```
**********************************************************
    5390 Wed May 20 11:22:24 2015
new/usr/src/cmd/make/bin/globals.cc
make: fix GCC warnings
**********************************************************
```

```
/*
 * 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 2004 Sun Microsystems, Inc. All rights reserved.
 * Use is subject to license terms.
 */

/*
 *      globals.cc
 *
 *      This declares all global variables
 */

/*
 * Included files
 */
#include <nl_types.h>
#include <mk/defs.h>
#include <sys/stat.h>

/*
 * Defined macros
 */

/*
 * typedefs & structs
 */

/*
 * Global variables used by make only
 */
        FILE            *dependency_report_file;

/*
 * Global variables used by make
 */
        Boolean         allrules_read=false;
        Name            posix_name;
        Name            svr4_name;
        Boolean         sdot_target;    /* used to identify s.m(/M)akefile */
        Boolean         all_parallel;                   /* TEAMWARE_MAKE_CMN */
        Boolean         assign_done;
        int foo;
```



```
        Boolean         build_failed_seen;
#ifdef DISTRIBUTED
        Boolean         building_serial;
#endif
        Name            built_last_make_run;
        Name            c_at;
#ifdef DISTRIBUTED
        Boolean         called_make = false;
#endif
        Boolean         cleanup;
        Boolean         close_report;
        Boolean         command_changed;
        Boolean         commands_done;
        Chain           conditional_targets;
        Name            conditionals;
        Boolean         continue_after_error;           /* '-k' */
        Property        current_line;
        Name            current_make_version;
        Name            current_target;
        short           debug_level;
        Cmd_line        default_rule;
        Name            default_rule_name;
        Name            default_target_to_build;
        Name            dmake_group;
        Name            dmake_max_jobs;
        Name            dmake_mode;
        DMake_mode      dmake_mode_type;
        Name            dmake_output_mode;
        DMake_output_mode       output_mode = txt1_mode;
        Name            dmake_odir;
        Name            dmake_rcfile;
        Name            done;
        Name            dot;
        Name            dot_keep_state;
        Name            dot_keep_state_file;
        Name            empty_name;
#if defined(HP_UX) ||  defined(linux)
        int             exit_status;
#endif
        Boolean         fatal_in_progress;
        int             file_number;
#if 0
        Boolean         filter_stderr;                  /* '-X' */
#endif
        Name            force;
        Name            ignore_name;
        Boolean         ignore_errors;                  /* '-i' */
        Boolean         ignore_errors_all;              /* '-i' */
        Name            init;
        int             job_msg_id;
        Boolean         keep_state;
        Name            make_state;
#ifdef TEAMWARE_MAKE_CMN
        timestruc_t     make_state_before;
#endif
        Dependency      makefiles_used;
        Name            makeflags;
//      Boolean         make_state_locked; // Moved to lib/mksh
        Name            make_version;
        char            mbs_buffer2[(MAXPATHLEN * MB_LEN_MAX)];
        char            *mbs_ptr;
        char            *mbs_ptr2;
        int             mtool_msgs_fd;
        Boolean         depinfo_already_read = false;
#ifdef NSE
        Name            derived_src;
```

```
128         Boolean         nse;                            /* NSE on */
129         Name            nse_backquote_seen;
130         char            nse_depinfo_lockfile[MAXPATHLEN];
131         Boolean         nse_depinfo_locked;
132         Boolean         nse_did_recursion;
133         Name            nse_shell_var_used;
134         Boolean         nse_watch_vars = false;
135         wchar_t         current_makefile[MAXPATHLEN];
136 #endif
137         Boolean         no_action_was_taken = true;     /* true if we've not **
138                                                         ** run any command   */

140         Boolean         no_parallel = false;            /* TEAMWARE_MAKE_CMN */
141 #ifdef SGE_SUPPORT
142         Boolean         grid = false;                   /* TEAMWARE_MAKE_CMN */
143 #endif
144         Name            no_parallel_name;
145         Name            not_auto;
146         Boolean         only_parallel;                  /* TEAMWARE_MAKE_CMN */
147         Boolean         parallel;                       /* TEAMWARE_MAKE_CMN */
148         Name            parallel_name;
149         Name            localhost_name;
150         int             parallel_process_cnt;
151         Percent         percent_list;
152         Dyntarget       dyntarget_list;
153         Name            plus;
154         Name            pmake_machinesfile;
155         Name            precious;
156         Name            primary_makefile;
157         Boolean         quest;                          /* '-q' */
158         short           read_trace_level;
159         Boolean         reading_dependencies = false;
160         Name            recursive_name;
161         int             recursion_level;
162         short           report_dependencies_level = 0;  /* -P */
163         Boolean         report_pwd;
164         Boolean         rewrite_statefile;
165         Running         running_list;
166         char            *sccs_dir_path;
167         Name            sccs_get_name;
168         Name            sccs_get_posix_name;
169         Cmd_line        sccs_get_rule;
170         Cmd_line        sccs_get_org_rule;
171         Cmd_line        sccs_get_posix_rule;
172         Name            get_name;
173         Cmd_line        get_rule;
174         Name            get_posix_name;
175         Cmd_line        get_posix_rule;
176         Boolean         send_mtool_msgs;                /* '-K' */
177         Boolean         all_precious;
178         Boolean         silent_all;                     /* '-s' */
179         Boolean         report_cwd;                     /* '-w' */
180         Boolean         silent;                         /* '-s' */
181         Name            silent_name;
182         char            *stderr_file = NULL;
183         char            *stdout_file = NULL;
184 #ifdef SGE_SUPPORT
185         char            script_file[MAXPATHLEN] = "";
186 #endif
187         Boolean         stdout_stderr_same;
188         Dependency      suffixes;
189         Name            suffixes_name;
190         Name            sunpro_dependencies;
191         Boolean         target_variants;
192         const char      *tmpdir = NOCATGETS("/tmp");
193         const char      *temp_file_directory = NOCATGETS(".");
```



```
192         char            *tmpdir = NOCATGETS("/tmp");
193         char            *temp_file_directory = NOCATGETS(".");
194         Name            temp_file_name;
195         short           temp_file_number;
196         time_t          timing_start;
197         wchar_t         *top_level_target;
198         Boolean         touch;                          /* '-t' */
199         Boolean         trace_reader;                   /* '-D' */
200         Boolean         build_unconditional;            /* '-u' */
201         pathpt          vroot_path = VROOT_DEFAULT;
202         Name            wait_name;
203         wchar_t         wcs_buffer2[MAXPATHLEN];
204         wchar_t         *wcs_ptr;
205         wchar_t         *wcs_ptr2;
206         nl_catd         catd;
207         long int        hostid;

209 /*
210  * File table of contents
211  */
```

```
**********************************************************
  103217 Wed May 20 11:22:24 2015
new/usr/src/cmd/make/bin/main.cc
make: fix GCC warnings
**********************************************************
_____unchanged_portion_omitted_
```

```
 176 #endif

 178 extern  Name            normalize_name(register wchar_t *name_string, register i

 180 extern  int             main(int, char * []);

 182 static  void            append_makeflags_string(Name, String);
 183 static  void            doalarm(int);
 184 static  void            enter_argv_values(int , char **, ASCII_Dyn_Array *);
 185 static  void            make_targets(int, char **, Boolean);
 186 static  int             parse_command_option(char);
 187 static  void            read_command_options(int, char **);
 188 static  void            read_environment(Boolean);
 189 static  void            read_files_and_state(int, char **);
 190 static  Boolean         read_makefile(Name, Boolean, Boolean, Boolean);
 191 static  void            report_recursion(Name);
 192 static  void            set_sgs_support(void);
 193 static  void            setup_for_projectdir(void);
 194 static  void            setup_makeflags_argv(void);
 195 static  void            report_dir_enter_leave(Boolean entering);

 197 extern void expand_value(Name, register String , Boolean);

 199 #ifdef DISTRIBUTED
 200         extern  int             dmake_ofd;
 201         extern  FILE*           dmake_ofp;
 202         extern  int             rxmPid;
 203         extern  XDR             xdrs_out;
 204 #endif
 205 #ifdef TEAMWARE_MAKE_CMN
 206         extern  char            verstring[];
 207 #endif

 209 jmp_buf jmpbuffer;
 210 #if !defined(linux)
 211 extern nl_catd catd;
 211 nl_catd catd;
 212 #endif

 214 /*
 215  *      main(argc, argv)
 216  *
 217  *      Parameters:
 218  *              argc                    You know what this is
 219  *              argv                    You know what this is
 220  *
 221  *      Static variables used:
 222  *              list_all_targets        make -T seen
 223  *              trace_status            make -p seen
 224  *
 225  *      Global variables used:
 226  *              debug_level             Should we trace make actions?
 227  *              keep_state              Set if .KEEP_STATE seen
 228  *              makeflags               The Name "MAKEFLAGS", used to get macro
 229  *              remote_command_name     Name of remote invocation cmd ("on")
 230  *              running_list            List of parallel running processes
 231  *              stdout_stderr_same      true if stdout and stderr are the same
 232  *              auto_dependencies       The Name "SUNPRO_DEPENDENCIES"
 233  *              temp_file_directory     Set to the dir where we create tmp file
 234  *              trace_reader            Set to reflect tracing status
```



```
 235  *              working_on_targets      Set when building user targets
 236  */
 237 int
 238 main(int argc, char *argv[])
 239 {
 240         /*
 241          * cp is a -> to the value of the MAKEFLAGS env var,
 242          * which has to be regular chars.
 243          */
 244         register char           *cp;
 245         char                    make_state_dir[MAXPATHLEN];
 246         Boolean                 parallel_flag = false;
 247         char                    *prognameptr;
 248         char                    *slash_ptr;
 249         mode_t                  um;
 250         int                     i;
 251 #ifdef TEAMWARE_MAKE_CMN
 252         struct itimerval        value;
 253         char                    def_dmakerc_path[MAXPATHLEN];
 254         Name                    dmake_name, dmake_name2;
 255         Name                    dmake_value, dmake_value2;
 256         Property                prop, prop2;
 257         struct stat             statbuf;
 258         int                     statval;
 259 #endif

 261 #ifndef PARALLEL
 262         struct stat             out_stat, err_stat;
 263 #endif
 264         hostid = gethostid();
 265 #ifdef TEAMWARE_MAKE_CMN
 266         avo_get_user(NULL, NULL); // Initialize user name
 267 #endif
 268         bsd_signals();

 270         (void) setlocale(LC_ALL, "");

 272 #if defined (HP_UX) || defined(linux)
 273        /* HP-UX users typically will not have NLSPATH set, and this binary
 274         * requires that it be set.  On HP-UX 9.0x, /usr/lib/nls/%L/%N.cat is
 275         * the path to set it to.
 276         */

 278        if (getenv(NOCATGETS("NLSPATH")) == NULL) {
 279          putenv(NOCATGETS("NLSPATH=/usr/lib/nls/%L/%N.cat"));
 280         }
 281 #endif

 283 #ifdef DMAKE_STATISTICS
 284         if (getenv(NOCATGETS("DMAKE_STATISTICS"))) {
 285                 getname_stat = true;
 286         }
 287 #endif


 290         /*
 291          * avo_init() sets the umask to 0.  Save it here and restore
 292          * it after the avo_init() call.
 293          */
 294 #if defined(TEAMWARE_MAKE_CMN) || defined(MAKETOOL)
 295         um = umask(0);
 296         avo_init(argv[0]);
 297         umask(um);

 299 #ifdef USE_DMS_CCR
 300         usageTracking = new Avo_usage_tracking(NOCATGETS("dmake"), argc, argv);
```

```
#else
        cleanup = new Avo_cleanup(NOCATGETS("dmake"), argc, argv);
#endif
#endif

#if defined(TEAMWARE_MAKE_CMN)
        catd = catopen(AVO_DOMAIN_DMAKE, NL_CAT_LOCALE);
        libcli_init();

#ifdef _CHECK_UPDATE_H
        /* This is for dmake only (not for Solaris make).
         * Check (in background) if there is an update (dmake patch)
         * and inform user
         */
        {
                Avo_err         *err;
                char            *dir;
                err = avo_find_run_dir(&dir);
                if (AVO_OK == err) {
                        AU_check_update_service(NOCATGETS("Dmake"), dir);
                }
        }
#endif /* _CHECK_UPDATE_H */
#endif

// ---> fprintf(stderr, catgets(catd, 15, 666, "--- SUN make ---\n"));


#if defined(TEAMWARE_MAKE_CMN) || defined(MAKETOOL)
/*
 * I put libmksdmsi18n_init() under #ifdef because it requires avo_i18n_init()
 * from avo_util library.
