

```
*****
605 Wed May 20 11:30:07 2015
new/usr/src/cmd/make/Makefile.com
make: unifdef for SUNOS4_AND_AFTER (defined)
*****
1 #
2 # This file and its contents are supplied under the terms of the
3 # Common Development and Distribution License (" CDDL"), version 1.0.
4 # You may only use this file in accordance with the terms of version
5 # 1.0 of the CDDL.
6 #
7 # A full copy of the text of the CDDL should have accompanied this
8 # source. A copy of the CDDL is also available via the Internet at
9 # http://www.illumos.org/license/CDDL.
10 #

11 # Copyright 2015, Richard Lowe.

14 MAKE_INCLUDE= $(SRC)/cmd/make/include
15 MAKE_DEFS= -DSYSV -DINTER
15 MAKE_DEFS= -DSYSV -DINTER -DSUNOS4_AND_AFTER
16 $(RELEASE_BUILD)MAKE_DEFS += -DNDEBUG
17 CFLAGS += $(CCVERBOSE)
18 CPPFLAGS += -I$(MAKE_INCLUDE) $(MAKE_DEFS)
```

new/usr/src/cmd/make/bin/depvar.cc

```
*****
3177 Wed May 20 11:30:08 2015
new/usr/src/cmd/make/bin/depvar.cc
make: unifdef for SUNOS4_AND_AFTER (defined)
*****
_____unchanged_portion_omitted_____
46 typedef struct _Depvar *Depvar;
48 static Depvar depvar_list;
49 static Depvar *bpatch = &depvar_list;
50 static Boolean variant_deps;
52 /*
53  * Add a name to the list.
54 */
56 void
57 depvar_add_to_list(Name name, Boolean cmdline)
58 {
59     Depvar dv;
61 #ifdef SUNOS4_AND_AFTER
61     dv = ALLOC(Depvar);
63 #else
64     dv = (Depvar) Malloc(sizeof(struct _Depvar));
65 #endif
62     dv->name = name;
63     dv->next = NULL;
64     dv->cmdline = cmdline;
65     *bpatch = dv;
66     bpatch = &dv->next;
67 }
69 /*
70  * The macro 'name' has been used in either the left-hand or
71  * right-hand side of a dependency. See if it is in the
72  * list. Two things are looked for. Names given as args
73  * to the -V list are checked so as to set the same/differ
74  * output for the -P option. Names given as macro=value
75  * command-line args are checked and, if found, an NSE
76  * warning is produced.
77 */
78 void
79 depvar_dep_macro_used(Name name)
80 {
81     Depvar dv;
83     for (dv = depvar_list; dv != NULL; dv = dv->next) {
84         if (name == dv->name) {
85 #ifdef NSE
86 #ifdef SUNOS4_AND_AFTER
86         if (dv->cmdline) {
87 #else
88             if (is_true(dv->cmdline)) {
89 #endif
90                 nse_dep_ccmdmacro(dv->name->string);
91             }
92             variant_deps = true;
93         }
94     }
96 #ifdef NSE
```

1

new/usr/src/cmd/make/bin/depvar.cc

```
97 /*
98  * The macro 'name' has been used in either the argument
99  * to a cd before a recursive make. See if it was
100 * defined on the command-line and, if so, complain.
101 */
102 void
103 depvar_rule_macro_used(Name name)
104 {
105     Depvar dv;
107     for (dv = depvar_list; dv != NULL; dv = dv->next) {
108         if (name == dv->name) {
117 #ifdef SUNOS4_AND_AFTER
119         if (dv->cmdline) {
120             if (is_true(dv->cmdline)) {
121 #endif
110                 nse_rule_ccmdmacro(dv->name->string);
112             }
113         }
114     }
115 }
_____unchanged_portion_omitted_____
2
```

new/usr/src/cmd/make/bin/nse.cc

```
*****
12541 Wed May 20 11:30:08 2015
new/usr/src/cmd/make/bin/nse.cc
make: unifdef for SUNOS4_AND_AFTER (defined)
*****
unchanged_portion_omitted_
65 static Nse_suffix      sufx_hdr;
66 static int             our_exit_status;

68 static void            nse_warning(void);
69 static Boolean          nse_gettoken(wchar_t **, wchar_t *);

71 /*
72 * Given a command that has just recursed to a sub make
73 * try to determine if it cd'ed to a directory that was
74 * defined by a make variable imported from the shell
75 * environment or a variable with backquotes in it.
76 * This routine will find something like:
77 *   cd $(DIR); $(MAKE)
78 * where DIR is imported from the shell environment.
79 * However it well not find:
80 *   CD = cd
81 *   $(CD) $(DIR); $(MAKE)
82 * or
83 *   CD = cd $(DIR)
84 *   $(CD); $(MAKE)
85 *
86 * This routine also checks for recursion to the same
87 * directory.
88 */
89 void
90 nse_check_cd(Property prop)
91 {
92     wchar_t          tok[512];
93     wchar_t          *p;
94     wchar_t          *our_template;
95     int              len;
96     Boolean          cd;
97 #ifdef SUNOS4_AND_AFTER
97     String_rec       string;
99 #else
100    String          string;
101 #endif
98     Name             name;
99     Name             target;
100    struct Line      *line;
101    struct Recursive *r;
102    Property         recurse;
103    wchar_t          strbuf[STRING_BUFFER_LENGTH];
104    wchar_t          tmpbuf[STRING_BUFFER_LENGTH];

106 #ifdef LTEST
107     printf("In nse_check_cd, nse = %d, nse_did_recursion = %d\n", nse, nse_d
108 #endif
113 #ifdef SUNOS4_AND_AFTER
109     if (!nse_did_recursion || !nse) {
115 #else
116     if (is_false(nse_did_recursion) || is_false(flag.nse)) {
117 #endif
110 #ifdef LTEST
111         printf ("returning, nse = %d, nse_did_recursion = %d\n", nse,
112 #endif
113         return;
114     }
115     line = &prop->body.line;
116 #ifdef LTEST
```

