 error = 0;
306                         } else if (strncmp(new_path, "/devices/", plen + 1)) {
307                                 /* This is not a valid /devices path */
308                                 error = 1;
309                         } else {
310                                 /* a /devices/ path was specified */
311                                 opts.o_devices_path = new_path + plen;
312                                 error = 0;
313                         }

315                 } else {
316                         /* an invalid device path was specified */
317                         error = 1;
318                 }

320                 if (error) {
321                         (void) fprintf(stderr, "%s: "
322                             "invalid device path specified\n",
323                             opts.o_progname);
324                         return (1);
325                 }

327                 opts.o_target = 1;
328         }

330         if ((opts.o_ancestors || opts.o_children) && (!opts.o_target)) {
331                 (void) fprintf(stderr, "%s: options require a device path\n",
332                     opts.o_progname);
333                 return (1);
334         }

336         if (opts.o_target) {
337                 prtconf_devinfo();
338                 return (0);
339         }

341         ret = sysinfo(SI_HW_PROVIDER, hw_provider, sizeof (hw_provider));
342         /*
343          * If 0 bytes are returned (the system returns '1', for the \0),
344          * we're probably on x86, default to Oracle.
344          * we're probably on x86, and there has been no si-hw-provider
345          * set in /etc/bootrc, default to Oracle.
345          */
346         if (ret <= 1) {
347                 (void) strncpy(hw_provider, "Oracle Corporation",
348                     sizeof (hw_provider));
349         }
```



```
350         (void) printf("System Configuration:  %s  %s\n", hw_provider,
351             opts.o_uts.machine);

353         pagesize = sysconf(_SC_PAGESIZE);
354         npages = sysconf(_SC_PHYS_PAGES);
355         (void) printf("Memory size: ");
356         if (pagesize == -1 || npages == -1)
357                 (void) printf("unable to determine\n");
358         else {
359                 const int64_t kbyte = 1024;
360                 const int64_t mbyte = 1024 * 1024;
361                 int64_t ii = (int64_t)pagesize * npages;

363                 if (ii >= mbyte)
364                         (void) printf("%ld Megabytes\n",
365                             (long)((ii+mbyte-1) / mbyte));
366                 else
367                         (void) printf("%ld Kilobytes\n",
368                             (long)((ii+kbyte-1) / kbyte));
369         }

371         if (opts.o_prominfo) {
372                 (void) printf("System Peripherals (PROM Nodes):\n\n");
373                 if (do_prominfo() == 0)
374                         return (0);
375                 (void) fprintf(stderr, "%s: Defaulting to non-PROM mode...\n",
376                     opts.o_progname);
377         }

379         (void) printf("System Peripherals (Software Nodes):\n\n");

381         (void) prtconf_devinfo();

383         return (0);
384 }
```

*unchanged_portion_omitted_*

```
**********************************************************
    2754 Sat Jun  8 20:37:52 2013
new/usr/src/pkg/manifests/driver-network-platform.mf
3788 /etc/bootrc is defunct and should be removed
**********************************************************
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#

#
# Copyright (c) 2010, Oracle and/or its affiliates. All rights reserved.
#

#
# The default for payload-bearing actions in this package is to appear in the
# global zone only.  See the include file for greater detail, as well as
# information about overriding the defaults.
#
<include global_zone_only_component>
set name=pkg.fmri value=pkg:/driver/network/platform@$(PKGVERS)
set name=pkg.description value="Platform Support, OS Functionality"
set name=pkg.summary value="Platform Support, OS Functionality"
set name=info.classification \
    value=org.opensolaris.category.2008:Drivers/Networking
set name=variant.arch value=i386
dir path=etc group=sys
dir path=kernel group=sys
dir path=kernel/drv group=sys
dir path=kernel/drv/$(ARCH64) group=sys
dir path=usr/share/man
dir path=usr/share/man/man7d
driver name=dnet clone_perms="dnet 0666 root sys" perms="* 0666 root sys" \
    alias=pci1011,14 \
    alias=pci1011,19 \
    alias=pci1011,2 \
    alias=pci1011,9 \
    alias=pci10b8,2001 \
    alias=pci1109,1400 \
    alias=pci1109,2400 \
    alias=pci2646,1
file path=etc/bootrc group=sys mode=0755
file path=etc/mach group=sys original_name=SUNWos86r:etc/mach preserve=true
file path=kernel/drv/$(ARCH64)/dnet group=sys
file path=kernel/drv/dnet group=sys
file path=kernel/drv/sd group=sys
file path=kernel/drv/sd.conf group=sys \
    original_name=SUNWos86r:kernel/drv/sd.conf preserve=true
file path=usr/share/man/man7d/dnet.7d
legacy pkg=SUNWos86r desc="Platform Support, OS Functionality (Root)" \
```