 */
        libmksdmsi18n_init();
#ifdef USE_DMS_CCR
        libpubdmsi18n_init();
#endif
#endif


#ifndef TEAMWARE_MAKE_CMN
        textdomain(NOCATGETS("SUNW_SPRO_MAKE"));
#endif /* TEAMWARE_MAKE_CMN */

#ifdef TEAMWARE_MAKE_CMN
        g_argc = argc;
        g_argv = (char **) malloc((g_argc + 1) * sizeof(char *));
        for (i = 0; i < argc; i++) {
                g_argv[i] = argv[i];
        }
        g_argv[i] = NULL;
#endif /* TEAMWARE_MAKE_CMN */

        /*
         * Set argv_zero_string to some form of argv[0] for
         * recursive MAKE builds.
         */

        if (*argv[0] == (int) slash_char) {
                /* argv[0] starts with a slash */
                argv_zero_string = strdup(argv[0]);
        } else if (strchr(argv[0], (int) slash_char) == NULL) {
                /* argv[0] contains no slashes */
                argv_zero_string = strdup(argv[0]);
        } else {
                /*
```



```
                 * argv[0] contains at least one slash,
                 * but doesn't start with a slash
                 */
                char    *tmp_current_path;
                char    *tmp_string;

                tmp_current_path = get_current_path();
                tmp_string = getmem(strlen(tmp_current_path) + 1 +
                                    strlen(argv[0]) + 1);
                (void) sprintf(tmp_string,
                               "%s/%s",
                               tmp_current_path,
                               argv[0]);
                argv_zero_string = strdup(tmp_string);
                retmem_mb(tmp_string);
        }

        /*
         * The following flags are reset if we don't have the
         * (.nse_depinfo or .make.state) files locked and only set
         * AFTER the file has been locked. This ensures that if the user
         * interrupts the program while file_lock() is waiting to lock
         * the file, the interrupt handler doesn't remove a lock
         * that doesn't belong to us.
         */
        make_state_lockfile = NULL;
        make_state_locked = false;

#ifdef NSE
        nse_depinfo_lockfile[0] = '\0';
        nse_depinfo_locked = false;
#endif

        /*
         * look for last slash char in the path to look at the binary
         * name. This is to resolve the hard link and invoke make
         * in svr4 mode.
         */

        /* Sun OS make standart */
        svr4 = false;
        posix = false;
        if(!strcmp(argv_zero_string, NOCATGETS("/usr/xpg4/bin/make"))) {
                svr4 = false;
                posix = true;
        } else {
                prognameptr = strrchr(argv[0], '/');
                if(prognameptr) {
                        prognameptr++;
                } else {
                        prognameptr = argv[0];
                }
                if(!strcmp(prognameptr, NOCATGETS("svr4.make"))) {
                        svr4 = true;
                        posix = false;
                }
        }
#if !defined(HP_UX) && !defined(linux)
        if (getenv(USE_SVR4_MAKE) || getenv(NOCATGETS("USE_SVID"))){
           svr4 = true;
           posix = false;
        }
#endif

        /*
         * Find the dmake_compat_mode: posix, sun, svr4, or gnu_style, .
```

```
433          */
434         char * dmake_compat_mode_var = getenv(NOCATGETS("SUN_MAKE_COMPAT_MODE"))
435         if (dmake_compat_mode_var != NULL) {
436                 if (0 == strcasecmp(dmake_compat_mode_var, NOCATGETS("GNU"))) {
437                         gnu_style = true;
438                 }
439                 //svr4 = false;
440                 //posix = false;
441         }

443         /*
444          * Temporary directory set up.
445          */
446         char * tmpdir_var = getenv(NOCATGETS("TMPDIR"));
447         if (tmpdir_var != NULL && *tmpdir_var == '/' && strlen(tmpdir_var) < MAX
448                 strcpy(mbs_buffer, tmpdir_var);
449                 for (tmpdir_var = mbs_buffer+strlen(mbs_buffer);
450                         *(--tmpdir_var) == '/' && tmpdir_var > mbs_buffer;
451                         *tmpdir_var = '\0');
452                 if (strlen(mbs_buffer) + 32 < MAXPATHLEN) { /* 32 = strlen("/dma
453                         sprintf(mbs_buffer2, NOCATGETS("%s/dmake.tst.%d.XXXXXX")
454                                 mbs_buffer, getpid());
455                         int fd = mkstemp(mbs_buffer2);
456                         if (fd >= 0) {
457                                 close(fd);
458                                 unlink(mbs_buffer2);
459                                 tmpdir = strdup(mbs_buffer);
460                         }
461                 }
462         }

464 #ifndef PARALLEL
465         /* find out if stdout and stderr point to the same place */
466         if (fstat(1, &out_stat) < 0) {
467                 fatal(catgets(catd, 1, 165, "fstat of standard out failed: %s"),
468         }
469         if (fstat(2, &err_stat) < 0) {
470                 fatal(catgets(catd, 1, 166, "fstat of standard error failed: %s"
471         }
472         if ((out_stat.st_dev == err_stat.st_dev) &&
473             (out_stat.st_ino == err_stat.st_ino)) {
474                 stdout_stderr_same = true;
475         } else {
476                 stdout_stderr_same = false;
477         }
478 #else
479         stdout_stderr_same = false;
480 #endif
481         /* Make the vroot package scan the path using shell semantics */
482         set_path_style(0);

484         setup_char_semantics();

486         setup_for_projectdir();

488         /*
489          * If running with .KEEP_STATE, curdir will be set with
490          * the connected directory.
491          */
492 #if defined(SUN5_0) || defined(HP_UX) || defined(linux)
493         (void) atexit(cleanup_after_exit);
494 #else
495         (void) on_exit(cleanup_after_exit, (char *) NULL);
496 #endif

498         load_cached_names();
```



```
500 /*
501  *      Set command line flags
502  */
503         setup_makeflags_argv();
504         read_command_options(mf_argc, mf_argv);
505         read_command_options(argc, argv);
506         if (debug_level > 0) {
507                 cp = getenv(makeflags->string_mb);
508                 (void) printf(catgets(catd, 1, 167, "MAKEFLAGS value: %s\n"), cp
509         }

511         setup_interrupt(handle_interrupt);

513         read_files_and_state(argc, argv);

515 #ifdef TEAMWARE_MAKE_CMN
516         /*
517          * Find the dmake_output_mode: TXT1, TXT2 or HTML1.
518          */
519         MBSTOWCS(wcs_buffer, NOCATGETS("DMAKE_OUTPUT_MODE"));
520         dmake_name2 = GETNAME(wcs_buffer, FIND_LENGTH);
521         prop2 = get_prop(dmake_name2->prop, macro_prop);
522         if (prop2 == NULL) {
523                 /* DMAKE_OUTPUT_MODE not defined, default to TXT1 mode */
524                 output_mode = txt1_mode;
525         } else {
526                 dmake_value2 = prop2->body.macro.value;
527                 if ((dmake_value2 == NULL) ||
528                     (IS_EQUAL(dmake_value2->string_mb, NOCATGETS("TXT1")))) {
529                         output_mode = txt1_mode;
530                 } else if (IS_EQUAL(dmake_value2->string_mb, NOCATGETS("TXT2")))
531                         output_mode = txt2_mode;
532                 } else if (IS_EQUAL(dmake_value2->string_mb, NOCATGETS("HTML1"))
533                         output_mode = html1_mode;
534                 } else {
535                         warning(catgets(catd, 1, 352, "Unsupported value '%s' fo
536                               dmake_value2->string_mb);
537                 }
538         }
539         /*
540          * Find the dmake_mode: distributed, parallel, or serial.
541          */
542     if ((!pmake_cap_r_specified) &&
543         (!pmake_machinesfile_specified)) {
544         MBSTOWCS(wcs_buffer, NOCATGETS("DMAKE_MODE"));
545         dmake_name2 = GETNAME(wcs_buffer, FIND_LENGTH);
546         prop2 = get_prop(dmake_name2->prop, macro_prop);
547         if (prop2 == NULL) {
548                 /* DMAKE_MODE not defined, default to distributed mode */
549                 dmake_mode_type = distributed_mode;
550                 no_parallel = false;
551         } else {
552                 dmake_value2 = prop2->body.macro.value;
553                 if ((dmake_value2 == NULL) ||
554                     (IS_EQUAL(dmake_value2->string_mb, NOCATGETS("distributed"))
555                         dmake_mode_type = distributed_mode;
556                         no_parallel = false;
557                 } else if (IS_EQUAL(dmake_value2->string_mb, NOCATGETS("parallel
558                         dmake_mode_type = parallel_mode;
559                         no_parallel = false;
560 #ifdef SGE_SUPPORT
561                         grid = false;
562                 } else if (IS_EQUAL(dmake_value2->string_mb, NOCATGETS("grid")))
563                         dmake_mode_type = parallel_mode;
564                         no_parallel = false;
```

```
565                         grid = true;
566 #endif
567                 } else if (IS_EQUAL(dmake_value2->string_mb, NOCATGETS("serial")
568                         dmake_mode_type = serial_mode;
569                         no_parallel = true;
570                 } else {
571                         fatal(catgets(catd, 1, 307, "Unknown dmake mode argument
572                 }
573         }

575         if ((!list_all_targets) &&
576             (report_dependencies_level == 0)) {
577                 /*
578                  * Check to see if either DMAKE_RCFILE or DMAKE_MODE is defined.
579                  * They could be defined in the env, in the makefile, or on the
580                  * command line.
581                  * If neither is defined, and $(HOME)/.dmakerc does not exists,
582                  * then print a message, and default to parallel mode.
583                  */
584 #ifdef DISTRIBUTED
585                 MBSTOWCS(wcs_buffer, NOCATGETS("DMAKE_RCFILE"));
586                 dmake_name = GETNAME(wcs_buffer, FIND_LENGTH);
587                 MBSTOWCS(wcs_buffer, NOCATGETS("DMAKE_MODE"));
588                 dmake_name2 = GETNAME(wcs_buffer, FIND_LENGTH);
589                 if ((((prop = get_prop(dmake_name->prop, macro_prop)) == NULL) |
590                      ((dmake_value = prop->body.macro.value) == NULL)) &&
591                     (((prop2 = get_prop(dmake_name2->prop, macro_prop)) == NULL)
592                      ((dmake_value2 = prop2->body.macro.value) == NULL))) {
593                         Boolean empty_dmakerc = true;
594                         char *homedir = getenv(NOCATGETS("HOME"));
595                         if ((homedir != NULL) && (strlen(homedir) < (sizeof(def_
596                                 sprintf(def_dmakerc_path, NOCATGETS("%s/.dmakerc
597                                 if ((((statval = stat(def_dmakerc_path, &statbuf
598                                         ((statval == 0) && (statbuf.st_size == 0
599                                 } else {
600                                         Avo_dmakerc     *rcfile = new Avo_dmaker
601                                         Avo_err         *err = rcfile->read(def_
602                                         if (err) {
603                                                 fatal(err->str);
604                                         }
605                                         empty_dmakerc = rcfile->was_empty();
606                                         delete rcfile;
607                                 }
608                         }
609                         if (empty_dmakerc) {
610                                 if (getenv(NOCATGETS("DMAKE_DEF_PRINTED")) == NU
611                                         putenv(NOCATGETS("DMAKE_DEF_PRINTED=TRUE
612                                         (void) fprintf(stdout, catgets(catd, 1,
613                                         (void) fprintf(stdout, catgets(catd, 1,
614                                 }
615                                 dmake_mode_type = parallel_mode;
616                                 no_parallel = false;
617                         }
618                 }
619 #else
620                 if(dmake_mode_type == distributed_mode) {
621                         (void) fprintf(stdout, NOCATGETS("dmake: Distributed mod
622                         (void) fprintf(stdout, NOCATGETS("       Defaulting to p
623                         dmake_mode_type = parallel_mode;
624                         no_parallel = false;
625                 }
626 #endif  /* DISTRIBUTED */
627         }
628     }
629 #endif
```



```
631 #ifdef TEAMWARE_MAKE_CMN
632         parallel_flag = true;
633         /* XXX - This is a major hack for DMake/Licensing. */
634         if (getenv(NOCATGETS("DMAKE_CHILD")) == NULL) {
635                 if (!avo_cli_search_license(argv[0], dmake_exit_callback, TRUE,
636                         /*
637                          * If the user can not get a TeamWare license,
638                          * default to serial mode.
639                          */
640                         dmake_mode_type = serial_mode;
641                         no_parallel = true;
642                 } else {
643                         putenv(NOCATGETS("DMAKE_CHILD=TRUE"));
644                 }
645                 start_time = time(NULL);
646                 /*
647                  * XXX - Hack to disable SIGALRM's from licensing library's
648                  *       setitimer().
649                  */
650                 value.it_interval.tv_sec = 0;
651                 value.it_interval.tv_usec = 0;
652                 value.it_value.tv_sec = 0;
653                 value.it_value.tv_usec = 0;
654                 (void) setitimer(ITIMER_REAL, &value, NULL);
655         }

657 //
658 // If dmake is running with -t option, set dmake_mode_type to serial.
