

new/usr/src/cmd/initpkg/Makefile

1

```

*****
2854 Thu Jun 6 18:30:48 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

```

new/usr/src/cmd/initpkg/Makefile

2

```

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

```

new/usr/src/cmd/prtconf/prtconf.c

1

8214 Thu Jun 6 18:30:49 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 = "abcdVvXpPff:M:dLuC";
170 #else
171 static const char *optstring = "abcdVvXpPff: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;
```

new/usr/src/cmd/prtconf/prtconf.c

2

```
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.
346  */
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

new/usr/src/pkg/manifests/driver-network-platform.mf

1

```
*****
2754 Thu Jun  6 18:30:50 2013
new/usr/src/pkg/manifests/driver-network-platform.mf
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 #

22 #
23 # Copyright (c) 2010, Oracle and/or its affiliates. All rights reserved.
24 #

26 #
27 # The default for payload-bearing actions in this package is to appear in the
28 # global zone only. See the include file for greater detail, as well as
29 # information about overriding the defaults.
30 #
31 <include global_zone_only_component>
32 set name=pkg.fmri value=pkg:/driver/network/platform@$(PKGVERS)
33 set name=pkg.description value="Platform Support, OS Functionality"
34 set name=pkg.summary value="Platform Support, OS Functionality"
35 set name=info.classification \
36     value=org.opensolaris.category.2008:Drivers/Networking
37 set name=variant.arch value=i386
38 dir path=etc group=sys
39 dir path=kernel group=sys
40 dir path=kernel/drv group=sys
41 dir path=kernel/drv/$(ARCH64) group=sys
42 dir path=usr/share/man
43 dir path=usr/share/man/man7d
44 driver name=dnet clone_perms="dnet 0666 root sys" perms="* 0666 root sys" \
45     alias=pci1011,14 \
46     alias=pci1011,19 \
47     alias=pci1011,2 \
48     alias=pci1011,9 \
49     alias=pci10b8,2001 \
50     alias=pci1109,1400 \
51     alias=pci1109,2400 \
52     alias=pci2646,1
53 file path=etc/bootrc group=sys mode=0755
54 file path=etc/mach group=sys original_name=SUNWos86r:etc/mach preserve=true
55 file path=kernel/drv/$(ARCH64)/dnet group=sys
56 file path=kernel/drv/dnet group=sys
57 file path=kernel/drv/sd group=sys
58     original_name=SUNWos86r:kernel/drv/sd.conf preserve=true
59 file path=usr/share/man/man7d/dnet.7d
60 legacy pkg=SUNWos86r desc="Platform Support, OS Functionality (Root)" \
```

new/usr/src/pkg/manifests/driver-network-platform.mf

2

```
61     name="Platform Support, OS Functionality (Root)"
62 license cr_Sun license=cr_Sun
63 license lic_CDDL license=lic_CDDL
64 # elx1 moved out of this package, so create a dependency for upgraded systems
65 depend fmri=driver/network/elx1 type=require
66 # iprb moved out of this package, so create a dependency for upgraded systems
67 depend fmri=driver/network/iprb type=require
68 # pcn moved out of this package, so create a dependency for upgraded systems
69 depend fmri=driver/network/pcn type=require
```

new/usr/src/uts/common/conf/param.c

1

```
*****
21322 Thu Jun  6 18:30:51 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 /*
```

new/usr/src/uts/common/conf/param.c

2

```
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  */
541 char architecture[] = "i386";
542 char architecture_32[] = "i386";
543 char hw_provider[SYS_NMLN] = "";

544 #elif defined(__amd64)

546 /*
547  * On amd64 machines, read hw_serial, hw_provider and srpc_domain from
548  * /etc/bootrc at boot time.
549  */
550 char architecture[] = "amd64";
551 char architecture_32[] = "i386";
552 char hw_provider[SYS_NMLN] = "";

553 #else
554 #error "unknown processor architecture"
555 #endif

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

558 /* Initialize isa_list */
559 char *isa_list = architecture;

560 static pgcnt_t original_physmem = 0;

561 #define MIN_DEFAULT_MAXUSERS      8u
562 #define MAX_DEFAULT_MAXUSERS     2048u
563 #define MAX_MAXUSERS              4096u

564 void
565 param_preset(void)
566 {
567     original_physmem = physmem;
568 }
_____unchanged_portion_omitted_____
```