1

new/usr/src/cmd/make/bin/nse.cc

```
117     printf("string = %s\n", line->command_template->command_line->string_mb)
118 #endif
120     wscpy(tmpbuf, line->command_template->command_line->string);
121     our_template = tmpbuf;
122     cd = false;
123     while (nse_gettoken(&our_template, tok)) {
124 #ifdef LTEST
125         printf("in gettoken loop\n");
126 #endif
127 #ifdef SUNOS4_AND_AFTER
128         if (IS_WEQUAL(tok, (wchar_t *) "cd")) {
129 #else
130             if (is_equal(tok, "cd")) {
131 #endif
132                 cd = true;
133             } else if (cd && tok[0] == '$') {
134                 nse_backquote_seen = NULL;
135                 nse_shell_var_used = NULL;
136                 nse_watch_vars = true;
137 #ifdef SUNOS4_AND_AFTER
138                 INIT_STRING_FROM_STACK(string, strbuf);
139                 name = GETNAME(tok, FIND_LENGTH);
140 #else
141                 init_string_from_stack(string, strbuf);
142                 name = getname(tok, FIND_LENGTH);
143 #endif
144                 expand_value(name, &string, false);
145                 nse_watch_vars = false;
146 #endif
147 #ifdef LTEST
148         printf("cd = %d, tok = $%n", cd);
149 #endif
150 #endif
151 #endif
152 /* Try to trim tok to just
153 * the variable.
154 */
155 if (nse_shell_var_used != NULL) {
156     nse_warning();
157     fprintf(stderr, "\tMake invoked recursively by c
158         nse_shell_var_used->string_mb,
159         line->command_template->command_line->string
160     }
161 if (nse_backquote_seen != NULL) {
162     nse_warning();
163     fprintf(stderr, "\tMake invoked recursively by c
164         nse_backquote_seen->string_mb,
165         line->command_template->command_line->string
166     }
167     cd = false;
168 } else if (cd && nse_backquotes(tok)) {
169     nse_warning();
170     fprintf(stderr, "\tMake invoked recursively by cd'ing to
171         line->command_template->command_line->string_mb);
172 } else {
173     cd = false;
174     cd = false;
175 }
176 */
177 /* Now check for recursion to "..".
178 */
179 if (primary_makefile != NULL) {
180     target = prop->body.line.target;
181     recurse = get_prop(target->prop, recursive_prop);
```

2

```

new/usr/src/cmd/make/bin/nse.cc

174         while (recurse != NULL) {
175             r = &recurse->body.recursive;
176 #ifdef SUNOS4_AND_AFTER
177             if (IS_WEQUAL(r->directory->string, (wchar_t *) ".") &&
178                 !IS_WEQUAL(r->makefiles->name->string,
179                 primary_makefile->string)) {
180 #else
181             if (is_equal(r->directory->string, ".") &&
182                 !is_equal(r->makefiles->name->string,
183                 primary_makefile->string)) {
184                 nse_warning();
185                 fprintf(stderr, "\tRecursion to makefile '%s' in
186                         r->makefiles->name->string_mb,
187                         line->command_template->command_line->string
188             }
189             recurse = get_prop(recurse->next, recursive_prop);
190         }
191     }
192 }
193 }

194 /* Print an NSE make warning line.
195 * If the -P flag was given then consider this a fatal
196 * error, otherwise, just a warning.
197 */
198 static void
199 nse_warning(void)
200 {
201 #ifdef SUNOS4_AND_AFTER
202     if (report_dependencies_level > 0) {
203 #else
204     if (is_true(flag.report_dependencies)) {
205 #endif
206         our_exit_status = 1;
207     }
208     if (primary_makefile != NULL) {
209         fprintf(stderr, "make: NSE warning from makefile %s/%s:\n",
210                 get_current_path(), primary_makefile->string_mb);
211     } else {
212         fprintf(stderr, "make: NSE warning from directory %s:\n",
213                 get_current_path());
214     }
215 }
216

217 unchanged_portion_omitted

218 /*
219 * Given a dependency and a target, see if the dependency
220 * is an SCCS file. Check for the last component of its name
221 * beginning with "s." and the component before that being "SCCS".
222 * The NSE does not consider a source file to be derived from
223 * an SCCS file.
224 */
225 void
226 nse_check_sccs(wchar_t *targ, wchar_t *dep)
227 {
228     wchar_t      *slash;
229     wchar_t      *p;

230 #ifdef SUNOS4_AND_AFTER
231     if (!nse) {
232 #else
233     if (is_false(flag.nse)) {
234 #endif
235         return;
236     }