659 // This is done because doname() calls touch_command() that runs serially.
660 // If we do not do that, maketool will have problems.
661 //
662         if(touch) {
663                 dmake_mode_type = serial_mode;
664                 no_parallel = true;
665         }
666 #else
667         parallel_flag = false;
668 #endif

670 #if defined (TEAMWARE_MAKE_CMN) && defined(REDIRECT_ERR)
671         /*
672          * Check whether stdout and stderr are physically same.
673          * This is in order to decide whether we need to redirect
674          * stderr separately from stdout.
675          * This check is performed only if __DMAKE_SEPARATE_STDERR
676          * is not set. This variable may be used in order to preserve
677          * the 'old' behaviour.
678          */
679         out_err_same = true;
680         char * dmake_sep_var = getenv(NOCATGETS("__DMAKE_SEPARATE_STDERR"));
681         if (dmake_sep_var == NULL || (0 != strcasecmp(dmake_sep_var, NOCATGETS("
682                 struct stat stdout_stat;
683                 struct stat stderr_stat;
684                 if( (fstat(1, &stdout_stat) == 0)
685                  && (fstat(2, &stderr_stat) == 0) )
686                 {
687                         if( (stdout_stat.st_dev != stderr_stat.st_dev)
688                          || (stdout_stat.st_ino != stderr_stat.st_ino) )
689                         {
690                                 out_err_same = false;
691                         }
692                 }
693         }
694 #endif

696 #ifdef DISTRIBUTED
```

```
697         /*
698          * At this point, DMake should startup an rxm with any and all
699          * DMake command line options. Rxm will, among other things,
700          * read the rc file.
701          */
702         if ((!list_all_targets) &&
703             (report_dependencies_level == 0) &&
704             (dmake_mode_type == distributed_mode)) {
705                 startup_rxm();
706         }
707 #endif
708
709 /*
710  *      Enable interrupt handler for alarms
711  */
712         (void) bsd_signal(SIGALRM, (SIG_PF)doalarm);

714 /*
715  *      Check if make should report
716  */
717         if (getenv(sunpro_dependencies->string_mb) != NULL) {
718                 FILE    *report_file;

720                 report_dependency("");
721                 report_file = get_report_file();
722                 if ((report_file != NULL) && (report_file != (FILE*)-1)) {
723                         (void) fprintf(report_file, "\n");
724                 }
725         }

727 /*
728  *      Make sure SUNPRO_DEPENDENCIES is exported (or not) properly
729  *      and NSE_DEP.
730  */
731         if (keep_state) {
732                 maybe_append_prop(sunpro_dependencies, macro_prop)->
733                   body.macro.exported = true;
734 #ifdef NSE
735                 (void) setenv(NOCATGETS("NSE_DEP"), get_current_path());
736 #endif
737         } else {
738                 maybe_append_prop(sunpro_dependencies, macro_prop)->
739                   body.macro.exported = false;
740         }

742         working_on_targets = true;
743         if (trace_status) {
744                 dump_make_state();
745 #if defined(SUN5_0) || defined(HP_UX) || defined(linux)
746                 fclose(stdout);
747                 fclose(stderr);
748                 exit_status = 0;
749 #endif
750                 exit(0);
751         }
752         if (list_all_targets) {
753                 dump_target_list();
754 #if defined(SUN5_0) || defined(HP_UX) || defined(linux)
755                 fclose(stdout);
756                 fclose(stderr);
757                 exit_status = 0;
758 #endif
759                 exit(0);
760         }
761         trace_reader = false;
```



```
763         /*
764          * Set temp_file_directory to the directory the .make.state
765          * file is written to.
766          */
767         if ((slash_ptr = strrchr(make_state->string_mb, (int) slash_char)) == NU
768                 temp_file_directory = strdup(get_current_path());
769         } else {
770                 *slash_ptr = (int) nul_char;
771                 (void) strcpy(make_state_dir, make_state->string_mb);
772                 *slash_ptr = (int) slash_char;
773                    /* when there is only one slash and it's the first
774                    ** character, make_state_dir should point to '/'.
775                    */
776                 if(make_state_dir[0] == '\0') {
777                    make_state_dir[0] = '/';
778                    make_state_dir[1] = '\0';
779                 }
780                 if (make_state_dir[0] == (int) slash_char) {
781                         temp_file_directory = strdup(make_state_dir);
782                 } else {
783                         char    tmp_current_path2[MAXPATHLEN];
784
785                         (void) sprintf(tmp_current_path2,
786                                         "%s/%s",
787                                         get_current_path(),
788                                         make_state_dir);
789                         temp_file_directory = strdup(tmp_current_path2);
790                 }
791         }

793 #ifdef DISTRIBUTED
794         building_serial = false;
795 #endif

797         report_dir_enter_leave(true);

799         make_targets(argc, argv, parallel_flag);

801         report_dir_enter_leave(false);

803 #ifdef NSE
804         exit(nse_exit_status());
805 #else
806         if (build_failed_ever_seen) {
807 #if defined(SUN5_0) || defined(HP_UX) || defined(linux)
808                 if (posix) {
809                         exit_status = 1;
810                 }
811 #endif
812                 exit(1);
813         }
814 #if defined(SUN5_0) || defined(HP_UX) || defined(linux)
815         exit_status = 0;
816 #endif
817         exit(0);
818 #endif
819         /* NOTREACHED */
820 }
```

_____***unchanged_portion_omitted_***

```
1255 /*
1256  *      read_command_options(argc, argv)
1257  *
1258  *      Scan the cmd line options and process the ones that start with "-"
1259  *
```

```
 *      Return value:
 *                              -M argument, if any
 *
 *      Parameters:
 *              argc            You know what this is
 *              argv            You know what this is
 *
 *      Global variables used:
 */
static void
read_command_options(register int argc, register char **argv)
{
        register int            ch;
        int                     current_optind = 1;
        int                     last_optind_with_double_hyphen = 0;
        int                     last_optind;
        int                     last_current_optind;
        register int            i;
        register int            j;
        register int            k;
        register int            makefile_next = 0; /*
                                                    * flag to note options:
                                                    * -c, f, g, j, m, o
                                                    */
        const char              *tptr;
        const char              *CMD_OPTS;

        extern char             *optarg;
        extern int              optind, opterr, optopt;

#define SUNPRO_CMD_OPTS "-~Bbc:Ddef:g:ij:K:kM:m:NnO:o:PpqRrSsTtuVvwx:"

#ifdef TEAMWARE_MAKE_CMN
#       define SVR4_CMD_OPTS   "-c:ef:g:ij:km:nO:o:pqrsTtVv"
#else
#       define SVR4_CMD_OPTS   "-ef:iknpqrstV"
#endif

        /*
         * Added V in SVR4_CMD_OPTS also, which is going to be a hidden
         * option, just to make sure that the getopt doesn't fail when some
         * users leave their USE_SVR4_MAKE set and try to use the makefiles
         * that are designed to issue commands like $(MAKE) -V. Anyway it
         * sets the same flag but ensures that getopt doesn't fail.
         */

        opterr = 0;
        optind = 1;
        while (1) {
                last_optind=optind;                     /* Save optind and curre
                last_current_optind=current_optind;     /* in case we have to re
                if (svr4) {
                        CMD_OPTS=SVR4_CMD_OPTS;
                        ch = getopt(argc, argv, SVR4_CMD_OPTS);
                } else {
                        CMD_OPTS=SUNPRO_CMD_OPTS;
                        ch = getopt(argc, argv, SUNPRO_CMD_OPTS);
                }
                if (ch == EOF) {
                        if(optind < argc) {
                                /*
                                 * Fixing bug 4102537:
                                 *    Strange behaviour of command make using --
                                 * Not all argv have been processed
                                 * Skip non-flag argv and continue processing.
                                 */
```



```
                                optind++;
                                current_optind++;
                                continue;
                        } else {
                                break;
                        }
                }
                if (ch == '?') {
                        if (optopt == '-') {
                                /* Bug 5060758: getopt() changed behavior (s10_6
                                 * and now we have to deal with cases when optio
                                 * with double hyphen appear here, from -$(MAKEF
                                 */
                                i = current_optind;
                                if (argv[i][0] == '-') {
                                  if (argv[i][1] == '-') {
                                    if (argv[i][2] != '\0') {
                                      /* Check if this option is allowed */
                                      tptr = strchr(CMD_OPTS, argv[i][2]);
                                      if (tptr) {
                                        if (last_optind_with_double_hyphen != cu
                                          /* This is first time we are trying to
                                           * problem with this option. If we com
                                           * time, we will go to fatal error.
                                           */
                                          last_optind_with_double_hyphen = curre

                                          /* Eliminate first hyphen character */
                                          for (j=0; argv[i][j] != '\0'; j++) {
                                            argv[i][j] = argv[i][j+1];
                                          }

                                          /* Repeat the processing of this argum
                                          optind=last_optind;
                                          current_optind=last_current_optind;
                                          continue;
                                        }
                                      }
                                    }
                                  }
                                }
                        }
                }

                if (ch == '?') {
                        if (svr4) {
#ifdef TEAMWARE_MAKE_CMN
                                fprintf(stderr,
                                        catgets(catd, 1, 267, "Usage : dmake [ -
                                fprintf(stderr,
                                        catgets(catd, 1, 268, "              [ -
                                fprintf(stderr,
                                        catgets(catd, 1, 269, "              [ -
#else
                                fprintf(stderr,
                                        catgets(catd, 1, 270, "Usage : make [ -f
                                fprintf(stderr,
                                        catgets(catd, 1, 271, "             [ -s
#endif
                                tptr = strchr(SVR4_CMD_OPTS, optopt);
                        } else {
#ifdef TEAMWARE_MAKE_CMN
                                fprintf(stderr,
                                        catgets(catd, 1, 272, "Usage : dmake [ -
                                fprintf(stderr,
```

```
1392                                    catgets(catd, 1, 273, "              [ -
1393                            fprintf(stderr,
1394                                    catgets(catd, 1, 274, "              [ -
1395                            fprintf(stderr,
1396                                    catgets(catd, 1, 275, "              [ -
1397 #else
1398                            fprintf(stderr,
1399                                    catgets(catd, 1, 276, "Usage : make [ -f
1400                            fprintf(stderr,
1401                                    catgets(catd, 1, 277, "             [ -e
1402                            fprintf(stderr,
1403                                    catgets(catd, 1, 278, "             [ -u
1404 #endif
1405                            tptr = strchr(SUNPRO_CMD_OPTS, optopt);
1406                    }
1407                    if (!tptr) {
1408                            fatal(catgets(catd, 1, 279, "Unknown option `-%c
1409                    } else {
1410                            fatal(catgets(catd, 1, 280, "Missing argument af
1411                    }
1412            }

1414 #if defined(linux)
1415            if (ch == 1) {
1416                    if(optind < argc) {
1417                            //optind++;
1418                            //current_optind++;
1419                    makefile_next = 0;
1420                    current_optind = optind;
1421                            continue;
1422                    } else {
1423                            break;
1424                    }
1425            }
1426 #endif


1429            makefile_next |= parse_command_option(ch);
1430            /*
1431             * If we're done processing all of the options of
1432             * ONE argument string...
