```
**********************************************************
   86742 Wed May 20 12:27:12 2015
new/usr/src/cmd/make/bin/main.cc
make: prefix errors and such with the right command name
**********************************************************
______unchanged_portion_omitted_

1255 /*
1256  *      parse_command_option(ch)
1257  *
1258  *      Parse make command line options.
1259  *
1260  *      Return value:
1261  *                              Indicates if any -f -c or -M were seen
1262  *
1263  *      Parameters:
1264  *              ch              The character to parse
1265  *
1266  *      Static variables used:
1267  *              dmake_group_specified   Set for make -g
1268  *              dmake_max_jobs_specified        Set for make -j
1269  *              dmake_mode_specified    Set for make -m
1270  *              dmake_add_mode_specified        Set for make -x
1271  *              dmake_compat_mode_specified     Set for make -x SUN_MAKE_COMPAT_
1272  *              dmake_output_mode_specified     Set for make -x DMAKE_OUTPUT_MOD
1273  *              dmake_odir_specified    Set for make -o
1274  *              dmake_rcfile_specified  Set for make -c
1275  *              env_wins                Set for make -e
1276  *              ignore_default_mk       Set for make -r
1277  *              trace_status            Set for make -p
1278  *
1279  *      Global variables used:
1280  *              .make.state path & name set for make -K
1281  *              continue_after_error    Set for make -k
1282  *              debug_level             Set for make -d
1283  *              do_not_exec_rule        Set for make -n
1284  *              filter_stderr           Set for make -X
1285  *              ignore_errors_all       Set for make -i
1286  *              no_parallel             Set for make -R
1287  *              quest                   Set for make -q
1288  *              read_trace_level        Set for make -D
1289  *              report_dependencies     Set for make -P
1290  *              silent_all              Set for make -s
1291  *              touch                   Set for make -t
1292  */
1293 static int
1294 parse_command_option(register char ch)
1295 {
1296         static int              invert_next = 0;
1297         int                     invert_this = invert_next;

1299         invert_next = 0;
1300         switch (ch) {
1301         case '-':                       /* Ignore "--" */
1302                 return 0;
1303         case '~':                       /* Invert next option */
1304                 invert_next = 1;
1305                 return 0;
1306         case 'B':                       /* Obsolete */
1307                 return 0;
1308         case 'b':                       /* Obsolete */
1309                 return 0;
1310         case 'c':                       /* Read alternative dmakerc file */
1311                 if (invert_this) {
1312                         dmake_rcfile_specified = false;
1313                 } else {
```



```
1314                         dmake_rcfile_specified = true;
1315                 }
1316                 return 2;
1317         case 'D':                       /* Show lines read */
1318                 if (invert_this) {
1319                         read_trace_level--;
1320                 } else {
1321                         read_trace_level++;
1322                 }
1323                 return 0;
1324         case 'd':                       /* Debug flag */
1325                 if (invert_this) {
1326                         debug_level--;
1327                 } else {
1328                         debug_level++;
1329                 }
1330                 return 0;
1331         case 'e':                       /* Environment override flag */
1332                 if (invert_this) {
1333                         env_wins = false;
1334                 } else {
1335                         env_wins = true;
1336                 }
1337                 return 0;
1338         case 'f':                       /* Read alternative makefile(s) */
1339                 return 1;
1340         case 'g':                       /* Use alternative DMake group */
1341                 if (invert_this) {
1342                         dmake_group_specified = false;
1343                 } else {
1344                         dmake_group_specified = true;
1345                 }
1346                 return 4;
1347         case 'i':                       /* Ignore errors */
1348                 if (invert_this) {
1349                         ignore_errors_all = false;
1350                 } else {
1351                         ignore_errors_all = true;
1352                 }
1353                 return 0;
1354         case 'j':                       /* Use alternative DMake max jobs */
1355                 if (invert_this) {
1356                         dmake_max_jobs_specified = false;
1357                 } else {
1358                         dmake_mode_type = parallel_mode;
1359                         no_parallel = false;
1360                         dmake_max_jobs_specified = true;
1361                 }
1362                 return 8;
1363         case 'K':                       /* Read alternative .make.state */
1364                 return 256;
1365         case 'k':                       /* Keep making even after errors */
1366                 if (invert_this) {
1367                         continue_after_error = false;