```

```
new/usr/src/cmd/make/bin/nse.cc

285 #ifdef SUNOS4_AND_AFTER
286     slash = wsrchr(dep, (int) slash_char);
287 #else
288     slash = rindex(dep, '/');
289 #endif
290     if (slash == NULL) {
291         return;
292     }
293     if (slash[1] != 's' || slash[2] != '.') {
294         return;
295     }

296     /*
297      * Find the next to last filename component.
298      */
299     for (p = slash - 1; p >= dep; p--) {
300         if (*p == '/') {
301             break;
302         }
303     }
304     p++;
305 #ifdef SUNOS4_AND_AFTER
306     MBSTOWCS(wcs_buffer, "SCCS/");
307     if (IS_WEQUALN(p, wcs_buffer, wslen(wcs_buffer))) {
308 #else
309     if (is_equaln(p, "SCCS/", 5)) {
310 #endif
311         nse_warning();
312         WCSTOMB(mbs_buffer, targ);
313         WCSTOMB(mbs_buffer2, dep);
314         fprintf(stderr, "\tFile '%s' depends upon SCCS file '%s'\n",
315                 mbs_buffer, mbs_buffer2);
316     }
317     return;
318 }
319

320 /* Given a filename check to see if it has 2 backquotes in it.
321  * Complain about this because the shell expands the backquotes
322  * but make does not so the files always appear to be out of date.
323  */
324 void
325 nse_check_file_backquotes(wchar_t *file)
326 {
327 #ifdef SUNOS4_AND_AFTER
328     if (!nse) {
329 #else
330     if (is_false(flag.nse)) {
331 #endif
332         return;
333     }
334     if (nse_backquotes(file)) {
335         nse_warning();
336         WCSTOMB(mbs_buffer, file);
337         fprintf(stderr, "\tfilename \"%s\" has backquotes in it\n",
338                 mbs_buffer);
339     }
340 }

341 /*
342  * Return true if the string has two backquotes in it.
343  */
344 Boolean
345 nse_backquotes(wchar_t *str)
346 {
347     wchar_t          *bq;
```

```

352 #ifdef SUNOS4_AND_AFTER
309     bq = wschr(str, (int) backquote_char);
310     if (bq) {
311         bq = wschr(&bq[1], (int) backquote_char);
312     }
313     if (bq) {
314         return true;
315     }
316     return false;
317 }

319 /*
320 * A macro that was defined on the command-line was found to affect the
321 * set of dependencies. The NSE "target explode" will not know about
322 * this and will not get the same set of dependencies.
323 */
324 void
325 nse_dep_cmdmacro(wchar_t *macro)
326 {
376 #ifdef SUNOS4_AND_AFTER
327     if (!nse) {
378 #else
379     if (is_false(flag.nse)) {
380 #endif
381         return;
382     }
383     nse_warning();
384     WCSTOMBS(mbs_buffer, macro);
385     fprintf(stderr, "\tVariable '%s' is defined on the command-line and\n\t"
386             "mbs_buffer);
387 }

388 /*
389 * A macro that was defined on the command-line was found to
390 * be part of the argument to a cd before a recursive make.
391 * This make cause the make to recurse to different places
392 * depending upon how it is invoked.
393 */
394 void
395 nse_rule_cmdmacro(wchar_t *macro)
396 {
397 #ifdef SUNOS4_AND_AFTER
398     if (!nse) {
400 #else
401     if (is_false(flag.nse)) {
402 #endif
403         return;
404     }
405     nse_warning();
406     WCSTOMBS(mbs_buffer, macro);
407     fprintf(stderr, "\tMake invoked recursively by cd'ing to a directory\n\t"
408             "mbs_buffer);
409 }

410 /*
411 * A dependency has been found with a wildcard in it.
412 * This causes the NSE problems because the set of dependencies
413 * can change without changing the Makefile.
414 */
415 void

```

```

360 nse_wildcard(wchar_t *targ, wchar_t *dep)
361 {
362     #ifdef SUNOS4_AND_AFTER
363         if (!nse) {
364             if (is_false(flag.nse)) {
365                 return;
366             }
367             nse_warning();
368             WCSTOMBS(mbs_buffer, targ);
369             WCSTOMBS(mbs_buffer2, dep);
370             fprintf(stderr, "\tFile '%s' has a wildcard in dependency '%s'\n",
371                     mbs_buffer, mbs_buffer2);
372     }

373     /* Read in the list of suffixes that are interpreted as source
374      * files.
375     */
376 void
377 nse_init_source_suffixes(void)
378 {
379     FILE             *fp;
380     wchar_t          suffix[100];
381     Nse_suffix       sufxf;
382     Nse_suffix       *bpatch;

383     fp = fopen(TARG_SUFFIX, "r");
384     if (fp == NULL) {
385         return;
386     }
387     bpatch = &sufxf_hdr;
388     while (fscanf(fp, "%s %*s", suffix) == 1) {
451 #ifdef SUNOS4_AND_AFTER
390         sufxf = ALLOC(Nse_suffix);
391         sufxf->suffix = wscopy(ALLOC_WC(wslen(suffix) + 1), suffix);
454 #else
392         sufxf = alloc(Nse_suffix);
393         sufxf->suffix = strcpy(malloc(strlen(suffix) + 1), suffix);
456 #endif
394         sufxf->next = NULL;
395         *bpatch = sufxf;
396         bpatch = &sufxf->next;
397     }
398     fclose(fp);

399 /*
400  * Check if a derived file (something with a dependency) appears
401  * to be a source file (by its suffix) but has no rule to build it.
402  * If so, complain.
403  *
404  * This generally arises from the old-style of make-depend that
405  * produces:
406  *     foo.c: foo.h
407  */
408 void
409 nse_check_derived_src(Name target, wchar_t *dep, Cmd_line command_template)
410 {
411     Nse_suffix       sufxf;
412     wchar_t          *suffix;
413     wchar_t          *depsufxf;

481 #ifdef SUNOS4_AND_AFTER
415     if (!nse) {