1433             */
1434            if (current_optind < optind) {
1435                    i = current_optind;
1436                    k = 0;
1437                    /* If there's an argument for an option... */
1438                    if ((optind - current_optind) > 1) {
1439                            k = i + 1;
1440                    }
1441                    switch (makefile_next) {
1442                    case 0:
1443                            argv[i] = NULL;
1444                            /* This shouldn't happen */
1445                            if (k) {
1446                                    argv[k] = NULL;
1447                            }
1448                            break;
1449                    case 1: /* -f seen */
1450                            argv[i] = (char *)NOCATGETS("-f");
1450                            argv[i] = NOCATGETS("-f");
1451                            break;
1452                    case 2: /* -c seen */
1453                            argv[i] = (char *)NOCATGETS("-c");
1453                            argv[i] = NOCATGETS("-c");
1454 #ifndef TEAMWARE_MAKE_CMN
1455                            warning(catgets(catd, 1, 281, "Ignoring Distribu
```



```
1456 #endif
1457                            break;
1458                    case 4: /* -g seen */
1459                            argv[i] = (char *)NOCATGETS("-g");
1459                            argv[i] = NOCATGETS("-g");
1460 #ifndef TEAMWARE_MAKE_CMN
1461                            warning(catgets(catd, 1, 282, "Ignoring Distribu
1462 #endif
1463                            break;
1464                    case 8: /* -j seen */
1465                            argv[i] = (char *)NOCATGETS("-j");
1465                            argv[i] = NOCATGETS("-j");
1466 #ifndef TEAMWARE_MAKE_CMN
1467                            warning(catgets(catd, 1, 283, "Ignoring Distribu
1468 #endif
1469                            break;
1470                    case 16: /* -M seen */
1471                            argv[i] = (char *)NOCATGETS("-M");
1471                            argv[i] = NOCATGETS("-M");
1472 #ifndef TEAMWARE_MAKE_CMN
1473                            warning(catgets(catd, 1, 284, "Ignoring Parallel
1474 #endif
1475                            break;
1476                    case 32: /* -m seen */
1477                            argv[i] = (char *)NOCATGETS("-m");
1477                            argv[i] = NOCATGETS("-m");
1478 #ifndef TEAMWARE_MAKE_CMN
1479                            warning(catgets(catd, 1, 285, "Ignoring Distribu
1480 #endif
1481                            break;
1482 #ifndef PARALLEL
1483                    case 128: /* -O seen */
1484                            argv[i] = (char *)NOCATGETS("-O");
1484                            argv[i] = NOCATGETS("-O");
1485                            break;
1486 #endif
1487                    case 256: /* -K seen */
1488                            argv[i] = (char *)NOCATGETS("-K");
1488                            argv[i] = NOCATGETS("-K");
1489                            break;
1490                    case 512:       /* -o seen */
1491                            argv[i] = (char *)NOCATGETS("-o");
1491                            argv[i] = NOCATGETS("-o");
1492 #ifndef TEAMWARE_MAKE_CMN
1493                            warning(catgets(catd, 1, 311, "Ignoring Distribu
1494 #endif
1495                            break;
1496                    case 1024: /* -x seen */
1497                            argv[i] = (char *)NOCATGETS("-x");
1497                            argv[i] = NOCATGETS("-x");
1498 #ifndef TEAMWARE_MAKE_CMN
1499                            warning(catgets(catd, 1, 353, "Ignoring Distribu
1500 #endif
1501                            break;
1502                    default: /* > 1 of -c, f, g, j, K, M, m, O, o, x seen */
1503                            fatal(catgets(catd, 1, 286, "Illegal command lin
1504                    }

1506                    makefile_next = 0;
1507                    current_optind = optind;
1508            }
1509    }
1510 }
______unchanged_portion_omitted_

1569 /*
```

```
1570  * Convert the MAKEFLAGS string value into a vector of char *, similar
1571  * to argv.
1572  */
1573 static void
1574 setup_makeflags_argv()
1575 {
1576         char            *cp;
1577         char            *cp1;
1578         char            *cp2;
1579         char            *cp3;
1580         char            *cp_orig;
1581         Boolean         add_hyphen;
1582         int             i;
1583         char            tmp_char;

1585         mf_argc = 1;
1586         cp = getenv(makeflags->string_mb);
1587         cp_orig = cp;

1589         if (cp) {
1590                 /*
1591                  * If new MAKEFLAGS format, no need to add hyphen.
1592                  * If old MAKEFLAGS format, add hyphen before flags.
1593                  */

1595                 if ((strchr(cp, (int) hyphen_char) != NULL) ||
1596                     (strchr(cp, (int) equal_char) != NULL)) {

1598                         /* New MAKEFLAGS format */

1600                         add_hyphen = false;
1601 #ifdef ADDFIX5060758
1602                         /* Check if MAKEFLAGS value begins with multiple
1603                          * hyphen characters, and remove all duplicates.
1604                          * Usually it happens when the next command is
1605                          * used: $(MAKE) -$(MAKEFLAGS)
1606                          * This is a workaround for BugID 5060758.
1607                          */
1608                         while (*cp) {
1609                                 if (*cp != (int) hyphen_char) {
1610                                         break;
1611                                 }
1612                                 cp++;
1613                                 if (*cp == (int) hyphen_char) {
1614                                         /* There are two hyphens. Skip one */
1615                                         cp_orig = cp;
1616                                         cp++;
1617                                 }
1618                                 if (!(*cp)) {
1619                                         /* There are hyphens only. Skip all */
1620                                         cp_orig = cp;
1621                                         break;
1622                                 }
1623                         }
1624 #endif
1625                 } else {

1627                         /* Old MAKEFLAGS format */

1629                         add_hyphen = true;
1630                 }
1631         }

1633         /* Find the number of arguments in MAKEFLAGS */
1634         while (cp && *cp) {
1635                 /* Skip white spaces */
```



```
1636                 while (cp && *cp && isspace(*cp)) {
1637                         cp++;
1638                 }
1639                 if (cp && *cp) {
1640                         /* Increment arg count */
1641                         mf_argc++;
1642                         /* Go to next white space */
1643                         while (cp && *cp && !isspace(*cp)) {
1644                                 if(*cp == (int) backslash_char) {
1645                                         cp++;
1646                                 }
1647                                 cp++;
1648                         }
1649                 }
1650         }
1651         /* Allocate memory for the new MAKEFLAGS argv */
1652         mf_argv = (char **) malloc((mf_argc + 1) * sizeof(char *));
1653         mf_argv[0] = (char *)NOCATGETS("MAKEFLAGS");
1653         mf_argv[0] = NOCATGETS("MAKEFLAGS");
1654         /*
1655          * Convert the MAKEFLAGS string value into a vector of char *,
1656          * similar to argv.
1657          */
1658         cp = cp_orig;
1659         for (i = 1; i < mf_argc; i++) {
1660                 /* Skip white spaces */
1661                 while (cp && *cp && isspace(*cp)) {
1662                         cp++;
1663                 }
1664                 if (cp && *cp) {
1665                         cp_orig = cp;
1666                         /* Go to next white space */
1667                         while (cp && *cp && !isspace(*cp)) {
1668                                 if(*cp == (int) backslash_char) {
1669                                         cp++;
1670                                 }
1671                                 cp++;
1672                         }
1673                         tmp_char = *cp;
1674                         *cp = (int) nul_char;
1675                         if (add_hyphen) {
1676                                 mf_argv[i] = getmem(2 + strlen(cp_orig));
1677                                 mf_argv[i][0] = '\0';
1678                                 (void) strcat(mf_argv[i], "-");
1679                                 // (void) strcat(mf_argv[i], cp_orig);
1680                                 unquote_str(cp_orig, mf_argv[i]+1);
1681                         } else {
1682                                 mf_argv[i] = getmem(2 + strlen(cp_orig));
1683                                 //mf_argv[i] = strdup(cp_orig);
1684                                 unquote_str(cp_orig, mf_argv[i]);
1685                         }
1686                         *cp = tmp_char;
1687                 }
1688         }
1689         mf_argv[i] = NULL;
1690 }
______unchanged_portion_omitted_

3017 /*
3018  * Append the DMake option and value to the MAKEFLAGS string.
3019  */
3020 static void
3021 append_makeflags_string(Name name, register String makeflags_string)
3022 {
3023         const char      *option;
3023         char            *option;
```

```
3025         if (strcmp(name->string_mb, NOCATGETS("DMAKE_GROUP")) == 0) {
3026                 option = NOCATGETS(" -g ");
3027         } else if (strcmp(name->string_mb, NOCATGETS("DMAKE_MAX_JOBS")) == 0) {
3028                 option = NOCATGETS(" -j ");
3029         } else if (strcmp(name->string_mb, NOCATGETS("DMAKE_MODE")) == 0) {
3030                 option = NOCATGETS(" -m ");
3031         } else if (strcmp(name->string_mb, NOCATGETS("DMAKE_ODIR")) == 0) {
3032                 option = NOCATGETS(" -o ");
3033         } else if (strcmp(name->string_mb, NOCATGETS("DMAKE_RCFILE")) == 0) {
3034                 option = NOCATGETS(" -c ");
3035         } else if (strcmp(name->string_mb, NOCATGETS("PMAKE_MACHINESFILE")) == 0
3036                 option = NOCATGETS(" -M ");
3037         } else if (strcmp(name->string_mb, NOCATGETS("DMAKE_OUTPUT_MODE")) == 0)
3038                 option = NOCATGETS(" -x DMAKE_OUTPUT_MODE=");
3039         } else if (strcmp(name->string_mb, NOCATGETS("SUN_MAKE_COMPAT_MODE")) ==
3040                 option = NOCATGETS(" -x SUN_MAKE_COMPAT_MODE=");
3041         } else {
3042                 fatal(catgets(catd, 1, 289, "Internal error: name not recognized
3043         }
3044         Property prop = maybe_append_prop(name, macro_prop);
3045         if( prop == 0 || prop->body.macro.value == 0 ||
3046             prop->body.macro.value->string_mb == 0 ) {
3047                 return;
3048         }
3049         char mbs_value[MAXPATHLEN + 100];
3050         strcpy(mbs_value, option);
3051         strcat(mbs_value, prop->body.macro.value->string_mb);
3052         MBSTOWCS(wcs_buffer, mbs_value);
3053         append_string(wcs_buffer, makeflags_string, FIND_LENGTH);
3054 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   26901 Wed May 20 11:22:25 2015
new/usr/src/cmd/make/bin/misc.cc
make: fix GCC warnings
**********************************************************
______unchanged_portion_omitted_

 859 /* resolve - check for specified file in specified directory
 860  *      sets up dir, following symlinks.
 861  *      returns zero for success, or
 862  *      -1 for error (with errno set properly)
 863  */
 864 static int
 865 resolve (const char *indir,     /* search directory */
 866          const char *cmd,       /* search for name */
 865 resolve (char   *indir, /* search directory */
 866          char   *cmd,   /* search for name */
 867          char   *dir,   /* directory buffer */
 868          char   **run)  /* resultion name ptr ptr */
 869 {
 870     char               *p;
 871     int                 rv = -1;
 872     int                 sll;
 873     char                symlink[MAXPATHLEN + 1];

 875     do {
 876         errno = ENAMETOOLONG;
 877         if ((strlen (indir) + strlen (cmd) + 2) > (size_t) MAXPATHLEN)
 878             break;

 880         sprintf(dir, "%s/%s", indir, cmd);
 881         if (check_if_exec(dir) != 0)  /* check if dir is an executable */
 882         {
 883                 break;          /* Not an executable program */
 884         }

 886         /* follow symbolic links */
 887         while ((sll = readlink (dir, symlink, MAXPATHLEN)) >= 0) {
 888             symlink[sll] = 0;
 889             if (*symlink == '/')
 890                 strcpy (dir, symlink);
 891             else
 892                 sprintf (strrchr (dir, '/'), "/%s", symlink);
 893         }
 894         if (errno != EINVAL)
 895             break;

 897         p = strrchr (dir, '/');
 898         *p++ = 0;
 899         if (run)                /* user wants resolution name */
 900             *run = p;
 901         rv = 0;                 /* complete, with success! */

 903     } while (0);

 905     return rv;
 906 }
______unchanged_portion_omitted_
```

```
**********************************************************
   58611 Wed May 20 11:22:25 2015
new/usr/src/cmd/make/bin/read.cc
make: fix GCC warnings
**********************************************************
```

```
/*
 * 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 <avo/avo_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() */

#if defined(HP_UX) || defined(linux)
#include <avo/types.h>
extern "C" Avo_err *avo_find_run_dir(char **dirp);
#endif

/*
 * 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;


#if defined(HP_UX) || defined(linux)
        Avo_err                 *findrundir_err;
        char                    *run_dir, makerules_dir[BUFSIZ];
#endif

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

#ifdef NSE
```

```
128         if (report_file){
129                 wscpy(previous_current_makefile, current_makefile);
130                 wscpy(current_makefile, wcb);
131         }
132 #endif
133         if (max_include_depth++ >= 40) {
134                 fatal(catgets(catd, 1, 66, "Too many nested include statements")
135         }
136         if (makefile->body.makefile.contents != NULL) {
137                 retmem(makefile->body.makefile.contents);
138         }
139         source->inp_buf =
140           source->inp_buf_ptr =
141             source->inp_buf_end = NULL;
142         source->error_converting = false;
143         makefile->body.makefile.contents = NULL;
144         makefile->body.makefile.size = 0;
145         if ((makefile_name->hash.length != 1) ||
146             (wcb[0] != (int) hyphen_char)) {
147                 if ((makefile->body.makefile.contents == NULL) &&
148                     (doname_it)) {
149                         if (makefile_path == NULL) {
150                                 add_dir_to_path(".",
151                                                 &makefile_path,
152                                                 -1);
153 #ifdef SUN5_0
154                                 add_dir_to_path(NOCATGETS("/usr/share/lib/make")
155                                                 &makefile_path,
156                                                 -1);
157                                 add_dir_to_path(NOCATGETS("/etc/default"),
158                                                 &makefile_path,
159                                                 -1);
160 #elif defined(HP_UX)
161                                 findrundir_err = avo_find_run_dir(&run_dir);
162                                 if (! findrundir_err) {
163                                         (void) sprintf(makerules_dir, NOCATGETS(
164                                         add_dir_to_path(makerules_dir,
165                                                         &makefile_path,
166                                                         -1);
167                                 }

169                                 add_dir_to_path(NOCATGETS("/opt/SUNWspro/share/l
170                                                 &makefile_path,
171                                                 -1);
172                                 add_dir_to_path(NOCATGETS("/usr/share/lib/make")
173                                                 &makefile_path,
174                                                 -1);
175 #elif defined(linux)
176                                 findrundir_err = avo_find_run_dir(&run_dir);
177                                 if (! findrundir_err) {
178                                         (void) sprintf(makerules_dir, NOCATGETS(
179                                         add_dir_to_path(makerules_dir,
180                                                         &makefile_path,
181                                                         -1);
182                                 }

184                                 add_dir_to_path(NOCATGETS("/usr/SUNWspro/lib"),
185                                                 &makefile_path,
186                                                 -1);
187                                 add_dir_to_path(NOCATGETS("/opt/SUNWspro/share/l
188                                                 &makefile_path,
189                                                 -1);
190                                 add_dir_to_path(NOCATGETS("/usr/share/lib/make")
191                                                 &makefile_path,
192                                                 -1);
193 #else
```



```
194                                 add_dir_to_path(NOCATGETS("/usr/include/make"),
195                                                 &makefile_path,
196                                                 -1);
197 #endif
198                         }
199                         save_makefile_type = makefile_type;
200                         makefile_type = reading_nothing;
201                         if (doname(makefile_name, true, false) == build_dont_kno
202                                 /* Try normalized filename */
203                                 string_start=get_wstring(makefile_name->string_m
204                                 for (string_end=string_start+1; *string_end != L
204                                 for (string_end=string_start+1; *string_end != N
205                                 normalized_makefile_name=normalize_name(string_s
206                                 if ((strcmp(makefile_name->string_mb, normalized
207                                         (doname(normalized_makefile_name, true,
208                                         n = access_vroot(makefile_name->string_m
209                                                  4,
210                                                  chase_path ?