```
    name="Platform Support, OS Functionality (Root)"
license cr_Sun license=cr_Sun
license lic_CDDL license=lic_CDDL
# elxl moved out of this package, so create a dependency for upgraded systems
depend fmri=driver/network/elxl type=require
# iprb moved out of this package, so create a dependency for upgraded systems
depend fmri=driver/network/iprb type=require
# pcn moved out of this package, so create a dependency for upgraded systems
depend fmri=driver/network/pcn type=require
```

```
**********************************************************
   21322 Sat Jun  8 20:37:53 2013
new/usr/src/uts/common/conf/param.c
3788 /etc/bootrc is defunct and should be removed
**********************************************************
______unchanged_portion_omitted_

 459 int loaded_classes = 1;         /* for loaded classes */
 460 kmutex_t class_lock;            /* lock for class[] */

 462 int nclass = sizeof (sclass) / sizeof (sclass_t);
 463 char initcls[] = "TS";
 464 char *defaultclass = initcls;

 466 /*
 467  * Tunable system parameters.
 468  */

 470 /*
 471  * The integers tune_* are done this way so that the tune
 472  * data structure may be "tuned" if necessary from the /etc/system
 473  * file. The tune data structure is initialized in param_init();
 474  */

 476 tune_t tune;

 478 /*
 479  * If freemem < t_getpgslow, then start to steal pages from processes.
 480  */
 481 int tune_t_gpgslo = 25;

 483 /*
 484  * Rate at which fsflush is run, in seconds.
 485  */
 486 #define DEFAULT_TUNE_T_FSFLUSHR 1
 487 int tune_t_fsflushr = DEFAULT_TUNE_T_FSFLUSHR;

 489 /*
 490  * The minimum available resident (not swappable) memory to maintain
 491  * in order to avoid deadlock.  In pages.
 492  */
 493 int tune_t_minarmem = 25;

 495 /*
 496  * The minimum available swappable memory to maintain in order to avoid
 497  * deadlock.  In pages.
 498  */
 499 int tune_t_minasmem = 25;

 501 int tune_t_flckrec = 512;       /* max # of active frlocks */

 503 /*
 504  * Number of currently available pages that cannot be 'locked'
 505  * This is set in init_pages_pp_maximum, and must be initialized
 506  * to zero here to detect an override in /etc/system
 507  */
 508 pgcnt_t pages_pp_maximum = 0;

 510 int boothowto;                  /* boot flags passed to kernel */
 511 struct var v;                   /* System Configuration Information */

 513 /*
 514  * System Configuration Information
 515  */

 517 /*
```



```
 518  * The physical system's host identifier, expressed as a decimal string.
 519  * Code should only directly access this value when writing to it (setting the
 520  * physical system's host identifier).  Code that reads the physical system's
 521  * host identifier should use zone_get_hostid(NULL) instead.
 522  */
 523 char hw_serial[HW_HOSTID_LEN] = "0";

 525 #if defined(__sparc)

 527 /*
 528  * On sparc machines, read hw_serial from the firmware at boot time
 529  * and simply assert Oracle is the hardware provider.
 530  */
 531 char architecture[] = "sparcv9";
 532 char architecture_32[] = "sparc";
 533 char hw_provider[] = "Oracle Corporation";

 535 #elif defined(__i386)

 537 /*
 538  * On x86 machines, read hw_serial, hw_provider and srpc_domain from
 539  * /etc/bootrc at boot time.
 540  */
 537 char architecture[] = "i386";
 538 char architecture_32[] = "i386";
 539 char hw_provider[SYS_NMLN] = "";

 541 #elif defined(__amd64)

 547 /*
 548  * On amd64 machines, read hw_serial, hw_provider and srpc_domain from
 549  * /etc/bootrc at boot time.
 550  */
 543 char architecture[] = "amd64";
 544 char architecture_32[] = "i386";
 545 char hw_provider[SYS_NMLN] = "";