```

```

483 #else
484     if (is_false(flag.nse)) {
485 #endif
486         return;
487     }
488 #ifdef SUNOS4_AND_AFTER
489     if (target->stat.is_derived_src) {
490 #else
491     if (is_true(target->stat.is_derived_src)) {
492 #endif
493         return;
494     }
495     if (command_template != NULL) {
496         return;
497     }
498 #ifdef SUNOS4_AND_AFTER
499     suffix = wsrchr(target->string, (int) period_char );
500 #else
501     suffix = rindex(target->string, '.');
502 #endif
503     if (suffix != NULL) {
504         for (sufx = sufz_hdr; sufz != NULL; sufz = sufz->next) {
505 #ifdef SUNOS4_AND_AFTER
506             if (IS_WEQUAL(sufz->suffix, suffix)) {
507 #else
508             if (is_equal(sufz->suffix, suffix)) {
509 #endif
510                 nse_warning();
511                 WCSTOMB(mbs_buffer, dep);
512                 fprintf(stderr, "\tProbable source file '%s' app
513                         target->string_mb, mbs_buffer);
514                 break;
515             }
516         }
517     }
518 /*
519  * See if a target is a potential source file and has no
520  * dependencies and no rule but shows up on the right-hand
521  * side. This tends to occur from old "make depend" output.
522 */
523 void
524 nse_check_no_deps_no_rule(Name target, Property line, Property command)
525 {
526     Nse_suffix      sufz;
527     wchar_t          *suffix;
528
529 #ifdef SUNOS4_AND_AFTER
530     if (!nse) {
531 #else
532     if (is_false(flag.nse)) {
533 #endif
534         return;
535     }
536 #ifdef SUNOS4_AND_AFTER
537     if (target->stat.is_derived_src) {
538 #else
539     if (is_true(target->stat.is_derived_src)) {
540 #endif
541         return;
542     }
543     if (line != NULL && line->body.line.dependencies != NULL) {
544         return;
545     }
546 #ifdef SUNOS4_AND_AFTER

```

```

458         if (command->body.line.sccs_command) {
459 #else
460         if (is_true(command->body.line.sccs_command)) {
461 #endif
462             return;
463         }
464 #ifdef SUNOS4_AND_AFTER
465         suffix = wsrchr(target->string, (int) period_char );
466 #else
467         suffix = rindex(target->string, '.');
468 #endif
469         if (suffix != NULL) {
470             for (sufz = sufz_hdr; sufz != NULL; sufz = sufz->next) {
471 #ifdef SUNOS4_AND_AFTER
472                 if (IS_WEQUAL(sufz->suffix, suffix)) {
473 #else
474                 if (is_equal(sufz->suffix, suffix)) {
475 #endif
476                     if (command->body.line.command_template == NULL)
477                         nse_warning();
478                     fprintf(stderr, "\tProbable source file
479                         target->string_mb);
480                 }
481             }
482         }
483     }
484 #endif
485     if (!nse) {
486 #else
487     if (is_false(flag.nse)) {
488 #endif
489         return;
490     }
491     nse_warning();
492     fprintf(stderr, "Recursive make to derive %s did not use a makefile\n",
493             target->string_mb);
494 }
495
496 unchanged_portion_omitted

```

new/usr/src/cmd/make/bin/nse_printdep.cc

```
*****
8831 Wed May 20 11:30:08 2015
new/usr/src/cmd/make/bin/nse_printdep.cc
make: unifdef for SUNOS4_AND_AFTER (defined)
*****
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 2003 Sun Microsystems, Inc. All rights reserved.
23 * Use is subject to license terms.
24 */

25 /*
26 * Included files
27 */
28 /*
29 #include <mk/defs.h>
30 #include <mksh/misc.h> /* get_prop() */

31 /*
32 * File table of contents
33 */
34 static void print_dependencies(register Name target, register Property line);
35 void print_deps(register Name target, register Property line);
36 static void print_deps(register Name target, register Property line);
37 static void print_more_deps(Name target, Name name);
38 static void print_filename(Name name);
39 static Boolean should_print_dep(Property line);
40 static void print_forest(Name target);
41 static void print_deplist(Dependency head);
42 void print_value(register Name value, Daemon daemon);
43 static void print_rule(register Name target);
44 static void print_rec_info(Name target);
45 static Boolean is_out_of_date(Property line);
46 extern void depvar_print_results (void);
47 extern int printf (const char *, ...);
48 extern int _flsbuf (unsigned int, FILE *);

49 /*
50 * print_dependencies(target, line)
51 *
52 * Print all the dependencies of a target. First print all the Makefiles.
53 * Then print all the dependencies. Finally, print all the .INIT
54 * dependencies.
55 *
56 * Parameters:
57 *      target      The target we print dependencies for
58 *      line       We get the dependency list from here
59 *
60 * Global variables used:
61 *
```