211                                                  makefile_path : NULL,
212                                                  VROOT_DEFAULT);
213                                         if (n == 0) {
214                                                 get_vroot_path((char **) NULL,
215                                                       &path,
216                                                       (char **) NULL);
217                                                 if ((path[0] == (int) period_cha
218                                                     (path[1] == (int) slash_char
219                                                          path += 2;
220                                                 }
221                                                 MBSTOWCS(wcs_buffer, path);
222                                                 makefile_name = GETNAME(wcs_buff
223                                                                 FIND_LENGTH);
224                                         }
225                                 }
226                                 retmem(string_start);
227                                 /*
228                                  * Commented out: retmem_mb(normalized_makefile_
229                                  * We have to return this memory, but it seems t
230                                  * in dmake or in Sun C++ 5.7 compiler (it works
231                                  * is compiled using Sun C++ 5.6).
232                                  */
233                                 // retmem_mb(normalized_makefile_name->string_mb
234                         }
235                         makefile_type = save_makefile_type;
236                 }
237                 source->string.free_after_use = false;
238                 source->previous = NULL;
239                 source->already_expanded = false;
240                 /* Lock the file for read, but not when -n. */
241                 if (lock_makefile &&
242                     !do_not_exec_rule) {

244                          make_state_lockfile = getmem(strlen(make_state->string_
245                          (void) sprintf(make_state_lockfile,
246                                         NOCATGETS("%s.lock"),
247                                         make_state->string_mb);
248                         (void) file_lock(make_state->string_mb,
249                                          make_state_lockfile,
250                                          (int *) &make_state_locked,
251                                          0);
252                         if(!make_state_locked) {
253                                 printf(NOCATGETS("-- NO LOCKING for read\n"));
254                                 retmem_mb(make_state_lockfile);
255                                 make_state_lockfile = 0;
256                                 return failed;
257                         }
258                 }
```

```
259                 if (makefile->body.makefile.contents == NULL) {
260                         save_makefile_type = makefile_type;
261                         makefile_type = reading_nothing;
262                         if ((doname_it) &&
263                             (doname(makefile_name, true, false) == build_failed)
264                                 if (complain) {
265                                         (void) fprintf(stderr,
266 #ifdef DISTRIBUTED
267                                                        catgets(catd, 1, 67, "dma
268 #else
269                                                        catgets(catd, 1, 237, "ma
270 #endif
271                                                        makefile_name->string_mb)
272                                 }
273                                 max_include_depth--;
274                                 makefile_type = save_makefile_type;
275                                 return failed;
276                         }
277                         makefile_type = save_makefile_type;
278                         //
279                         // Before calling exists() make sure that we have the ri
280                         //
281                         makefile_name->stat.time = file_no_time;
282
283                         if (exists(makefile_name) == file_doesnt_exist) {
284                                 if (complain ||
285                                     (makefile_name->stat.stat_errno != ENOENT))
286                                         if (must_exist) {
287                                                 fatal(catgets(catd, 1, 68, "Can'
288                                                       makefile_name->string_mb,
289                                                       errmsg(makefile_name->
290                                                              stat.stat_errno));
291                                         } else {
292                                                 warning(catgets(catd, 1, 69, "Ca
293                                                         makefile_name->string_mb
294                                                         errmsg(makefile_name->
295                                                                stat.stat_errno))
296                                         }
297                                 }
298                                 max_include_depth--;
299                                 if(make_state_locked && (make_state_lockfile !=
300                                         (void) unlink(make_state_lockfile);
301                                         retmem_mb(make_state_lockfile);
302                                         make_state_lockfile = NULL;
303                                         make_state_locked = false;
304                                 }
305                                 retmem(wcb);
306                                 retmem_mb((char *)source);
307                                 return failed;
308                         }
309                         /*
310                          * These values are the size and bytes of
311                          * the MULTI-BYTE makefile.
312                          */
313                         orig_makefile->body.makefile.size =
314                           makefile->body.makefile.size =
315                             source->bytes_left_in_file =
316                               makefile_name->stat.size;
317                         if (report_file) {
318                                 for (dpp = &makefiles_used;
319                                      *dpp != NULL;
320                                      dpp = &(*dpp)->next);
321                                 dp = ALLOC(Dependency);
322                                 dp->next = NULL;
323                                 dp->name = makefile_name;
324                                 dp->automatic = false;
```



```
325                                 dp->stale = false;
326                                 dp->built = false;
327                                 *dpp = dp;
328                         }
329                         source->fd = open_vroot(makefile_name->string_mb,
330                                                 O_RDONLY,
331                                                 0,
332                                                 NULL,
333                                                 VROOT_DEFAULT);
334                         if (source->fd < 0) {
335                                 if (complain || (errno != ENOENT)) {
336                                         if (must_exist) {
337                                                 fatal(catgets(catd, 1, 70, "Can'
338                                                       makefile_name->string_mb,
339                                                       errmsg(errno));
340                                         } else {
341                                                 warning(catgets(catd, 1, 71, "Ca
342                                                         makefile_name->string_mb
343                                                         errmsg(errno));
344                                         }
345                                 }
346                                 max_include_depth--;
347                                 return failed;
348                         }
349                         (void) fcntl(source->fd, F_SETFD, 1);
350                         orig_makefile->body.makefile.contents =
351                           makefile->body.makefile.contents =
352                             source->string.text.p =
353                               source->string.buffer.start =
354                                 ALLOC_WC((int) (makefile_name->stat.size + 2));
355                         if (makefile_type == reading_cpp_file) {
356                                 forget_after_parse = true;
357                         }
358                         source->string.text.end = source->string.text.p;
359                         source->string.buffer.end =
360                           source->string.text.p + makefile_name->stat.size;
361                 } else {
362                         /* Do we ever reach here? */
363                         source->fd = -1;
364                         source->string.text.p =
365                           source->string.buffer.start =
366                             makefile->body.makefile.contents;
367                         source->string.text.end =
368                           source->string.buffer.end =
369                             source->string.text.p + makefile->body.makefile.size
370                         source->bytes_left_in_file =
371                           makefile->body.makefile.size;
372                 }
373                 file_being_read = wcb;
374         } else {
375                 char            *stdin_text_p;
376                 char            *stdin_text_end;
377                 char            *stdin_buffer_start;
378                 char            *stdin_buffer_end;
379                 char            *p_mb;
380                 int             num_mb_chars;
381                 size_t          num_wc_chars;
382
383                 MBSTOWCS(wcs_buffer, NOCATGETS("Standard in"));
384                 makefile_name = GETNAME(wcs_buffer, FIND_LENGTH);
385                 /*
386                  * Memory to read standard in, then convert it
387                  * to wide char strings.
388                  */
389                 stdin_buffer_start =
390                   stdin_text_p = getmem(length = 1024);
```

```
391                 stdin_buffer_end = stdin_text_p + length;
392                 MBSTOWCS(wcs_buffer, NOCATGETS("standard input"));
393                 file_being_read = (wchar_t *) wsdup(wcs_buffer);
394                 line_number = 0;
395                 while ((n = read(fileno(stdin),
396                                  stdin_text_p,
397                                  length)) > 0) {
398                         length -= n;
399                         stdin_text_p += n;
400                         if (length == 0) {
401                                 p_mb = getmem(length = 1024 +
402                                               (stdin_buffer_end -
403                                                stdin_buffer_start));
404                                 (void) strncpy(p_mb,
405                                                stdin_buffer_start,
406                                                (stdin_buffer_end -
407                                                 stdin_buffer_start));
408                                 retmem_mb(stdin_buffer_start);
409                                 stdin_text_p = p_mb +
410                                   (stdin_buffer_end - stdin_buffer_start);
411                                 stdin_buffer_start = p_mb;
412                                 stdin_buffer_end =
413                                   stdin_buffer_start + length;
414                                 length = 1024;
415                         }
416                 }
417                 if (n < 0) {
418                         fatal(catgets(catd, 1, 72, "Error reading standard input
419                               errmsg(errno));
420                 }
421                 stdin_text_p = stdin_buffer_start;
422                 stdin_text_end = stdin_buffer_end - length;
423                 num_mb_chars = stdin_text_end - stdin_text_p;

425                 /*
426                  * Now, convert the sequence of multibyte chars into
427                  * a sequence of corresponding wide character codes.
428                  */
429                 source->string.free_after_use = false;
430                 source->previous = NULL;
431                 source->bytes_left_in_file = 0;
432                 source->fd = -1;
433                 source->already_expanded = false;
434                 source->string.buffer.start =
435                   source->string.text.p = ALLOC_WC(num_mb_chars + 1);
436                 source->string.buffer.end =
437                     source->string.text.p + num_mb_chars;
438                 num_wc_chars = mbstowcs(source->string.text.p,
439                                         stdin_text_p,
440                                         num_mb_chars);
441                 if ((int) num_wc_chars >= 0) {
442                         source->string.text.end =
443                           source->string.text.p + num_wc_chars;
444                 }
445                 (void) retmem_mb(stdin_text_p);
446         }
447         line_number = 1;
448         if (trace_reader) {
449                 (void) printf(catgets(catd, 1, 73, ">>>>>>>>>>>>>>>> Reading mak
450                               makefile_name->string_mb);
451         }
452         parse_makefile(makefile_name, source);
453         if (trace_reader) {
454                 (void) printf(catgets(catd, 1, 74, ">>>>>>>>>>>>>>>> End of make
455                               makefile_name->string_mb);
456         }
```



```
457 #ifdef NSE
458         if (report_file && (previous_current_makefile[0] != NULL)) {
459                 wscpy(current_makefile, previous_current_makefile);
460         }
461 #endif
462         if(file_being_read) {
463                 retmem(file_being_read);
464         }
465         file_being_read = previous_file_being_read;
466         line_number = previous_line_number;
467         makefile_type = reading_nothing;
468         max_include_depth--;
469         if (make_state_locked) {
470                 /* Unlock .make.state. */
471                 unlink(make_state_lockfile);
472                 make_state_locked = false;
473                 retmem_mb(make_state_lockfile);
474         }
475         if (forget_after_parse) {
476                 retmem(makefile->body.makefile.contents);
477                 makefile->body.makefile.contents = NULL;
478         }
479         retmem_mb((char *)source);
480         return succeeded;
481 }
```

_____***unchanged_portion_omitted_***

****
16569 Wed May 20 11:22:26 2015
new/usr/src/cmd/make/include/mk/defs.h
make: fix GCC warnings
****
_____***unchanged_portion_omitted_***

```
 179 /*
 180  * Typedefs for all structs
 181  */
 182 typedef struct _Cmd_line        *Cmd_line, Cmd_line_rec;
 183 typedef struct _Dependency      *Dependency, Dependency_rec;
 184 typedef struct _Macro           *Macro, Macro_rec;
 185 typedef struct _Name_vector     *Name_vector, Name_vector_rec;
 186 typedef struct _Percent         *Percent, Percent_rec;
 187 typedef struct _Dyntarget       *Dyntarget;
 188 typedef struct _Recursive_make  *Recursive_make, Recursive_make_rec;
 189 typedef struct _Running         *Running, Running_rec;


 192 /*
 193  *      extern declarations for all global variables.