new/usr/src/uts/common/krtld/kobj_bootflags.c

1

```
*****
6328 Thu Jun  6 18:30:52 2013
new/usr/src/uts/common/krtld/kobj_bootflags.c
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 2010 Sun Microsystems, Inc. All rights reserved.
23 * Use is subject to license terms.
24 */

27 #include <sys/types.h>
28 #include <sys/reboot.h>
29 #include <sys/cmn_err.h>
30 #include <sys/bootconf.h>
31 #include <sys/promif.h>
32 #include <sys/obpdefs.h>
33 #include <sys/sunddi.h>
34 #include <sys/system.h>
35 #include <sys/kobj.h>
36 #include <sys/kobj_impl.h>
37 #include <util/getoptstr.h>

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

41 /*
42  * Parse the boot line to determine boot flags.
43  */
44 void
45 bootflags(struct bootops *ops)
46 {
47     struct gos_params params;
48     uchar_t num_0_opt = 0;
49     char *cp;
50     int c;
51     char scratch[BOOTARGS_MAX];

53     if (BOP_GETPROP(ops, "bootargs", kern_bootargs) == -1) {
54         boothowto |= RB_ASKNAME;
55         return;
56     }

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

60     cp = kern_bootargs;
```

new/usr/src/uts/common/krtld/kobj_bootflags.c

2

```
62 #if defined(_OBP)
63     /*
64      * x86: The boot scripts don't prepend the kernel name to the boot
65      * arguments. (And beware making it do so: if the run-kernel command
66      * returns, it will loop, and you will end up with multiple copies
67      * of the kernel name.)
68      * x86: The boot scripts (i.e., /etc/bootrc) don't prepend the kernel
69      * name to the boot arguments. (And beware making it do so: if the
70      * run-kernel command returns, it will loop, and you will end up with
71      * multiple copies of the kernel name.)
72      */
69     if (cp[0] != '-') {
70         /* if user booted kadb or kmdb, load kmdb */
71         if (cp[0] == 'k' && (cp[1] == 'a' || cp[1] == 'm') &&
72             cp[2] == 'd' && cp[3] == 'b' &&
73             (cp[4] == ' ' || cp[4] == ' ' || cp[4] == 0))
74             boothowto |= RB_KMDB;
75         SKIP_WORD(cp);      /* Skip the kernel's filename. */
76     }
77 #endif
78     SKIP_SPC(cp);

80 #if defined(_OBP)
81     /* skip bootblk args */
82     params.gos_opts = "abcdDf:F:GHHi:km:o:0:rsvVwxZ:";
83 #else
84     params.gos_opts = "abcdgGhi:km:0:rsvwX";
85 #endif
86     params.gos_strp = cp;
87     getoptstr_init(&params);
88     while ((c = getoptstr(&params)) != -1) {

90         switch (c) {
91             case 'a':
92                 boothowto |= RB_ASKNAME;
93                 break;
94             case 'b':
95                 boothowto |= RB_NOBOOTRC;
96                 break;
97             case 'c':
98                 boothowto |= RB_CONFIG;
99                 break;
100             case 'd':
101                 boothowto |= RB_DEBUGENTER;
102                 break;
103 #if defined(_OBP)
104             case 'D':
105             case 'F':
106                 break;
107             case 'f':
108                 (void) prom_setprop(prom_optionsnode(), "diag-level",
109                                     (void) prom_setprop(prom_optionsnode(), "diag-level",
110                                     (char *)params.gos_optargp,
111                                     params.gos_optarglen + 1);
112                 break;
113 #endif
114             case 'g':
115                 boothowto |= RB_FORTHDEBUG;
116                 break;
117             case 'G':
118                 boothowto |= RB_FORTHDEBUGDBP;
119                 break;
120             case 'h':
121                 boothowto |= RB_HALT;
122                 break;
122 #if defined(_OBP)
```

```

123         case 'H':
124             break;
125     #endif
126     case 'i':
127         if (params.gos_optarglen + 1 > sizeof (initname)) {
128             _kobj_printf(ops, "krtld: initname too long. "
129                 "Ignoring.\n");
130         } else {
131             (void) strncpy(initname, params.gos_optargp,
132                 params.gos_optarglen);
133             initname[params.gos_optarglen] = '\0';
134         }
135         break;
136     case 'k':
137         boothowto |= RB_KMDB;
138         break;
139     case 'm':
140         if (strlen(initargs) + 3 + params.gos_optarglen + 1 >
141             sizeof (initargs)) {
142             _kobj_printf(ops,
143                 "unix: init options too long. "
144                 "Ignoring -m.\n");
145             break;
146         }
147         /* gos_optargp is not null terminated */
148         (void) strncpy(scratch, params.gos_optargp,
149             params.gos_optarglen);
150         scratch[params.gos_optarglen] = '\0';
151         (void) strlcat(initargs, "-m ", sizeof (initargs));
152         (void) strlcat(initargs, scratch,
153             sizeof (initargs));
154         (void) strlcat(initargs, " ", sizeof (initargs));
155         break;
156     #if defined(_OBP)
157         /* Ignore argument meant for wanboot standalone */
158     case 'o':
159         break;
160     #endif
161     case 'O': {
162         char **str = &kobj_kmdb_argv[num_0_opt];
163
164         if (++num_0_opt > (sizeof (kobj_kmdb_argv) /
165             sizeof (char *)) - 1) {
166             _kobj_printf(ops, "krtld: too many kmdb "
167                 "options - ignoring option #%.d.\n",
168                 num_0_opt);
169             continue;
170         }
171
172         *str = kobj_alloc(params.gos_optarglen + 1, KM_TMP);
173         (void) strncpy(*str, params.gos_optargp,
174             params.gos_optarglen);
175         (*str)[params.gos_optarglen] = '\0';
176         break;
177     }
178     case 'r':
179         if (strlen(initargs) + 3 + 1 > sizeof (initargs)) {
180             _kobj_printf(ops, "unix: init options too "
181                 "long. Ignoring -r.\n");
182             break;
183         }
184         boothowto |= RB_RECONFIG;
185         (void) strlcat(initargs, "-r ", sizeof (initargs));
186         break;
187     case 's':
188         if (strlen(initargs) + 3 + 1 > sizeof (initargs)) {