1

new/usr/src/cmd/make/bin/nse_printdep.cc

```
62 *          done           The Name ".DONE"
63 *          init           The Name ".INIT"
64 *          makefiles_used List of all makefiles read
65 */
66 void
67 print_dependencies(register Name target, register Property line)
68 {
69     Dependency dp;
70     static Boolean makefiles_printed = false;
71
72 #ifdef SUNOS4_AND_AFTER
73     if (target_variants) {
74 #else
75     if (is_true(flag.target_variants)) {
76 #endif
77         depvar_print_results();
78     }
79
80     if (!makefiles_printed) {
81         /*
82          * Search the makefile list for the primary makefile,
83          * then print it and its inclusions. After that go back
84          * and print the default.mk file and its inclusions.
85          */
86         for (dp = makefiles_used; dp != NULL; dp = dp->next) {
87             if (dp->name == primary_makefile) {
88                 break;
89             }
90         }
91         if (dp) {
92             print_deplist(dp);
93             for (dp = makefiles_used; dp != NULL; dp = dp->next) {
94                 if (dp->name == primary_makefile) {
95                     break;
96                 }
97                 (void)printf(" %s", dp->name->string_mb);
98             }
99             (void)printf("\n");
100            makefiles_printed = true;
101        }
102        print_deps(target, line);
103    }
104 #ifdef SUNOS4_AND_AFTER
105    print_more_deps(target, init);
106    print_more_deps(target, done);
107 */
108 #else
109     if (target_variants) {
110         print_more_deps(target, cached_names.init);
111         print_more_deps(target, cached_names.done);
112         if (is_true(flag.target_variants)) {
113 #endif
114             print_forest(target);
115         }
116     }
117
118     unchanged_portion_omitted_
119
120 /*
121 *      print_deps(target, line, go_recursive)
122 *
123 *      Print a regular dependency list. Append to this information which
124 *      indicates whether or not the target is recursive.
125 *
126 *      Parameters:
127 *              target      target to print dependencies for
128 *
```

2

```

149 *           line      We get the dependency list from here
150 *           go_recursive Should we show all dependencies recursively?
151 *
152 *     Global variables used:
153 *           recursive_name The Name ".RECURSIVE", printed
154 */
155 static void
156 print_deps(register Name target, register Property line)
157 {
158     register Dependency    dep;
159
160 #ifdef SUNOS4_AND_AFTER
161     if ((target->dependency_printed) ||
162         (target == force)) {
163 #else
164     if (is_true(target->dependency_printed)) {
165 #endif
166         return;
167     }
168     target->dependency_printed = true;
169
170     /* only print entries that are actually derived and are not leaf
171      * files and are not the result of sccs get.
172      */
173     if (should_print_dep(line)) {
174 #ifdef NSE
175         nse_check_no_deps_no_rule(target, line, line);
176 #endif
177         if ((report_dependencies_level == 2) ||
178             (report_dependencies_level == 4)) {
179             if (is_out_of_date(line)) {
180                 (void) printf("1 ");
181             } else {
182                 (void) printf("0 ");
183             }
184             print_filename(target);
185             (void) printf(":t");
186             print_deplist(line->body.line.dependencies);
187             print_rec_info(target);
188             (void) printf("\n");
189             for (dep = line->body.line.dependencies;
190                  dep != NULL;
191                  dep = dep->next) {
192                 print_deps(dep->name,
193                           get_prop(dep->name->prop, line_prop));
194             }
195         }
196     }
197     unchanged_portion_omitted
198
199     /* should_print_dep(line)
200
201     Test if we should print the dependencies of this target.
202     The line must exist and either have children dependencies
203     or have a command that is not an SCCS command.
204
205     Return value:
206             true if the dependencies should be printed
207
208     Parameters:
209             line      We get the dependency list from here
210
211     Global variables used:
212 */