 194  *      The actual declarations are in globals.cc
 195  */
 196 extern  Boolean         allrules_read;
 197 extern  Name            posix_name;
 198 extern  Name            svr4_name;
 199 extern  Boolean         sdot_target;
 200 extern  Boolean         all_parallel;
 201 extern  Boolean         assign_done;
 202 extern  Boolean         build_failed_seen;
 203 #ifdef DISTRIBUTED
 204 extern  Boolean         building_serial;
 205 #endif
 206 extern  Name            built_last_make_run;
 207 extern  Name            c_at;
 208 #ifdef DISTRIBUTED
 209 extern  Boolean         called_make;
 210 #endif
 211 extern  Boolean         command_changed;
 212 extern  Boolean         commands_done;
 213 extern  Chain           conditional_targets;
 214 extern  Name            conditionals;
 215 extern  Boolean         continue_after_error;
 216 extern  Property        current_line;
 217 extern  Name            current_make_version;
 218 extern  Name            current_target;
 219 extern  short           debug_level;
 220 extern  Cmd_line        default_rule;
 221 extern  Name            default_rule_name;
 222 extern  Name            default_target_to_build;
 223 extern  Boolean         depinfo_already_read;
 224 extern  Name            dmake_group;
 225 extern  Name            dmake_max_jobs;
 226 extern  Name            dmake_mode;
 227 extern  DMake_mode      dmake_mode_type;
 228 extern  Name            dmake_output_mode;
 229 extern  DMake_output_mode       output_mode;
 230 extern  Name            dmake_odir;
 231 extern  Name            dmake_rcfile;
 232 extern  Name            done;
 233 extern  Name            dot;
 234 extern  Name            dot_keep_state;
 235 extern  Name            dot_keep_state_file;
 236 extern  Name            empty_name;
```



```
 237 extern  Boolean         fatal_in_progress;
 238 extern  int             file_number;
 239 extern  Name            force;
 240 extern  Name            ignore_name;
 241 extern  Boolean         ignore_errors;
 242 extern  Boolean         ignore_errors_all;
 243 extern  Name            init;
 244 extern  int             job_msg_id;
 245 extern  Boolean         keep_state;
 246 extern  Name            make_state;
 247 #ifdef TEAMWARE_MAKE_CMN
 248 extern  timestruc_t     make_state_before;
 249 #endif
 250 extern  Boolean         make_state_locked;
 251 extern  Dependency      makefiles_used;
 252 extern  Name            makeflags;
 253 extern  Name            make_version;
 254 extern  char            mbs_buffer2[];
 255 extern  char            *mbs_ptr;
 256 extern  char            *mbs_ptr2;
 257 extern  Boolean         no_action_was_taken;
 258 extern  int             mtool_msgs_fd;
 259 extern  Boolean         no_parallel;
 260 #ifdef SGE_SUPPORT
 261 extern  Boolean         grid;
 262 #endif
 263 extern  Name            no_parallel_name;
 264 extern  Name            not_auto;
 265 extern  Boolean         only_parallel;
 266 extern  Boolean         parallel;
 267 extern  Name            parallel_name;
 268 extern  Name            localhost_name;
 269 extern  int             parallel_process_cnt;
 270 extern  Percent         percent_list;
 271 extern  Dyntarget       dyntarget_list;
 272 extern  Name            plus;
 273 extern  Name            pmake_machinesfile;
 274 extern  Name            precious;
 275 extern  Name            primary_makefile;
 276 extern  Boolean         quest;
 277 extern  short           read_trace_level;
 278 extern  Boolean         reading_dependencies;
 279 extern  int             recursion_level;
 280 extern  Name            recursive_name;
 281 extern  short           report_dependencies_level;
 282 extern  Boolean         report_pwd;
 283 extern  Boolean         rewrite_statefile;
 284 extern  Running         running_list;
 285 extern  char            *sccs_dir_path;
 286 extern  Name            sccs_get_name;
 287 extern  Name            sccs_get_posix_name;
 288 extern  Cmd_line        sccs_get_rule;
 289 extern  Cmd_line        sccs_get_org_rule;
 290 extern  Cmd_line        sccs_get_posix_rule;
 291 extern  Name            get_name;
 292 extern  Name            get_posix_name;
 293 extern  Cmd_line        get_rule;
 294 extern  Cmd_line        get_posix_rule;
 295 extern  Boolean         send_mtool_msgs;
 296 extern  Boolean         all_precious;
 297 extern  Boolean         report_cwd;
 298 extern  Boolean         silent_all;
 299 extern  Boolean         silent;
 300 extern  Name            silent_name;
 301 extern  char            *stderr_file;
 302 extern  char            *stdout_file;
```

```
303 #ifdef SGE_SUPPORT
304 extern  char            script_file[];
305 #endif
306 extern  Boolean         stdout_stderr_same;
307 extern  Dependency      suffixes;
308 extern  Name            suffixes_name;
309 extern  Name            sunpro_dependencies;
310 extern  Boolean         target_variants;
311 extern  const char      *tmpdir;
312 extern  const char      *temp_file_directory;
311 extern  char            *tmpdir;
312 extern  char            *temp_file_directory;
313 extern  Name            temp_file_name;
314 extern  short           temp_file_number;
315 extern  wchar_t         *top_level_target;
316 extern  Boolean         touch;
317 extern  Boolean         trace_reader;
318 extern  Boolean         build_unconditional;
319 extern  pathpt          vroot_path;
320 extern  Name            wait_name;
321 extern  wchar_t         wcs_buffer2[];
322 extern  wchar_t         *wcs_ptr;
323 extern  wchar_t         *wcs_ptr2;
324 extern  nl_catd         catd;
325 extern  long int        hostid;

327 /*
328  * Declarations of system defined variables
329  */
330 #if !defined(linux)
331 /* On linux this variable is defined in 'signal.h' */
332 extern  char            *sys_siglist[];
333 #endif

335 /*
336  * Declarations of system supplied functions
337  */
338 extern  int             file_lock(char *, char *, int *, int);

340 /*
341  * Declarations of functions declared and used by make
342  */
343 extern  void            add_pending(Name target, int recursion_level, Boolean do
344 extern  void            add_running(Name target, Name true_target, Property comm
345 extern  void            add_serial(Name target, int recursion_level, Boolean do_
346 extern  void            add_subtree(Name target, int recursion_level, Boolean do
347 extern  void            append_or_replace_macro_in_dyn_array(ASCII_Dyn_Array *Ar
348 #ifdef DISTRIBUTED
349 extern  Doname          await_dist(Boolean waitflg);
350 #endif
351 #ifdef TEAMWARE_MAKE_CMN
352 extern  void            await_parallel(Boolean waitflg);
353 #endif
354 extern  void            build_suffix_list(Name target_suffix);
355 extern  Boolean         check_auto_dependencies(Name target, int auto_count, Nam
356 extern  void            check_state(Name temp_file_name);
357 extern  void            cond_macros_into_string(Name np, String_rec *buffer);
358 extern  void            construct_target_string();
359 extern  void            create_xdrs_ptr(void);
360 extern  void            depvar_add_to_list (Name name, Boolean cmdline);
361 #ifdef DISTRIBUTED
362 extern  void            distribute_rxm(Avo_DoJobMsg *dmake_job_msg);
363 extern  int             getRxmMessage(void);
364 extern  Avo_JobResultMsg* getJobResultMsg(void);
365 extern  Avo_AcknowledgeMsg* getAcknowledgeMsg(void);
366 #endif
```



```
367 extern  Doname          doname(register Name target, register Boolean do_get, re
368 extern  Doname          doname_check(register Name target, register Boolean do_g
369 extern  Doname          doname_parallel(Name target, Boolean do_get, Boolean imp
370 extern  Doname          dosys(register Name command, register Boolean ignore_err
371 extern  void            dump_make_state(void);
372 extern  void            dump_target_list(void);
373 extern  void            enter_conditional(register Name target, Name name, Name
374 extern  void            enter_dependencies(register Name target, Chain target_gr
375 extern  void            enter_dependency(Property line, register Name depe, Bool
376 extern  void            enter_equal(Name name, Name value, register Boolean appe
377 extern  Percent         enter_percent(register Name target, Chain target_group,
378 extern  Dyntarget       enter_dyntarget(register Name target);
379 extern  Name_vector     enter_name(String string, Boolean tail_present, register
380 extern  Boolean         exec_vp(register char *name, register char **argv, char
381 extern  Doname          execute_parallel(Property line, Boolean waitflg, Boolean
382 extern  Doname          execute_serial(Property line);
383 extern  timestruc_t&    exists(register Name target);
384 extern  void            fatal(char *, ...);
385 extern  void            fatal_reader(char *, ...);
386 extern  Doname          find_ar_suffix_rule(register Name target, Name true_targ
387 extern  Doname          find_double_suffix_rule(register Name target, Property *
388 extern  Doname          find_percent_rule(register Name target, Property *comman
389 extern  int             find_run_directory (char *cmd, char *cwd, char *dir, cha
390 extern  Doname          find_suffix_rule(Name target, Name target_body, Name tar
391 extern  Chain           find_target_groups(register Name_vector target_list, reg
392 extern  void            finish_children(Boolean docheck);
393 extern  void            finish_running(void);
394 extern  void            free_chain(Name_vector ptr);
395 extern  void            gather_recursive_deps(void);
396 extern  char            *get_current_path(void);
397 extern  int             get_job_msg_id(void);
398 extern  FILE            *get_mtool_msgs_fp(void);
399 #ifdef DISTRIBUTED
400 extern  Boolean         get_dmake_group_specified(void);
401 extern  Boolean         get_dmake_max_jobs_specified(void);
402 extern  Boolean         get_dmake_mode_specified(void);
403 extern  Boolean         get_dmake_odir_specified(void);
404 extern  Boolean         get_dmake_rcfile_specified(void);
405 extern  Boolean         get_pmake_machinesfile_specified(void);
406 #endif
407 #if defined(DISTRIBUTED) || defined(MAKETOOL) /* tolik */
408 extern  XDR             *get_xdrs_ptr(void);
409 #endif
410 extern  wchar_t         *getmem_wc(register int size);
411 #if !defined(linux)
412 /* On linux getwd(char *) is defined in 'unistd.h' */
413 #ifdef __cplusplus
414 extern "C" {
415 #endif
416 extern  char            *getwd(char *);
417 #ifdef __cplusplus
418 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
    2251 Wed May 20 11:22:26 2015
new/usr/src/cmd/make/include/mksh/misc.h
make: fix GCC warnings
**********************************************************
```

```
   1 #ifndef _MKSH_MISC_H
   2 #define _MKSH_MISC_H
   3 /*
   4  * CDDL HEADER START
   5  *
   6  * The contents of this file are subject to the terms of the
   7  * Common Development and Distribution License (the "License").
   8  * You may not use this file except in compliance with the License.
   9  *
  10  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
  11  * or http://www.opensolaris.org/os/licensing.
  12  * See the License for the specific language governing permissions
  13  * and limitations under the License.
  14  *
  15  * When distributing Covered Code, include this CDDL HEADER in each
  16  * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  17  * If applicable, add the following below this CDDL HEADER, with the
  18  * fields enclosed by brackets "[]" replaced with your own identifying
  19  * information: Portions Copyright [yyyy] [name of copyright owner]
  20  *
  21  * CDDL HEADER END
  22  */
  23 /*
  24  * Copyright 2004 Sun Microsystems, Inc. All rights reserved.
  25  * Use is subject to license terms.
  26  */

  28 #include <mksh/defs.h>

  30 extern void     append_char(wchar_t from, register String to);
  31 extern Property append_prop(register Name target, register Property_id type);
  32 extern void     append_string(register wchar_t *from, register String to, regist
  33 extern void     enable_interrupt(register void (*handler) (int));
  34 extern char     *errmsg(int errnum);
  35 extern void     fatal_mksh(const char *message, ...);
  36 extern void     fatal_reader_mksh(const char *pattern, ...);
  35 extern void     fatal_mksh(char * message, ...);
  36 extern void     fatal_reader_mksh(char * pattern, ...);
  37 extern char     *get_current_path_mksh(void);
  38 extern Property get_prop(register Property start, register Property_id type);
  39 extern char     *getmem(register int size);
  40 extern Name     getname_fn(wchar_t *name, register int len, register Boolean don
  41 extern void     store_name(Name name);
  42 extern void     free_name(Name name);
  43 extern void     handle_interrupt_mksh(int);
  44 extern Property maybe_append_prop(register Name target, register Property_id typ
  45 extern void     retmem(wchar_t *p);
  46 extern void     retmem_mb(caddr_t p);
  47 extern void     setup_char_semantics(void);
  48 extern void     setup_interrupt(register void (*handler) (int));
  49 extern void     warning_mksh(char * message, ...);

  51 extern void     append_string(register char *from, register String to, register
  52 extern wchar_t  *get_wstring(char * from);


  55 #endif
```

```
**********************************************************
    1755 Wed May 20 11:22:27 2015
new/usr/src/cmd/make/include/vroot/report.h
make: fix GCC warnings
**********************************************************
```

```
 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 1994 Sun Microsystems, Inc. All rights reserved.