1368                 } else {
1369                         continue_after_error = true;
1370                         continue_after_error_ever_seen = true;
1371                 }
1372                 return 0;
1373         case 'M':                       /* Read alternative make.machines file
1374                 if (invert_this) {
1375                         pmake_machinesfile_specified = false;
1376                 } else {
1377                         pmake_machinesfile_specified = true;
1378                         dmake_mode_type = parallel_mode;
1379                         no_parallel = false;
```

```
1380                }
1381                return 16;
1382        case 'm':                       /* Use alternative DMake build mode */
1383                if (invert_this) {
1384                        dmake_mode_specified = false;
1385                } else {
1386                        dmake_mode_specified = true;
1387                }
1388                return 32;
1389        case 'x':                       /* Use alternative DMake mode */
1390                if (invert_this) {
1391                        dmake_add_mode_specified = false;
1392                } else {
1393                        dmake_add_mode_specified = true;
1394                }
1395                return 1024;
1396        case 'N':                       /* Reverse -n */
1397                if (invert_this) {
1398                        do_not_exec_rule = true;
1399                } else {
1400                        do_not_exec_rule = false;
1401                }
1402                return 0;
1403        case 'n':                       /* Print, not exec commands */
1404                if (invert_this) {
1405                        do_not_exec_rule = false;
1406                } else {
1407                        do_not_exec_rule = true;
1408                }
1409                return 0;
1410        case 'O':                       /* Integrate with maketool, obsolete */
1411                return 0;
1412        case 'o':                       /* Use alternative dmake output dir */
1413                if (invert_this) {
1414                        dmake_odir_specified = false;
1415                } else {
1416                        dmake_odir_specified = true;
1417                }
1418                return 512;
1419        case 'P':                       /* Print for selected targets */
1420                if (invert_this) {
1421                        report_dependencies_level--;
1422                } else {
1423                        report_dependencies_level++;
1424                }
1425                return 0;
1426        case 'p':                       /* Print description */
1427                if (invert_this) {
1428                        trace_status = false;
1429                        do_not_exec_rule = false;
1430                } else {
1431                        trace_status = true;
1432                        do_not_exec_rule = true;
1433                }
1434                return 0;
1435        case 'q':                       /* Question flag */
1436                if (invert_this) {
1437                        quest = false;
1438                } else {
1439                        quest = true;
1440                }
1441                return 0;
1442        case 'R':                       /* Don't run in parallel */
1443                if (invert_this) {
1444                        pmake_cap_r_specified = false;
1445                        no_parallel = false;
```



```
1446                } else {
1447                        pmake_cap_r_specified = true;
1448                        dmake_mode_type = serial_mode;
1449                        no_parallel = true;
1450                }
1451                return 0;
1452        case 'r':                               /* Turn off internal rules */
1453                if (invert_this) {
1454                        ignore_default_mk = false;
1455                } else {
1456                        ignore_default_mk = true;
1457                }
1458                return 0;
1459        case 'S':                               /* Reverse -k */
1460                if (invert_this) {
1461                        continue_after_error = true;
1462                } else {
1463                        continue_after_error = false;
1464                        stop_after_error_ever_seen = true;
1465                }
1466                return 0;
1467        case 's':                               /* Silent flag */
1468                if (invert_this) {
1469                        silent_all = false;
1470                } else {
1471                        silent_all = true;
1472                }
1473                return 0;
1474        case 'T':                               /* Print target list */
1475                if (invert_this) {
1476                        list_all_targets = false;
1477                        do_not_exec_rule = false;
1478                } else {
1479                        list_all_targets = true;
1480                        do_not_exec_rule = true;
1481                }
1482                return 0;
1483        case 't':                               /* Touch flag */
1484                if (invert_this) {
1485                        touch = false;
1486                } else {
1487                        touch = true;
1488                }
1489                return 0;