 547 #else
 548 #error "unknown processor architecture"
 549 #endif

 551 char srpc_domain[SYS_NMLN] = "";
 552 char platform[SYS_NMLN] = "";   /* read from the devinfo root node */

 554 /* Initialize isa_list */
 555 char *isa_list = architecture;

 557 static pgcnt_t original_physmem = 0;

 559 #define MIN_DEFAULT_MAXUSERS    8u
 560 #define MAX_DEFAULT_MAXUSERS    2048u
 561 #define MAX_MAXUSERS            4096u

 563 void
 564 param_preset(void)
 565 {
 566         original_physmem = physmem;
 567 }
______unchanged_portion_omitted_
```

```
**********************************************************
    6149 Sat Jun  8 20:37:54 2013
new/usr/src/uts/common/krtld/kobj_bootflags.c
3788 /etc/bootrc is defunct and should be removed
fix _OBP comment in kobj_bootflags.c
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright 2010 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */


#include <sys/types.h>
#include <sys/reboot.h>
#include <sys/cmn_err.h>
#include <sys/bootconf.h>
#include <sys/promif.h>
#include <sys/obpdefs.h>
#include <sys/sunddi.h>
#include <sys/systm.h>
#include <sys/kobj.h>
#include <sys/kobj_impl.h>
#include <util/getoptstr.h>

char *kobj_kmdb_argv[11];       /* 10 arguments and trailing NULL */

/*
 * Parse the boot line to determine boot flags.
 */
void
bootflags(struct bootops *ops)
{
        struct gos_params params;
        uchar_t num_O_opt = 0;
        char *cp;
        int c;
        char scratch[BOOTARGS_MAX];

        if (BOP_GETPROP(ops, "bootargs", kern_bootargs) == -1) {
                boothowto |= RB_ASKNAME;
                return;
        }

        (void) BOP_GETPROP(ops, "boot-file", kern_bootfile);

        cp = kern_bootargs;
```



```
#if defined(_OBP)
        /*
         * Sparc only, _OBP isn't defined on x86 any more.
         * x86: The boot scripts (i.e., /etc/bootrc) don't prepend the kernel
         * name to the boot arguments.  (And beware making it do so: if the
         * run-kernel command returns, it will loop, and you will end up with
         * multiple copies of the kernel name.)
         */
        if (cp[0] != '-') {
                /* if user booted kadb or kmdb, load kmdb */
                if (cp[0] == 'k' && (cp[1] == 'a' || cp[1] == 'm') &&
                    cp[2] == 'd' && cp[3] == 'b' &&
                    (cp[4] == ' ' || cp[4] == '	' || cp[4] == 0))
                        boothowto |= RB_KMDB;
                SKIP_WORD(cp);          /* Skip the kernel's filename. */
        }
#endif
        SKIP_SPC(cp);

#if defined(_OBP)
        /* skip bootblk args */
        params.gos_opts = "abcdDf:F:gGHhi:km:o:O:rsvVwxZ:";
#else
        params.gos_opts = "abcdgGhi:km:O:rsvwx";
#endif
        params.gos_strp = cp;
        getoptstr_init(&params);
        while ((c = getoptstr(&params)) != -1) {

                switch (c) {
                case 'a':
                        boothowto |= RB_ASKNAME;
                        break;
                case 'b':
                        boothowto |= RB_NOBOOTRC;
                        break;
                case 'c':
                        boothowto |= RB_CONFIG;
                        break;
                case 'd':
                        boothowto |= RB_DEBUGENTER;
                        break;
#if defined(_OBP)
                case 'D':
                case 'F':
                        break;
                case 'f':
                        (void) prom_setprop(prom_optionsnode(), "diag-level",
                        (void)prom_setprop(prom_optionsnode(), "diag-level",
                            (char *)params.gos_optargp,
                            params.gos_optarglen + 1);
                        break;
#endif
                case 'g':
                        boothowto |= RB_FORTHDEBUG;
                        break;
                case 'G':
                        boothowto |= RB_FORTHDEBUGDBP;
                        break;
                case 'h':
                        boothowto |= RB_HALT;
                        break;
#if defined(_OBP)
                case 'H':
                        break;
```