```

```

269 static Boolean
270 should_print_dep(Property line)
271 {
272     if (line == NULL) {
273         return false;
274     }
275     if (line->body.line.dependencies != NULL) {
276         return true;
277     }
278 #ifdef SUNOS4_AND_AFTER
279     if (line->body.line.sccs_command) {
280 #else
281     if (is_true(line->body.line.sccs_command)) {
282 #endif
283         return false;
284     }
285     /* Print out the root nodes of all the dependency trees
286      * in this makefile.
287      */
288     static void
289     print_forest(Name target)
290     {
291         Name_set::iterator np, e;
292         Property        line;
293
294         for (np = hashtab.begin(), e = hashtab.end(); np != e; np++) {
295 #ifdef SUNOS4_AND_AFTER
296             if (np->is_target && !np->has_parent && np != target) {
297 #else
298             if (is_true(np->is_target) &&
299                 is_false(np->has_parent) &&
300                 np != target) {
301 #endif
302                 (void) doname_check(np, true, false, false);
303                 line = get_prop(np->prop, line_prop);
304                 printf("-\n");
305                 print_deps(np, line);
306             }
307         }
308     }
309
310 #ifndef SUNOS4_AND_AFTER
311     printdesc()
312     {
313         Name_set::iterator p, e;
314         register Property    prop;
315         register Dependency   dep;
316         register Cmd_line     rule;
317         Percent            percent, percent_depe;
318
319         /* Default target */
320         if (default_target_to_build != NULL) {
321             print_rule(default_target_to_build);
322             default_target_to_build->dependency_printed= true;
323         };
324         (void)printf("\n");
325
326         /* .AR_REPLACE */
327         if (ar_replace_rule != NULL) {
328             (void)printf("%s:\n", cached_names.ar_replace->string_mb);
329             for (rule= ar_replace_rule; rule != NULL; rule= rule->next)
330                 (void)printf("\t%s\n", rule->command_line->string_mb);
331         }
332     }
333
334     /* AR_REPLACE */
335     if (ar_replace_rule != NULL) {
336         (void)printf("%s:\n", cached_names.ar_replace->string_mb);
337         for (rule= ar_replace_rule; rule != NULL; rule= rule->next)
338             (void)printf("\t%s\n", rule->command_line->string_mb);
339     }
340
341     /* AR_REPLACE */
342     if (ar_replace_rule != NULL) {
343         (void)printf("%s:\n", cached_names.ar_replace->string_mb);
344         for (rule= ar_replace_rule; rule != NULL; rule= rule->next)
345             (void)printf("\t%s\n", rule->command_line->string_mb);
346     }
347
348     /* AR_REPLACE */
349     if (ar_replace_rule != NULL) {
350         (void)printf("%s:\n", cached_names.ar_replace->string_mb);
351         for (rule= ar_replace_rule; rule != NULL; rule= rule->next)
352             (void)printf("\t%s\n", rule->command_line->string_mb);
353     }
354
355     /* AR_REPLACE */
356     if (ar_replace_rule != NULL) {
357         (void)printf("%s:\n", cached_names.ar_replace->string_mb);
358         for (rule= ar_replace_rule; rule != NULL; rule= rule->next)
359             (void)printf("\t%s\n", rule->command_line->string_mb);
360     }
361
362     /* AR_REPLACE */
363     if (ar_replace_rule != NULL) {
364         (void)printf("%s:\n", cached_names.ar_replace->string_mb);
365         for (rule= ar_replace_rule; rule != NULL; rule= rule->next)
366             (void)printf("\t%s\n", rule->command_line->string_mb);
367     }
368
369     /* AR_REPLACE */
370     if (ar_replace_rule != NULL) {
371         (void)printf("%s:\n", cached_names.ar_replace->string_mb);
372         for (rule= ar_replace_rule; rule != NULL; rule= rule->next)
373             (void)printf("\t%s\n", rule->command_line->string_mb);
374     }
375
376     /* AR_REPLACE */
377     if (ar_replace_rule != NULL) {
378         (void)printf("%s:\n", cached_names.ar_replace->string_mb);
379         for (rule= ar_replace_rule; rule != NULL; rule= rule->next)
380             (void)printf("\t%s\n", rule->command_line->string_mb);
381     }
382
383     /* AR_REPLACE */
384     if (ar_replace_rule != NULL) {
385         (void)printf("%s:\n", cached_names.ar_replace->string_mb);
386         for (rule= ar_replace_rule; rule != NULL; rule= rule->next)
387             (void)printf("\t%s\n", rule->command_line->string_mb);
388     }
389
390     /* AR_REPLACE */
391     if (ar_replace_rule != NULL) {
392         (void)printf("%s:\n", cached_names.ar_replace->string_mb);
393         for (rule= ar_replace_rule; rule != NULL; rule= rule->next)
394             (void)printf("\t%s\n", rule->command_line->string_mb);
395     }
396
397     /* AR_REPLACE */
398     if (ar_replace_rule != NULL) {
399         (void)printf("%s:\n", cached_names.ar_replace->string_mb);
400         for (rule= ar_replace_rule; rule != NULL; rule= rule->next)
401             (void)printf("\t%s\n", rule->command_line->string_mb);
402     }
403
404     /* AR_REPLACE */
405     if (ar_replace_rule != NULL) {
406         (void)printf("%s:\n", cached_names.ar_replace->string_mb);
407         for (rule= ar_replace_rule; rule != NULL; rule= rule->next)
408             (void)printf("\t%s\n", rule->command_line->string_mb);
409     }
410
411     /* AR_REPLACE */
412     if (ar_replace_rule != NULL) {
413         (void)printf("%s:\n", cached_names.ar_replace->string_mb);
414         for (rule= ar_replace_rule; rule != NULL; rule= rule->next)
415             (void)printf("\t%s\n", rule->command_line->string_mb);
416     }
417
418     /* AR_REPLACE */
419     if (ar_replace_rule != NULL) {
420         (void)printf("%s:\n", cached_names.ar_replace->string_mb);
421         for (rule= ar_replace_rule; rule != NULL; rule= rule->next)
422             (void)printf("\t%s\n", rule->command_line->string_mb);
423     }
424
425     /* AR_REPLACE */
426     if (ar_replace_rule != NULL) {
427         (void)printf("%s:\n", cached_names.ar_replace->string_mb);
428         for (rule= ar_replace_rule; rule != NULL; rule= rule->next)
429             (void)printf("\t%s\n", rule->command_line->string_mb);
430     }
431
432     /* AR_REPLACE */
433     if (ar_replace_rule != NULL) {
434         (void)printf("%s:\n", cached_names.ar_replace->string_mb);
435         for (rule= ar_replace_rule; rule != NULL; rule= rule->next)
436             (void)printf("\t%s\n", rule->command_line->string_mb);
437     }
438
439     /* AR_REPLACE */
440     if (ar_replace_rule != NULL) {
441         (void)printf("%s:\n", cached_names.ar_replace->string_mb);
442         for (rule= ar_replace_rule; rule != NULL; rule= rule->next)
443             (void)printf("\t%s\n", rule->command_line->string_mb);
444     }
445
446     /* AR_REPLACE */
447     if (ar_replace_rule != NULL) {
448         (void)printf("%s:\n", cached_names.ar_replace->string_mb);
449         for (rule= ar_replace_rule; rule != NULL; rule= rule->next)
450             (void)printf("\t%s\n", rule->command_line->string_mb);
451     }
452
453     /* AR_REPLACE */
454     if (ar_replace_rule != NULL) {
455         (void)printf("%s:\n", cached_names.ar_replace->string_mb);
456         for (rule= ar_replace_rule; rule != NULL; rule= rule->next)
457             (void)printf("\t%s\n", rule->command_line->string_mb);
458     }
459
460     /* AR_REPLACE */
461     if (ar_replace_rule != NULL) {
462         (void)printf("%s:\n", cached_names.ar_replace->string_mb);
463         for (rule= ar_replace_rule; rule != NULL; rule= rule->next)
464             (void)printf("\t%s\n", rule->command_line->string_mb);
465     }
466
467     /* AR_REPLACE */
468     if (ar_replace_rule != NULL) {
469         (void)printf("%s:\n", cached_names.ar_replace->string_mb);
470         for (rule= ar_replace_rule; rule != NULL; rule= rule->next)
471             (void)printf("\t%s\n", rule->command_line->string_mb);
472     }
473
474     /* AR_REPLACE */
475     if (ar_replace_rule != NULL) {