1490        case 'u':                               /* Unconditional flag */
1491                if (invert_this) {
1492                        build_unconditional = false;
1493                } else {
1494                        build_unconditional = true;
1495                }
1496                return 0;
1497        case 'V':                              /* SVR4 mode */
1498                svr4 = true;
1499                return 0;
1500        case 'v':                              /* Version flag */
1501                if (invert_this) {
1502                } else {
1503                        fprintf(stdout, "%s: %s\n", getprogname(), verstring);
1503                        fprintf(stdout, "dmake: %s\n", verstring);
1504                        exit_status = 0;
1505                        exit(0);
1506                }
1507                return 0;
1508        case 'w':                               /* Unconditional flag */
1509                if (invert_this) {
1510                        report_cwd = false;
```

```
1511                 } else {
1512                         report_cwd = true;
1513                 }
1514                 return 0;
1515 #if 0
1516         case 'X':                       /* Filter stdout */
1517                 if (invert_this) {
1518                         filter_stderr = false;
1519                 } else {
1520                         filter_stderr = true;
1521                 }
1522                 return 0;
1523 #endif
1524         default:
1525                 break;
1526         }
1527         return 0;
1528 }
```

*unchanged_portion_omitted_*

```
3164 static void
3165 report_dir_enter_leave(Boolean entering)
3166 {
3167         char    rcwd[MAXPATHLEN];
3168 static  char *  mlev = NULL;
3169         char *  make_level_str = NULL;
3170         int     make_level_val = 0;

3172         make_level_str = getenv("MAKELEVEL");
3173         if(make_level_str) {
3174                 make_level_val = atoi(make_level_str);
3175         }
3176         if(mlev == NULL) {
3177                 mlev = (char*) malloc(MAXPATHLEN);
3178         }
3179         if(entering) {
3180                 sprintf(mlev, "MAKELEVEL=%d", make_level_val + 1);
3181         } else {
3182                 make_level_val--;
3183                 sprintf(mlev, "MAKELEVEL=%d", make_level_val);
3184         }
3185         putenv(mlev);

3187         if(report_cwd) {
3188                 if(make_level_val <= 0) {
3189                         if(entering) {
3190                                 sprintf(rcwd,
3191                                     gettext("%s: Entering directory `%s'\n"),
3192                                     getprogname(),
3193                                     get_current_path());
3194                         } else {
3195                                 sprintf(rcwd,
3196                                     gettext("%s: Leaving directory `%s'\n"),
3197                                     getprogname(),
3198                                     get_current_path());
3190                                 sprintf( rcwd
3191                                        , gettext("dmake: Entering directory `%s'
3192                                        , get_current_path());
3193                         } else {
3194                                 sprintf( rcwd
3195                                        , gettext("dmake: Leaving directory `%s'\
3196                                        , get_current_path());
3199                         }
3200                 } else {
3201                         if(entering) {
3202                                 sprintf(rcwd,
```



```
3203                                     gettext("%s[%d]: Entering directory `%s'\n")
3204                                     getprogname(),
3205                                     make_level_val, get_current_path());
3206                         } else {
3207                                 sprintf(rcwd,
3208                                     gettext("%s[%d]: Leaving directory `%s'\n"),
3209                                     getprogname(),
3210                                     make_level_val, get_current_path());
3200                                 sprintf( rcwd
3201                                        , gettext("dmake[%d]: Entering directory
3202                                        , make_level_val, get_current_path());
3203                         } else {
3204                                 sprintf( rcwd
3205                                        , gettext("dmake[%d]: Leaving directory `
3206                                        , make_level_val, get_current_path());
3211                         }
3212                 }
3213                 printf("%s", rcwd);
3214         }
3215 }
```

*unchanged_portion_omitted_*

```
**********************************************************
   19333 Wed May 20 12:27:13 2015
new/usr/src/cmd/make/bin/misc.cc
make: prefix errors and such with the right command name
**********************************************************
______unchanged_portion_omitted_

 102 /*****************************************
 103  *
 104  *      String manipulation
 105  */

 107 /*****************************************
 108  *
 109  *      Nameblock property handling
 110  */

 112 /*****************************************
 113  *
 114  *      Error message handling
 115  */

 117 /*
 118  *      fatal(format, args...)