```

```

189         _kobj_printf(ops, "unix: init options too "
190             "long. Ignoring -s.\n");
191         break;
192     }
193     boothowto |= RB_SINGLE;
194     (void) strlcat(initargs, "-s ", sizeof (initargs));
195     break;
196     case 'v':
197         if (strlen(initargs) + 3 + 1 > sizeof (initargs)) {
198             _kobj_printf(ops, "unix: init options too "
199                 "long. Ignoring -v.\n");
200             break;
201         }
202         boothowto |= RB_VERBOSE;
203         (void) strlcat(initargs, "-v ", sizeof (initargs));
204         break;
205     #if defined(_OBP)
206     case 'V':
207         break;
208     case 'Z':
209         break;
210     #endif
211     case 'w':
212         boothowto |= RB_WRITABLE;
213         break;
214     case 'x':
215         boothowto |= RB_NOBOOTCLUSTER;
216         break;
217     case '?':
218         switch (params.gos_last_opt) {
219             case 'i':
220                 _kobj_printf(ops, "krtld: Required argument "
221                     "for -i flag missing. Ignoring.\n");
222                 break;
223             default:
224                 _kobj_printf(ops, "krtld: Ignoring invalid "
225                     "kernel option -%.c.\n",
226                     params.gos_last_opt);
227         }
228         break;
229     default:
230         _kobj_printf(ops, "krtld: Ignoring unimplemented "
231             "option -%.c.\n", c);
232     }
233 }
234
235 if ((boothowto & (RB_DEBUGENTER | RB_KMDB)) == RB_DEBUGENTER) {
236     _kobj_printf(ops, "krtld: -d is not valid without -k.\n");
237     boothowto &= ~RB_DEBUGENTER;
238 }
239
240 if (*params.gos_strp) {
241     /* Unused arguments. */
242     if (params.gos_strp[0] == '-' && ISSPACE(params.gos_strp[1])) {
243         /* EMPTY */
244         /* Lousy install arguments. Silently ignore. */
245     } else {
246         _kobj_printf(ops, "krtld: Unused kernel arguments: "
247             "%s.\n", params.gos_strp);
248     }
249 }
250 }

```

unchanged_portion_omitted