476         (void)printf("%s:\n", cached_names.ar_replace->string_mb);
477         for (rule= ar_replace_rule; rule != NULL; rule= rule->next)
478             (void)printf("\t%s\n", rule->command_line->string_mb);
479     }
480
481     /* AR_REPLACE */
482     if (ar_replace_rule != NULL) {
483         (void)printf("%s:\n", cached_names.ar_replace->string_mb);
484         for (rule= ar_replace_rule; rule != NULL; rule= rule->next)
485             (void)printf("\t%s\n", rule->command_line->string_mb);
486     }
487
488     /* AR_REPLACE */
489     if (ar_replace_rule != NULL) {
490         (void)printf("%s:\n", cached_names.ar_replace->string_mb);
491         for (rule= ar_replace_rule; rule != NULL; rule= rule->next)
492             (void)printf("\t%s\n", rule->command_line->string_mb);
493     }
494
495     /* AR_REPLACE */
496     if (ar_replace_rule != NULL) {
497         (void)printf("%s:\n", cached_names.ar_replace->string_mb);
498         for (rule= ar_replace_rule; rule != NULL; rule= rule->next)
499             (void)printf("\t%s\n", rule->command_line->string_mb);
500     }
501
502     /* AR_REPLACE */
503     if (ar_replace_rule != NULL) {
504         (void)printf("%s:\n", cached_names.ar_replace->string_mb);
505         for (rule= ar_replace_rule; rule != NULL; rule= rule->next)
506             (void)printf("\t%s\n", rule->command_line->string_mb);
507     }
508
509     /* AR_REPLACE */
510     if (ar_replace_rule != NULL) {
511         (void)printf("%s:\n", cached_names.ar_replace->string_mb);
512         for (rule= ar_replace_rule; rule != NULL; rule= rule->next)
513             (void)printf("\t%s\n", rule->command_line->string_mb);
514     }
515
516     /* AR_REPLACE */
517     if (ar_replace_rule != NULL) {
518         (void)printf("%s:\n", cached_names.ar_replace->string_mb);
519         for (rule= ar_replace_rule; rule != NULL; rule= rule->next)
520             (void)printf("\t%s\n", rule->command_line->string_mb);
521     }
522
523     /* AR_REPLACE */
524     if (ar_replace_rule != NULL) {
525         (void)printf("%s:\n", cached_names.ar_replace->string_mb);
526         for (rule= ar_replace_rule; rule != NULL; rule= rule->next)
527             (void)printf("\t%s\n", rule->command_line->string_mb);
528     }
529
530     /* AR_REPLACE */
531     if (ar_replace_rule != NULL) {
532         (void)printf("%s:\n", cached_names.ar_replace->string_mb);
533         for (rule= ar_replace_rule; rule != NULL; rule= rule->next)
534             (void)printf("\t%s\n", rule->command_line->string_mb);
535     }
536
537     /* AR_REPLACE */
538     if (ar_replace_rule != NULL) {
539         (void)printf("%s:\n", cached_names.ar_replace->string_mb);
540         for (rule= ar_replace_rule; rule != NULL; rule= rule->next)
541             (void)printf("\t%s\n", rule->command_line->string_mb);
542     }
543
544     /* AR_REPLACE */
545     if (ar_replace_rule != NULL) {
546         (void)printf("%s:\n", cached_names.ar_replace->string_mb);
547         for (rule= ar_replace_rule; rule != NULL; rule= rule->next)
548             (void)printf("\t%s\n", rule->command_line->string_mb);
549     }
550
551     /* AR_REPLACE */
552     if (ar_replace_rule != NULL) {
553         (void)printf("%s:\n", cached_names.ar_replace->string_mb);
554         for (rule= ar_replace_rule; rule != NULL; rule= rule->next)
555             (void)printf("\t%s\n", rule->command_line->string_mb);
556     }
557
558     /* AR_REPLACE */
559     if (ar_replace_rule != NULL) {
560         (void)printf("%s:\n", cached_names.ar_replace->string_mb);
561         for (rule= ar_replace_rule; rule != NULL; rule= rule->next)
562             (void)printf("\t%s\n", rule->command_line->string_mb);
563     }
564
565     /* AR_REPLACE */
566     if (ar_replace_rule != NULL) {
567         (void)printf("%s:\n", cached_names.ar_replace->string_mb);
568         for (rule= ar_replace_rule; rule != NULL; rule= rule->next)
569             (void)printf("\t%s\n", rule->command_line->string_mb);
570     }
571
572     /* AR_REPLACE */
573     if (ar_replace_rule != NULL) {
574         (void)printf("%s:\n", cached_names.ar_replace->string_mb);
575         for (rule= ar_replace_rule; rule != NULL; rule= rule->next)
576             (void)printf("\t%s\n", rule->command_line->string_mb);
577     }
578
579     /* AR_REPLACE */
580     if (ar_replace_rule != NULL) {
581         (void)printf("%s:\n", cached_names.ar_replace->string_mb);
582         for (rule= ar_replace_rule; rule != NULL; rule= rule->next)
583             (void)printf("\t%s\n", rule->command_line->string_mb);
584     }
585
586     /* AR_REPLACE */
587     if (ar_replace_rule != NULL) {
588         (void)printf("%s:\n", cached_names.ar_replace->string_mb);
589         for (rule= ar_replace_rule; rule != NULL; rule= rule->next)
590             (void)printf("\t%s\n", rule->command_line->string_mb);
591     }
592
593     /* AR_REPLACE */
594     if (ar_replace_rule != NULL) {
595         (void)printf("%s:\n", cached_names.ar_replace->string_mb);
596         for (rule= ar_replace_rule; rule != NULL; rule= rule->next)
597             (void)printf("\t%s\n", rule->command_line->string_mb);
598     }
599
599 }
```

```

349         };
350
351         /* .DEFAULT */
352         if (default_rule != NULL) {
353             (void)printf("%s:\n", cached_names.default_rule->string_mb);
354             for (rule= default_rule; rule != NULL; rule= rule->next)
355                 (void)printf("\t%s\n", rule->command_line->string_mb);
356         };
357
358         /* .IGNORE */
359         if (is_true(flag.ignore_errors))
360             (void)printf("%s:\n", cached_names.ignore->string_mb);
361
362         /* .KEEP_STATE */
363         if (is_true(flag.keep_state))
364             (void)printf("%s:\n", cached_names.dot_keep_state->string_mb);
365
366         /* .PRECIOUS */
367         (void)printf("%s: ", cached_names.precious->string_mb);
368         for (p = hashtab.begin(), e = hashtab.end(); p != e; p++)
369             if (is_true(p->stat.is_precious | all_precious))
370                 (void)printf("%s ", p->string_mb);
371         (void)printf("\n");
372
373         /* .SCCS_GET */
374         if (sccts_get_rule != NULL) {
375             (void)printf("%s:\n", cached_names.sccts_get->string_mb);
376             for (rule= sccts_get_rule; rule != NULL; rule= rule->next)
377                 (void)printf("\t%s\n", rule->command_line->string_mb);
378         };
379
380         /* .SILENT */
381         if (is_true(flag.silent))
382             (void)printf("%s:\n", cached_names.silent->string_mb);
383
384         /* .SUFFIXES */
385         (void)printf("%s: ", cached_names.suffixes->string_mb);
386         for (dep= suffixes; dep != NULL; dep= dep->next) {
387             (void)printf("%s ", dep->name->string_mb);
388             build_suffix_list(dep->name);
389         };
390         (void)printf("\n\n");
391
392         /* % rules */
393         for (percent= percent_list; percent != NULL; percent= percent->next) {
394             (void) printf("%s:", percent->name->string_mb);
395
396             for (percent_depe= percent->dependencies; percent_depe != NULL;
397                  (void) printf(" %s", percent_depe->name->string_mb));
398
399             (void) printf("\n");
400
401             for (rule= percent->command_template; rule != NULL; rule= rule->
402                  (void)printf("\t%s\n", rule->command_line->string_mb));
403         };
404
405         /* Suffix rules */
406         for (p = hashtab.begin(), e = hashtab.end(); p != e; p++)
407             if (is_false(p->dependency_printed) && (p->string[0] ==
408                 print_rule(p);
409                 p->dependency_printed= true;
410             );
411
412         /* Macro assignments */
413         for (p = hashtab.begin(), e = hashtab.end(); p != e; p++)
414             if (((prop= get_prop(p->prop, macro_prop)) != NULL) &&