 119  *
 120  *      Print a message and die
 121  *
 122  *      Parameters:
 123  *              format          printf type format string
 124  *              args            Arguments to match the format
 125  *
 126  *      Global variables used:
 127  *              fatal_in_progress Indicates if this is a recursive call
 128  *              parallel_process_cnt Do we need to wait for anything?
 129  *              report_pwd      Should we report the current path?
 130  */
 131 /*VARARGS*/
 132 void
 133 fatal(const char *message, ...)
 134 {
 135         va_list args;

 137         va_start(args, message);
 138         (void) fflush(stdout);
 139         (void) fprintf(stderr, gettext("%s: Fatal error: "), getprogname());
 139         (void) fprintf(stderr, gettext("make: Fatal error: "));
 140         (void) vfprintf(stderr, message, args);
 141         (void) fprintf(stderr, "\n");
 142         va_end(args);
 143         if (report_pwd) {
 144                 (void) fprintf(stderr,
 145                                 gettext("Current working directory %s\n"),
 146                                 get_current_path());
 147         }
 148         (void) fflush(stderr);
 149         if (fatal_in_progress) {
 150                 exit_status = 1;
 151                 exit(1);
 152         }
 153         fatal_in_progress = true;
 154         /* Let all parallel children finish */
 155         if ((dmake_mode_type == parallel_mode) &&
 156             (parallel_process_cnt > 0)) {
 157                 (void) fprintf(stderr,
 158                                 gettext("Waiting for %d %s to finish\n"),
 159                                 parallel_process_cnt,
```



```
 160                                 parallel_process_cnt == 1 ?
 161                                 gettext("job") : gettext("jobs"));
 162                 (void) fflush(stderr);
 163         }

 165         while (parallel_process_cnt > 0) {
 166                 await_parallel(true);
 167                 finish_children(false);
 168         }

 170         job_adjust_fini();

 172         exit_status = 1;
 173         exit(1);
 174 }

 176 /*
 177  *      warning(format, args...)
 178  *
 179  *      Print a message and continue.
 180  *
 181  *      Parameters:
 182  *              format          printf type format string
 183  *              args            Arguments to match the format
 184  *
 185  *      Global variables used:
 186  *              report_pwd      Should we report the current path?
 187  */
 188 /*VARARGS*/
 189 void
 190 warning(char * message, ...)
 191 {
 192         va_list args;

 194         va_start(args, message);
 195         (void) fflush(stdout);
 196         (void) fprintf(stderr, gettext("%s: Warning: "), getprogname());
 196         (void) fprintf(stderr, gettext("make: Warning: "));
 197         (void) vfprintf(stderr, message, args);
 198         (void) fprintf(stderr, "\n");
 199         va_end(args);
 200         if (report_pwd) {
 201                 (void) fprintf(stderr,
 202                                 gettext("Current working directory %s\n"),
 203                                 get_current_path());
 204         }
 205         (void) fflush(stderr);
 206 }
______unchanged_portion_omitted_
```

```
**********************************************************
   56713 Wed May 20 12:27:13 2015
new/usr/src/cmd/make/bin/read.cc
make: prefix errors and such with the right command name
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright 2006 Sun Microsystems, Inc. All rights reserved.
 * Use is subject to license terms.
 */

/*
 *      read.c
 *
 *      This file contains the makefile reader.