23  * Use is subject to license terms.
24  */

26 #ifndef _REPORT_H_
27 #define _REPORT_H_

29 #include <stdio.h>

31 extern FILE     *get_report_file(void);
32 extern char     *get_target_being_reported_for(void);
33 extern void     report_dependency(const char *name);
33 extern void     report_dependency(register char *name);
34 extern int      file_lock(char *name, char *lockname, int *file_locked, int time
35 #ifdef NSE
36 extern char     *setenv(char *name, char *value);
37 #endif

39 #define SUNPRO_DEPENDENCIES "SUNPRO_DEPENDENCIES"
40 #define LD      "LD"
41 #define COMP    "COMP"

43 /* the following definitions define the interface between make and
44  * NSE - the two systems must track each other.
45  */
46 #define NSE_DEPINFO             ".nse_depinfo"
47 #define NSE_DEPINFO_LOCK        ".nse_depinfo.lock"
48 #define NSE_DEP_ENV             "NSE_DEP"
49 #define NSE_TFS_PUSH            "/usr/nse/bin/tfs_push"
50 #define NSE_TFS_PUSH_LEN        8
51 #define NSE_VARIANT_ENV         "NSE_VARIANT"
52 #define NSE_RT_SOURCE_NAME      "Shared_Source"

54 #endif
```

```
**********************************************************
    2332 Wed May 20 11:22:27 2015
new/usr/src/cmd/make/include/vroot/vroot.h
make: fix GCC warnings
**********************************************************
_____unchanged_portion_omitted_
  39 typedef patht           *pathpt;

  41 extern  void            add_dir_to_path(const char *path, register pathpt *point
  41 extern  void            add_dir_to_path(register char *path, register pathpt *po
  42 extern  void            flush_path_cache(void);
  43 extern  void            flush_vroot_cache(void);
  44 extern  const char      *get_path_name(void);
  44 extern  char            *get_path_name(void);
  45 extern  char            *get_vroot_path(register char **vroot, register char **p
  46 extern  const char      *get_vroot_name(void);
  46 extern  char            *get_vroot_name(void);
  47 extern  int             open_vroot(char *path, int flags, int mode, pathpt vroot
  48 extern  pathpt          parse_path_string(register char *string, register int re
  49 extern  void            scan_path_first(void);
  50 extern  void            scan_vroot_first(void);
  51 extern  void            set_path_style(int style);

  53 extern  int             access_vroot(char *path, int mode, pathpt vroot_path, pa

  55 extern  int             execve_vroot(char *path, char **argv, char **environ, pa

  57 extern  int             lstat_vroot(char *path, struct stat *buffer, pathpt vroo
  58 extern  int             stat_vroot(char *path, struct stat *buffer, pathpt vroot
  59 extern  int             readlink_vroot(char *path, char *buffer, int buffer_size


  62 extern nl_catd  libvroot_catd;
  63 #endif
```

```
**********************************************************
   23005 Wed May 20 11:22:28 2015
new/usr/src/cmd/make/lib/mksh/dosys.cc
make: fix GCC warnings
**********************************************************
______unchanged_portion_omitted_

 297 /*
 298  *      doshell(command, ignore_error)
 299  *
 300  *      Used to run command lines that include shell meta-characters.
 301  *      The make macro SHELL is supposed to contain a path to the shell.
 302  *
 303  *      Return value:
 304  *                              The pid of the process we started
 305  *
 306  *      Parameters:
 307  *              command         The command to run
 308  *              ignore_error    Should we abort on error?
 309  *
 310  *      Global variables used:
 311  *              filter_stderr   If -X is on we redirect stderr
 312  *              shell_name      The Name "SHELL", used to get the path to shell
 313  */
 314 int
 315 doshell(wchar_t *command, register Boolean ignore_error, Boolean redirect_out_er
 316 {
 317         char                    *argv[6];
 318         int                     argv_index = 0;
 319         int                     cmd_argv_index;
 320         int                     length;
 321         char                    nice_prio_buf[MAXPATHLEN];
 322         register Name           shell = getvar(shell_name);
 323         register char           *shellname;
 324         char                    *tmp_mbs_buffer;


 327         if (IS_EQUAL(shell->string_mb, "")) {
 328                 shell = shell_name;
 329         }
 330         if ((shellname = strrchr(shell->string_mb, (int) slash_char)) == NULL) {
 331                 shellname = shell->string_mb;
 332         } else {
 333                 shellname++;
 334         }

 336         /*
 337          * Only prepend the /usr/bin/nice command to the original command
 338          * if the nice priority, nice_prio, is NOT zero (0).
 339          * Nice priorities can be a positive or a negative number.
 340          */
 341         if (nice_prio != 0) {
 342                 argv[argv_index++] = (char *)NOCATGETS("nice");
 342                 argv[argv_index++] = NOCATGETS("nice");
 343                 (void) sprintf(nice_prio_buf, NOCATGETS("-%d"), nice_prio);
 344                 argv[argv_index++] = strdup(nice_prio_buf);
 345         }
 346         argv[argv_index++] = shellname;
 347 #if defined(linux)
 348         if(0 == strcmp(shell->string_mb, (char*)NOCATGETS("/bin/sh"))) {
 349                 argv[argv_index++] = (char*)(ignore_error ? NOCATGETS("-c") : NO
 350         } else {
 351                 argv[argv_index++] = (char*)NOCATGETS("-c");
 352         }
 353 #else
 354         argv[argv_index++] = (char*)(ignore_error ? NOCATGETS("-c") : NOCATGETS(
```



```
 355 #endif
 356         if ((length = wslen(command)) >= MAXPATHLEN) {
 357                 tmp_mbs_buffer = getmem((length * MB_LEN_MAX) + 1);
 358                 (void) wcstombs(tmp_mbs_buffer, command, (length * MB_LEN_MAX) +
 359                 cmd_argv_index = argv_index;
 360                 argv[argv_index++] = strdup(tmp_mbs_buffer);
 361                 retmem_mb(tmp_mbs_buffer);
 362         } else {
 363                 WCSTOMBS(mbs_buffer, command);
 364                 cmd_argv_index = argv_index;
 365 #if defined(linux)
 366                 int mbl = strlen(mbs_buffer);
 367                 if(mbl > 2) {
 368                         if(mbs_buffer[mbl-1] == '\n' && mbs_buffer[mbl-2] == '\\
 369                                 mbs_buffer[mbl] = '\n';
 370                                 mbs_buffer[mbl+1] = 0;
 371                         }
 372                 }
 373 #endif
 374                 argv[argv_index++] = strdup(mbs_buffer);
 375         }
 376         argv[argv_index] = NULL;
 377         (void) fflush(stdout);
 378         if ((childPid = fork()) == 0) {
 379                 enable_interrupt((void (*) (int)) SIG_DFL);
 380                 if (redirect_out_err) {
 381                         redirect_io(stdout_file, stderr_file);
 382                 }
 383 #if 0
 384                 if (filter_stderr) {
 385                         redirect_stderr();
 386                 }
 387 #endif
 388                 if (nice_prio != 0) {
 389                         (void) execve(NOCATGETS("/usr/bin/nice"), argv, environ)
 390                         fatal_mksh(catgets(libmksdmsi18n_catd, 1, 92, "Could not
 391                               errmsg(errno));
 392                 } else {
 393                         (void) execve(shell->string_mb, argv, environ);
 394                         fatal_mksh(catgets(libmksdmsi18n_catd, 1, 93, "Could not
 395                               shell->string_mb,
 396                               errmsg(errno));
 397                 }
 398         }
 399         if (childPid  == -1) {
 400                 fatal_mksh(catgets(libmksdmsi18n_catd, 1, 94, "fork failed: %s")
 401                       errmsg(errno));
 402         }
 403         retmem_mb(argv[cmd_argv_index]);
 404         return childPid;
 405 }
______unchanged_portion_omitted_
```

```
**********************************************************
   39291 Wed May 20 11:22:29 2015
new/usr/src/cmd/make/lib/mksh/macro.cc
make: fix GCC warnings
**********************************************************
```

```
/*
 * 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.
 */


/*
 *      macro.cc
 *
 *      Handle expansion of make macros
 */

/*
 * Included files
 */
#include <mksh/dosys.h>         /* sh_command2string() */
#include <mksh/i18n.h>          /* get_char_semantics_value() */
#include <mksh/macro.h>
#include <mksh/misc.h>          /* retmem() */
#include <mksh/read.h>          /* get_next_block_fn() */
#include <mksdmsi18n/mksdmsi18n.h>      /* libmksdmsi18n_init() */

#include <widec.h>

#endif /* ! codereview */
/*
 * File table of contents
 */
static void     add_macro_to_global_list(Name macro_to_add);
#ifdef NSE
static void     expand_value_with_daemon(Name name, register Property macro, reg
#else
static void     expand_value_with_daemon(Name, register Property macro, register
#endif

static void     init_arch_macros(void);
static void     init_mach_macros(void);
static Boolean  init_arch_done = false;
static Boolean  init_mach_done = false;
```



```
long env_alloc_num = 0;
long env_alloc_bytes = 0;

/*
 *      getvar(name)
 *
 *      Return expanded value of macro.
 *
 *      Return value:
 *                              The expanded value of the macro
 *
 *      Parameters:
 *              name            The name of the macro we want the value for
 *
 *      Global variables used:
 */
Name
getvar(register Name name)
{
        String_rec              destination;
        wchar_t                 buffer[STRING_BUFFER_LENGTH];
        register Name           result;

        if ((name == host_arch) || (name == target_arch)) {
                if (!init_arch_done) {
                        init_arch_done = true;
                        init_arch_macros();
                }
        }
        if ((name == host_mach) || (name == target_mach)) {
                if (!init_mach_done) {
                        init_mach_done = true;
                        init_mach_macros();
                }
        }

        INIT_STRING_FROM_STACK(destination, buffer);
        expand_value(maybe_append_prop(name, macro_prop)->body.macro.value,
                     &destination,
                     false);
        result = GETNAME(destination.buffer.start, FIND_LENGTH);
        if (destination.free_after_use) {
                retmem(destination.buffer.start);
        }
        return result;
}

/*
 *      expand_value(value, destination, cmd)
 *
 *      Recursively expands all macros in the string value.
 *      destination is where the expanded value should be appended.
 *
 *      Parameters:
 *              value           The value we are expanding
 *              destination     Where to deposit the expansion
 *              cmd             If we are evaluating a command line we
 *                              turn \ quoting off
 *
 *      Global variables used:
 */
void
expand_value(Name value, register String destination, Boolean cmd)
{
        Source_rec              sourceb;
        register Source         source = &sourceb;
```

```
128         register wchar_t        *source_p = NULL;
129         register wchar_t        *source_end = NULL;
130         wchar_t                 *block_start = NULL;
131         int                     quote_seen = 0;

133         if (value == NULL) {
134                 /*
135                  * Make sure to get a string allocated even if it
136                  * will be empty.
137                  */
138                 MBSTOWCS(wcs_buffer, "");
139                 append_string(wcs_buffer, destination, FIND_LENGTH);
140                 destination->text.end = destination->text.p;
141                 return;
142         }
143         if (!value->dollar) {
144                 /*
145                  * If the value we are expanding does not contain
146                  * any $, we don't have to parse it.
147                  */
148                 APPEND_NAME(value,
149                         destination,
150                         (int) value->hash.length
151                 );
152                 destination->text.end = destination->text.p;
153                 return;
154         }

156         if (value->being_expanded) {
157                 fatal_reader_mksh(catgets(libmksdmsi18n_catd, 1, 113, "Loop dete
158                         value->string_mb);
159         }
160         value->being_expanded = true;
161         /* Setup the structure we read from */
162         Wstring vals(value);
163         sourceb.string.text.p = sourceb.string.buffer.start = wsdup(vals.get_str
164         sourceb.string.free_after_use = true;
165         sourceb.string.text.end =
166           sourceb.string.buffer.end =
167             sourceb.string.text.p + value->hash.length;
168         sourceb.previous = NULL;
169         sourceb.fd = -1;
170         sourceb.inp_buf =
171           sourceb.inp_buf_ptr =
172             sourceb.inp_buf_end = NULL;
173         sourceb.error_converting = false;
174         /* Lift some pointers from the struct to local register variables */
175         CACHE_SOURCE(0);
176 /* We parse the string in segments */
177 /* We read chars until we find a $, then we append what we have read so far */
178 /* (since last $ processing) to the destination. When we find a $ we call */
179 /* expand_macro() and let it expand that particular $ reference into dest */
180         block_start = source_p;
181         quote_seen = 0;
182         for (; 1; source_p++) {
183                 switch (GET_CHAR()) {
184                 case backslash_char:
185                         /* Quote $ in macro value */
186                         if (!cmd) {
187                                 quote_seen = ~quote_seen;
188                         }
189                         continue;
190                 case dollar_char:
191                         /* Save the plain string we found since */
192                         /* start of string or previous $ */
193                         if (quote_seen) {
```



```
194                                 append_string(block_start,
195                                         destination,
196                                         source_p - block_start - 1);
197                                 block_start = source_p;
198                                 break;
199                         }
200                         append_string(block_start,
201                                 destination,
202                                 source_p - block_start);
203                         source->string.text.p = ++source_p;
204                         UNCACHE_SOURCE();
205                         /* Go expand the macro reference */
206                         expand_macro(source, destination, sourceb.string.buffer.