```

```

415
416
417
418
419
420         (prop->body.macro.value != NULL)) {
421             (void)printf("%s", p->string_mb);
422             print_value(prop->body.macro.value,
423                         prop->body.macro.daemon);
424         };
425
426         /* Delays */
427         for (p = hashtab.begin(), e = hashtab.end(); p != e; p++)
428             for (prop= get_prop(p->prop, conditional_prop);
429                  prop != NULL;
430                  prop= get_prop(prop->next, conditional_prop)) {
431                     (void)printf("%s := %s",
432                                 p->string_mb,
433                                 prop->body.conditional.name->string_mb);
434                     print_value(prop->body.conditional.value, no_dae);
435                 };
436
437         /* All other dependencies */
438         for (p = hashtab.begin(), e = hashtab.end(); p != e; p++)
439             if (is_false(p->dependency_printed) && (p->cols != no_
440                 print_rule(p));
441         (void)printf("\n");
442         exit(0);
443     }
444
445     /* This is a set of routines for dumping the internal make state
446      * Used for the -p option
447      */
448     void
449     print_value(register Name value, Daemon daemon)
450 #ifdef SUNOS4_AND_AFTER
451
452 #else
453
454 #endif
455 {
456     Chain cp;
457
458     if (value == NULL)
459         (void)printf("=\n");
460     else
461         switch (daemon) {
462             case no_daemon:
463                 (void)printf("= %s\n", value->string_mb);
464                 break;
465             case chain_daemon:
466                 for (cp= (Chain) value; cp != NULL; cp= cp->next)
467                     (void)printf(cp->next == NULL ? "%s" : "%s ",
468                                 cp->name->string_mb);
469                 (void)printf("\n");
470                 break;
471         };
472
473     unchanged_portion_omitted_

```