 */

/*
 * Included files
 */
#include <alloca.h>             /* alloca() */
#include <errno.h>              /* errno */
#include <fcntl.h>              /* fcntl() */
#include <mk/defs.h>
#include <mksh/macro.h>         /* expand_value(), expand_macro() */
#include <mksh/misc.h>          /* getmem() */
#include <mksh/read.h>          /* get_next_block_fn() */
#include <sys/uio.h>            /* read() */
#include <unistd.h>             /* read(), unlink() */
#include <libintl.h>



/*
 * typedefs & structs
 */

/*
 * Static variables
 */

static int line_started_with_space=0; // Used to diagnose spaces instead of tabs

/*
 * File table of contents
 */
static  void            parse_makefile(register Name true_makefile_name, registe
static  Source          push_macro_value(register Source bp, register wchar_t *b
```



```
extern  void            enter_target_groups_and_dependencies(Name_vector target,
extern  Name            normalize_name(register wchar_t *name_string, register i

/*
 *      read_simple_file(makefile_name, chase_path, doname_it,
 *              complain, must_exist, report_file, lock_makefile)
 *
 *      Make the makefile and setup to read it. Actually read it if it is stdio
 *
 *      Return value:
 *                              false if the read failed
 *
 *      Parameters:
 *              makefile_name   Name of the file to read
 *              chase_path      Use the makefile path when opening file
 *              doname_it       Call doname() to build the file first
 *              complain        Print message if doname/open fails
 *              must_exist      Generate fatal if file is missing
 *              report_file     Report file when running -P
 *              lock_makefile   Lock the makefile when reading
 *
 *      Static variables used:
 *
 *      Global variables used:
 *              do_not_exec_rule Is -n on?
 *              file_being_read Set to the name of the new file
 *              line_number     The number of the current makefile line
 *              makefiles_used  A list of all makefiles used, appended to
 */


Boolean
read_simple_file(register Name makefile_name, register Boolean chase_path, regis
{
        static short            max_include_depth;
        register Property       makefile = maybe_append_prop(makefile_name,
                                                             makefile_prop);
        Boolean                 forget_after_parse = false;
        static pathpt           makefile_path;
        register int            n;
        char                    *path;
        register Source         source = ALLOC(Source);
        Property                orig_makefile = makefile;
        Dependency              *dpp;
        Dependency              dp;
        register int            length;
        wchar_t                 *previous_file_being_read = file_being_read;
        int                     previous_line_number = line_number;
        wchar_t                 previous_current_makefile[MAXPATHLEN];
        Makefile_type           save_makefile_type;
        Name                    normalized_makefile_name;
        register wchar_t        *string_start;
        register wchar_t        *string_end;



        wchar_t * wcb = get_wstring(makefile_name->string_mb);

        if (max_include_depth++ >= 40) {
                fatal(gettext("Too many nested include statements"));
        }
        if (makefile->body.makefile.contents != NULL) {
                retmem(makefile->body.makefile.contents);
        }
        source->inp_buf =
          source->inp_buf_ptr =
```

```
128             source->inp_buf_end = NULL;
129         source->error_converting = false;
130         makefile->body.makefile.contents = NULL;
131         makefile->body.makefile.size = 0;
132         if ((makefile_name->hash.length != 1) ||
133             (wcb[0] != (int) hyphen_char)) {
134                 if ((makefile->body.makefile.contents == NULL) &&
135                     (doname_it)) {
136                         if (makefile_path == NULL) {
137                                 char *pfx = make_install_prefix();
138                                 char *path;
139 
140                                 add_dir_to_path(".",
141                                                 &makefile_path,
142                                                 -1);
143 
144                                 // As regularly installed
145                                 asprintf(&path, "%s/../share/lib/make", pfx);
146                                 add_dir_to_path(path, &makefile_path, -1);
147                                 free(path);
148 
149                                 // Tools build
150                                 asprintf(&path, "%s/../../share/", pfx);
151                                 add_dir_to_path(path, &makefile_path, -1);
152                                 free(path);
153 
154                                 add_dir_to_path("/usr/share/lib/make",
155                                                 &makefile_path,
156                                                 -1);
157                                 add_dir_to_path("/etc/default",
158                                                 &makefile_path,
159                                                 -1);
160 
161                                 free(pfx);
162                         }
163                         save_makefile_type = makefile_type;
164                         makefile_type = reading_nothing;
165                         if (doname(makefile_name, true, false) == build_dont_kno
166                                 /* Try normalized filename */
167                                 string_start=get_wstring(makefile_name->string_m
168                                 for (string_end=string_start+1; *string_end != L
169                                 normalized_makefile_name=normalize_name(string_s
170                                 if ((strcmp(makefile_name->string_mb, normalized
171                                         (doname(normalized_makefile_name, true,
172                                         n = access_vroot(makefile_name->string_m
173                                                  4,
174                                                  chase_path ?