```
#endif
                case 'i':
                        if (params.gos_optarglen + 1 > sizeof (initname)) {
                                _kobj_printf(ops, "krtld: initname too long.  "
                                    "Ignoring.\n");
                        } else {
                                (void) strncpy(initname, params.gos_optargp,
                                    params.gos_optarglen);
                                initname[params.gos_optarglen] = '\0';
                        }
                        break;
                case 'k':
                        boothowto |= RB_KMDB;
                        break;
                case 'm':
                        if (strlen(initargs) + 3 + params.gos_optarglen + 1 >
                            sizeof (initargs)) {
                                _kobj_printf(ops,
                                    "unix: init options too long.  "
                                    "Ignoring -m.\n");
                                break;
                        }
                        /* gos_optargp is not null terminated */
                        (void) strncpy(scratch, params.gos_optargp,
                            params.gos_optarglen);
                        scratch[params.gos_optarglen] = '\0';
                        (void) strlcat(initargs, "-m ", sizeof (initargs));
                        (void) strlcat(initargs, scratch,
                            sizeof (initargs));
                        (void) strlcat(initargs, " ", sizeof (initargs));
                        break;
#if defined(_OBP)
                /* Ignore argument meant for wanboot standalone */
                case 'o':
                        break;
#endif
                case 'O': {
                        char **str = &kobj_kmdb_argv[num_O_opt];

                        if (++num_O_opt > (sizeof (kobj_kmdb_argv) /
                            sizeof (char *)) - 1) {
                                _kobj_printf(ops, "krtld: too many kmdb "
                                    "options - ignoring option #%d.\n",
                                    num_O_opt);
                                continue;
                        }

                        *str = kobj_alloc(params.gos_optarglen + 1, KM_TMP);
                        (void) strncpy(*str, params.gos_optargp,
                            params.gos_optarglen);
                        (*str)[params.gos_optarglen] = '\0';
                        break;
                }
                case 'r':
                        if (strlen(initargs) + 3 + 1 > sizeof (initargs)) {
                                _kobj_printf(ops, "unix: init options too "
                                    "long.  Ignoring -r.\n");
                                break;
                        }
                        boothowto |= RB_RECONFIG;
                        (void) strlcat(initargs, "-r ", sizeof (initargs));
                        break;
                case 's':
                        if (strlen(initargs) + 3 + 1 > sizeof (initargs)) {
                                _kobj_printf(ops, "unix: init options too "
                                    "long.  Ignoring -s.\n");
```



```
                                break;
                        }
                        boothowto |= RB_SINGLE;
                        (void) strlcat(initargs, "-s ", sizeof (initargs));
                        break;
                case 'v':
                        if (strlen(initargs) + 3 + 1 > sizeof (initargs)) {
                                _kobj_printf(ops, "unix: init options too "
                                    "long.  Ignoring -v.\n");
                                break;
                        }
                        boothowto |= RB_VERBOSE;
                        (void) strlcat(initargs, "-v ", sizeof (initargs));
                        break;
#if defined(_OBP)
                case 'V':
                        break;
                case 'Z':
                        break;
#endif
                case 'w':
                        boothowto |= RB_WRITABLE;
                        break;
                case 'x':
                        boothowto |= RB_NOBOOTCLUSTER;
                        break;
                case '?':
                        switch (params.gos_last_opt) {
                        case 'i':
                                _kobj_printf(ops, "krtld: Required argument "
                                    "for -i flag missing.  Ignoring.\n");
                                break;
                        default:
                                _kobj_printf(ops, "krtld: Ignoring invalid "
                                    "kernel option -%c.\n",
                                    params.gos_last_opt);
                        }
                        break;
                default:
                        _kobj_printf(ops, "krtld: Ignoring unimplemented "
                            "option -%c.\n", c);
                }
        }

        if ((boothowto & (RB_DEBUGENTER | RB_KMDB)) == RB_DEBUGENTER) {
                _kobj_printf(ops, "krtld: -d is not valid without -k.\n");
                boothowto &= ~RB_DEBUGENTER;
        }

        if (*params.gos_strp) {
                /* Unused arguments. */
                if (params.gos_strp[0] == '-' && ISSPACE(params.gos_strp[1])) {
                        /*EMPTY*/
                        /* Lousy install arguments.  Silently ignore. */
                } else {
                        _kobj_printf(ops, "krtld: Unused kernel arguments: "
                            "'%s'.\n", params.gos_strp);
                }
        }
}
```

unchanged_portion_omitted_