175                                                  makefile_path : NULL,
176                                                  VROOT_DEFAULT);
177                                         if (n == 0) {
178                                                 get_vroot_path((char **) NULL,
179                                                        &path,
180                                                        (char **) NULL);
181                                                 if ((path[0] == (int) period_cha
182                                                     (path[1] == (int) slash_char
183                                                         path += 2;
184                                                 }
185                                                 MBSTOWCS(wcs_buffer, path);
186                                                 makefile_name = GETNAME(wcs_buff
187                                                                 FIND_LENGTH);
188                                         }
189                                 }
190                                 retmem(string_start);
191                                 /*
192                                  * Commented out: retmem_mb(normalized_makefile_
193                                  * We have to return this memory, but it seems t
```



```
194                                  * in dmake or in Sun C++ 5.7 compiler (it works
195                                  * is compiled using Sun C++ 5.6).
196                                  */
197                                 // retmem_mb(normalized_makefile_name->string_mb
198                         }
199                         makefile_type = save_makefile_type;
200                 }
201                 source->string.free_after_use = false;
202                 source->previous = NULL;
203                 source->already_expanded = false;
204                 /* Lock the file for read, but not when -n. */
205                 if (lock_makefile &&
206                     !do_not_exec_rule) {
207 
208                          make_state_lockfile = getmem(strlen(make_state->string_
209                          (void) sprintf(make_state_lockfile,
210                                         "%s.lock",
211                                         make_state->string_mb);
212                         (void) file_lock(make_state->string_mb,
213                                          make_state_lockfile,
214                                          (int *) &make_state_locked,
215                                          0);
216                         if(!make_state_locked) {
217                                 printf("-- NO LOCKING for read\n");
218                                 retmem_mb(make_state_lockfile);
219                                 make_state_lockfile = 0;
220                                 return failed;
221                         }
222                 }
223                 if (makefile->body.makefile.contents == NULL) {
224                         save_makefile_type = makefile_type;
225                         makefile_type = reading_nothing;
226                         if ((doname_it) &&
227                             (doname(makefile_name, true, false) == build_failed)
228                                 if (complain) {
229                                         (void) fprintf(stderr,
230                                                        gettext("%s: Couldn't mak
231                                                        getprogname(),
230                                                        gettext("make: Couldn't m
232                                                        makefile_name->string_mb)
233                                 }
234                                 max_include_depth--;
235                                 makefile_type = save_makefile_type;
236                                 return failed;
237                         }
238                         makefile_type = save_makefile_type;
239                         //
240                         // Before calling exists() make sure that we have the ri
241                         //
242                         makefile_name->stat.time = file_no_time;
243 
244                         if (exists(makefile_name) == file_doesnt_exist) {
245                                 if (complain ||
246                                     (makefile_name->stat.stat_errno != ENOENT))
247                                         if (must_exist) {
248                                                 fatal(gettext("Can't find '%s':
249                                                       makefile_name->string_mb,
250                                                       errmsg(makefile_name->
251                                                              stat.stat_errno));
252                                         } else {
253                                                 warning(gettext("Can't find '%s'
254                                                         makefile_name->string_mb
255                                                         errmsg(makefile_name->
256                                                                stat.stat_errno))
257                                         }
258                                 }
```

```
259                                 max_include_depth--;
260                                 if(make_state_locked && (make_state_lockfile !=
261                                         (void) unlink(make_state_lockfile);
262                                         retmem_mb(make_state_lockfile);
263                                         make_state_lockfile = NULL;
264                                         make_state_locked = false;
265                                 }
266                                 retmem(wcb);
267                                 retmem_mb((char *)source);
268                                 return failed;
269                         }
270                         /*
271                          * These values are the size and bytes of
272                          * the MULTI-BYTE makefile.
273                          */
274                         orig_makefile->body.makefile.size =
275                           makefile->body.makefile.size =
276                             source->bytes_left_in_file =
277                               makefile_name->stat.size;
278                         if (report_file) {
279                                 for (dpp = &makefiles_used;
280                                      *dpp != NULL;
281                                      dpp = &(*dpp)->next);
282                                 dp = ALLOC(Dependency);
283                                 dp->next = NULL;
284                                 dp->name = makefile_name;
285                                 dp->automatic = false;
286                                 dp->stale = false;
287                                 dp->built = false;
288                                 *dpp = dp;
289                         }
290                         source->fd = open_vroot(makefile_name->string_mb,
291                                                 O_RDONLY,
292                                                 0,
293                                                 NULL,
294                                                 VROOT_DEFAULT);
295                         if (source->fd < 0) {
296                                 if (complain || (errno != ENOENT)) {
297                                         if (must_exist) {
298                                                 fatal(gettext("Can't open '%s':
299                                                       makefile_name->string_mb,
300                                                       errmsg(errno));
301                                         } else {
302                                                 warning(gettext("Can't open '%s'
303                                                         makefile_name->string_mb
304                                                         errmsg(errno));
305                                         }
306                                 }
307                                 max_include_depth--;
308                                 return failed;
309                         }
310                         (void) fcntl(source->fd, F_SETFD, 1);
311                         orig_makefile->body.makefile.contents =
312                           makefile->body.makefile.contents =
313                             source->string.text.p =
314                               source->string.buffer.start =
315                                 ALLOC_WC((int) (makefile_name->stat.size + 2));
316                         if (makefile_type == reading_cpp_file) {
317                                 forget_after_parse = true;
318                         }
319                         source->string.text.end = source->string.text.p;
320                         source->string.buffer.end =
321                           source->string.text.p + makefile_name->stat.size;
322                 } else {
323                         /* Do we ever reach here? */
324                         source->fd = -1;
```



```
325                         source->string.text.p =
326                           source->string.buffer.start =
327                             makefile->body.makefile.contents;
328                         source->string.text.end =
329                           source->string.buffer.end =
330                             source->string.text.p + makefile->body.makefile.size
331                         source->bytes_left_in_file =
332                           makefile->body.makefile.size;
333                 }
334                 file_being_read = wcb;
335         } else {
336                 char            *stdin_text_p;
337                 char            *stdin_text_end;
338                 char            *stdin_buffer_start;
339                 char            *stdin_buffer_end;
340                 char            *p_mb;
341                 int             num_mb_chars;
342                 size_t          num_wc_chars;

344                 MBSTOWCS(wcs_buffer, "Standard in");
345                 makefile_name = GETNAME(wcs_buffer, FIND_LENGTH);
346                 /*
347                  * Memory to read standard in, then convert it
348                  * to wide char strings.
349                  */
350                 stdin_buffer_start =
351                   stdin_text_p = getmem(length = 1024);
352                 stdin_buffer_end = stdin_text_p + length;
353                 MBSTOWCS(wcs_buffer, "standard input");
354                 file_being_read = (wchar_t *) wsdup(wcs_buffer);
355                 line_number = 0;
356                 while ((n = read(fileno(stdin),
357                                  stdin_text_p,
358                                  length)) > 0) {
359                         length -= n;
360                         stdin_text_p += n;
361                         if (length == 0) {
362                                 p_mb = getmem(length = 1024 +
363                                               (stdin_buffer_end -
364                                                stdin_buffer_start));
365                                 (void) strncpy(p_mb,
366                                                stdin_buffer_start,
367                                                (stdin_buffer_end -
368                                                 stdin_buffer_start));
369                                 retmem_mb(stdin_buffer_start);
370                                 stdin_text_p = p_mb +
371                                   (stdin_buffer_end - stdin_buffer_start);
372                                 stdin_buffer_start = p_mb;
373                                 stdin_buffer_end =
374                                   stdin_buffer_start + length;
375                                 length = 1024;
376                         }
377                 }
378                 if (n < 0) {
379                         fatal(gettext("Error reading standard input: %s"),
380                               errmsg(errno));
381                 }
382                 stdin_text_p = stdin_buffer_start;
383                 stdin_text_end = stdin_buffer_end - length;
384                 num_mb_chars = stdin_text_end - stdin_text_p;

386                 /*
387                  * Now, convert the sequence of multibyte chars into
388                  * a sequence of corresponding wide character codes.
389                  */
390                 source->string.free_after_use = false;
```

```
391                 source->previous = NULL;
392                 source->bytes_left_in_file = 0;
393                 source->fd = -1;
394                 source->already_expanded = false;
395                 source->string.buffer.start =
396                   source->string.text.p = ALLOC_WC(num_mb_chars + 1);
397                 source->string.buffer.end =
398                     source->string.text.p + num_mb_chars;
399                 num_wc_chars = mbstowcs(source->string.text.p,
400                                         stdin_text_p,
401                                         num_mb_chars);
402                 if ((int) num_wc_chars >= 0) {
403                         source->string.text.end =
404                           source->string.text.p + num_wc_chars;
405                 }
406                 (void) retmem_mb(stdin_text_p);
407         }
408         line_number = 1;
409         if (trace_reader) {
410                 (void) printf(gettext(">>>>>>>>>>>>>>>> Reading makefile %s\n"),
411                               makefile_name->string_mb);
412         }
413         parse_makefile(makefile_name, source);
414         if (trace_reader) {
415                 (void) printf(gettext(">>>>>>>>>>>>>>>> End of makefile %s\n"),
416                               makefile_name->string_mb);
417         }
418         if(file_being_read) {
419                 retmem(file_being_read);
420         }
421         file_being_read = previous_file_being_read;
422         line_number = previous_line_number;
423         makefile_type = reading_nothing;
424         max_include_depth--;
425         if (make_state_locked) {
426                 /* Unlock .make.state. */
427                 unlink(make_state_lockfile);
428                 make_state_locked = false;
429                 retmem_mb(make_state_lockfile);
430         }
431         if (forget_after_parse) {
432                 retmem(makefile->body.makefile.contents);
433                 makefile->body.makefile.contents = NULL;
434         }
435         retmem_mb((char *)source);
436         return succeeded;
437 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   51190 Wed May 20 12:27:14 2015
new/usr/src/cmd/make/bin/read2.cc
make: prefix errors and such with the right command name
**********************************************************
______unchanged_portion_omitted_

1836 /*
1837  *      fatal_reader(format, args...)
1838  *
1839  *      Parameters:
1840  *              format          printf style format string
1841  *              args            arguments to match the format
1842  *
1843  *      Global variables used:
1844  *              file_being_read Name of the makefile being read
1845  *              line_number     Line that is being read
1846  *              report_pwd      Indicates whether current path should be shown
1847  *              temp_file_name  When reading tempfile we report that name
1848  */
1849 /*VARARGS*/
1850 void
1851 fatal_reader(char * pattern, ...)
1852 {
1853         va_list args;
1854         char    message[1000];

1856         va_start(args, pattern);
1857         if (file_being_read != NULL) {
1858                 WCSTOMBS(mbs_buffer, file_being_read);
1859                 if (line_number != 0) {
1860                         (void) sprintf(message,
1861                                        gettext("%s, line %d: %s"),
1862                                        mbs_buffer,
1863                                        line_number,
1864                                        pattern);
1865                 } else {
1866                         (void) sprintf(message,
1867                                        "%s: %s",
1868                                        mbs_buffer,
1869                                        pattern);
1870                 }
1871                 pattern = message;
1872         }

1874         (void) fflush(stdout);
1875         (void) fprintf(stderr, gettext("%s: Fatal error in reader: "),
1876             getprogname());
1875         (void) fprintf(stderr, gettext("make: Fatal error in reader: "));
1877         (void) vfprintf(stderr, pattern, args);
1878         (void) fprintf(stderr, "\n");
1879         va_end(args);

1881         if (temp_file_name != NULL) {
1882                 (void) fprintf(stderr,
1883                                gettext("%s: Temp-file %s not removed\n"),
1884                                getprogname(),
1882                                gettext("make: Temp-file %s not removed\n"),
1885                                temp_file_name->string_mb);
1886                 temp_file_name = NULL;
1887         }

1889         if (report_pwd) {
1890                 (void) fprintf(stderr,
1891                                gettext("Current working directory %s\n"),
1892                                get_current_path());
```



```
1893         }
1894         (void) fflush(stderr);
1895         exit_status = 1;
1896         exit(1);
1897 }
______unchanged_portion_omitted_
```