207                         CACHE_SOURCE(1);
208                         block_start = source_p + 1;
209                         break;
210                 case nul_char:
211                         /* The string ran out. Get some more */
212                         append_string(block_start,
213                                 destination,
214                                 source_p - block_start);
215                         GET_NEXT_BLOCK_NOCHK(source);
216                         if (source == NULL) {
217                                 destination->text.end = destination->text.p;
218                                 value->being_expanded = false;
219                                 return;
220                         }
221                         if (source->error_converting) {
222                                 fatal_reader_mksh(NOCATGETS("Internal error: Inv
223                         }
224                         block_start = source_p;
225                         source_p--;
226                         continue;
227                 }
228                 quote_seen = 0;
229         }
230         retmem(sourceb.string.buffer.start);
231 }

233 /*
234  *      expand_macro(source, destination, current_string, cmd)
235  *
236  *      Should be called with source->string.text.p pointing to
237  *      the first char after the $ that starts a macro reference.
238  *      source->string.text.p is returned pointing to the first char after
239  *      the macro name.
240  *      It will read the macro name, expanding any macros in it,
241  *      and get the value. The value is then expanded.
242  *      destination is a String that is filled in with the expanded macro.
243  *      It may be passed in referencing a buffer to expand the macro into.
244  *      Note that most expansions are done on demand, e.g. right
245  *      before the command is executed and not while the file is
246  *      being parsed.
247  *
248  *      Parameters:
249  *              source          The source block that references the string
250  *                              to expand
251  *              destination     Where to put the result
252  *              current_string  The string we are expanding, for error msg
253  *              cmd             If we are evaluating a command line we
254  *                              turn \ quoting off
255  *
256  *      Global variables used:
257  *              funny           Vector of semantic tags for characters
258  *              is_conditional  Set if a conditional macro is refd
259  *              make_word_mentioned Set if the word "MAKE" is mentioned
```

```
 *              makefile_type   We deliver extra msg when reading makefiles
 *              query           The Name "?", compared against
 *              query_mentioned Set if the word "?" is mentioned
 */
void
expand_macro(register Source source, register String destination, wchar_t *curre
{
        static Name             make = (Name)NULL;
        static wchar_t          colon_sh[4];
        static wchar_t          colon_shell[7];
        String_rec              string;
        wchar_t                 buffer[STRING_BUFFER_LENGTH];
        register wchar_t        *source_p = source->string.text.p;
        register wchar_t        *source_end = source->string.text.end;
        register int            closer = 0;
        wchar_t                 *block_start = (wchar_t *)NULL;
        int                     quote_seen = 0;
        register int            closer_level = 1;
        Name                    name = (Name)NULL;
        wchar_t                 *colon = (wchar_t *)NULL;
        wchar_t                 *percent = (wchar_t *)NULL;
        wchar_t                 *eq = (wchar_t *) NULL;
        Property                macro = NULL;
        wchar_t                 *p = (wchar_t*)NULL;
        String_rec              extracted;
        wchar_t                 extracted_string[MAXPATHLEN];
        wchar_t                 *left_head = NULL;
        wchar_t                 *left_tail = NULL;
        wchar_t                 *right_tail = NULL;
        int                     left_head_len = 0;
        int                     left_tail_len = 0;
        int                     tmp_len = 0;
        wchar_t                 *right_hand[128];
        int                     i = 0;
        enum {
                no_extract,
                dir_extract,
                file_extract
        }                       extraction = no_extract;
        enum {
                no_replace,
                suffix_replace,
                pattern_replace,
                sh_replace
        }                       replacement = no_replace;

        if (make == NULL) {
                MBSTOWCS(wcs_buffer, NOCATGETS("MAKE"));
                make = GETNAME(wcs_buffer, FIND_LENGTH);

                MBSTOWCS(colon_sh, NOCATGETS(":sh"));
                MBSTOWCS(colon_shell, NOCATGETS(":shell"));
        }

        right_hand[0] = NULL;

        /* First copy the (macro-expanded) macro name into string. */
        INIT_STRING_FROM_STACK(string, buffer);
recheck_first_char:
        /* Check the first char of the macro name to figure out what to do. */
        switch (GET_CHAR()) {
        case nul_char:
                GET_NEXT_BLOCK_NOCHK(source);
                if (source == NULL) {
                        WCSTOMBS(mbs_buffer, current_string);
                        fatal_reader_mksh(catgets(libmksdmsi18n_catd, 1, 114, "’
```



```
                                        mbs_buffer);
                }
                if (source->error_converting) {
                        fatal_reader_mksh(NOCATGETS("Internal error: Invalid byt
                }
                goto recheck_first_char;
        case parenleft_char:
                /* Multi char name. */
                closer = (int) parenright_char;
                break;
        case braceleft_char:
                /* Multi char name. */
                closer = (int) braceright_char;
                break;
        case newline_char:
                fatal_reader_mksh(catgets(libmksdmsi18n_catd, 1, 115, "’$’ at en
        default:
                /* Single char macro name. Just suck it up */
                append_char(*source_p, &string);
                source->string.text.p = source_p + 1;
                goto get_macro_value;
        }

        /* Handle multi-char macro names */
        block_start = ++source_p;
        quote_seen = 0;
        for (; 1; source_p++) {
                switch (GET_CHAR()) {
                case nul_char:
                        append_string(block_start,
                                      &string,
                                      source_p - block_start);
                        GET_NEXT_BLOCK_NOCHK(source);
                        if (source == NULL) {
                                if (current_string != NULL) {
                                        WCSTOMBS(mbs_buffer, current_string);
                                        fatal_reader_mksh(catgets(libmksdmsi18n_
                                                closer ==
                                                (int) braceright_char ?
                                                (int) braceleft_char :
                                                (int) parenleft_char,
                                                mbs_buffer);
                                } else {
                                        fatal_reader_mksh(catgets(libmksdmsi18n_
                                }
                        }
                        if (source->error_converting) {
                                fatal_reader_mksh(NOCATGETS("Internal error: Inv
                        }
                        block_start = source_p;
                        source_p--;
                        continue;
                case newline_char:
                        fatal_reader_mksh(catgets(libmksdmsi18n_catd, 1, 118, "U
                                closer == (int) braceright_char ?
                                (int) braceleft_char :
                                (int) parenleft_char);
                case backslash_char:
                        /* Quote dollar in macro value. */
                        if (!cmd) {
                                quote_seen = ~quote_seen;
                        }
                        continue;
                case dollar_char:
                        /*
                         * Macro names may reference macros.
```

```
392                          * This expands the value of such macros into the
393                          * macro name string.
394                          */
395                         if (quote_seen) {
396                                 append_string(block_start,
397                                               &string,
398                                               source_p - block_start - 1);
399                                 block_start = source_p;
400                                 break;
401                         }
402                         append_string(block_start,
403                                       &string,
404                                       source_p - block_start);
405                         source->string.text.p = ++source_p;
406                         UNCACHE_SOURCE();
407                         expand_macro(source, &string, current_string, cmd);
408                         CACHE_SOURCE(0);
409                         block_start = source_p;
410                         source_p--;
411                         break;
412                 case parenleft_char:
413                         /* Allow nested pairs of () in the macro name. */
414                         if (closer == (int) parenright_char) {
415                                 closer_level++;
416                         }
417                         break;
418                 case braceleft_char:
419                         /* Allow nested pairs of {} in the macro name. */
420                         if (closer == (int) braceright_char) {
421                                 closer_level++;
422                         }
423                         break;
424                 case parenright_char:
425                 case braceright_char:
426                         /*
427                          * End of the name. Save the string in the macro
428                          * name string.
429                          */
430                         if ((*source_p == closer) && (--closer_level <= 0)) {
431                                 source->string.text.p = source_p + 1;
432                                 append_string(block_start,
433                                               &string,
434                                               source_p - block_start);
435                                 goto get_macro_value;
436                         }
437                         break;
438                 }
439                 quote_seen = 0;
440         }
441         /*
442          * We got the macro name. We now inspect it to see if it
443          * specifies any translations of the value.
444          */
445 get_macro_value:
446         name = NULL;
447         /* First check if we have a $(@D) type translation. */
448         if ((get_char_semantics_value(string.buffer.start[0]) &
449              (int) special_macro_sem) &&
450             (string.text.p - string.buffer.start >= 2) &&
451             ((string.buffer.start[1] == 'D') ||
452              (string.buffer.start[1] == 'F'))) {
453                 switch (string.buffer.start[1]) {
454                 case 'D':
455                         extraction = dir_extract;
456                         break;
457                 case 'F':
```



```
458                         extraction = file_extract;
459                         break;
460                 default:
461                         WCSTOMBS(mbs_buffer, string.buffer.start);
462                         fatal_reader_mksh(catgets(libmksdmsi18n_catd, 1, 119, "I
463                                           mbs_buffer);
464                 }
465                 /* Internalize the macro name using the first char only. */
466                 name = GETNAME(string.buffer.start, 1);
467                 (void) wscpy(string.buffer.start, string.buffer.start + 2);
468         }
469         /* Check for other kinds of translations. */
470         if ((colon = (wchar_t *) wschr(string.buffer.start,
471                                        (int) colon_char)) != NULL) {
472                 /*
473                  * We have a $(FOO:.c=.o) type translation.
474                  * Get the name of the macro proper.
475                  */
476                 if (name == NULL) {
477                         name = GETNAME(string.buffer.start,
478                                        colon - string.buffer.start);
479                 }
480                 /* Pickup all the translations. */
481                 if (IS_WEQUAL(colon, colon_sh) || IS_WEQUAL(colon, colon_shell))
482                         replacement = sh_replace;
483                 } else if ((svr4) ||
484                            ((percent = (wchar_t *) wschr(colon + 1,
485                                                          (int) percent_char)) ==
486                         while (colon != NULL) {
487                                 if ((eq = (wchar_t *) wschr(colon + 1,
488                                                             (int) equal_char)) =
489                                         fatal_reader_mksh(catgets(libmksdmsi18n_
490                                 }
491                                 left_tail_len = eq - colon - 1;
492                                 if(left_tail) {
493                                         retmem(left_tail);
494                                 }
495                                 left_tail = ALLOC_WC(left_tail_len + 1);
496                                 (void) wsncpy(left_tail,
497                                               colon + 1,
498                                               eq - colon - 1);
499                                 left_tail[eq - colon - 1] = (int) nul_char;
500                                 replacement = suffix_replace;
501                                 if ((colon = (wchar_t *) wschr(eq + 1,
502                                                                (int) colon_char)
503                                         tmp_len = colon - eq;
504                                         if(right_tail) {
505                                                 retmem(right_tail);
506                                         }
507                                         right_tail = ALLOC_WC(tmp_len);
508                                         (void) wsncpy(right_tail,
509                                                       eq + 1,
510                                                       colon - eq - 1);
511                                         right_tail[colon - eq - 1] =
512                                           (int) nul_char;
513                                 } else {
514                                         if(right_tail) {
515                                                 retmem(right_tail);
516                                         }
517                                         right_tail = ALLOC_WC(wslen(eq) + 1);
518                                         (void) wscpy(right_tail, eq + 1);
519                                 }
520                         }
521                 } else {
522                         if ((eq = (wchar_t *) wschr(colon + 1,
523                                                     (int) equal_char)) == NULL)
```

```
524                             fatal_reader_mksh(catgets(libmksdmsi18n_catd, 1,
525                     }
526                     if ((percent = (wchar_t *) wschr(colon + 1,
527                                                      (int) percent_char)) ==
528                             fatal_reader_mksh(catgets(libmksdmsi18n_catd, 1,
529                     }
530                     if (eq < percent) {
531                             fatal_reader_mksh(catgets(libmksdmsi18n_catd, 1,
532                     }

534                     if (percent > (colon + 1)) {
535                             tmp_len = percent - colon;
536                             if(left_head) {
537                                     retmem(left_head);
538                             }
539                             left_head = ALLOC_WC(tmp_len);
540                             (void) wsncpy(left_head,
541                                           colon + 1,
542                                           percent - colon - 1);
543                             left_head[percent-colon-1] = (int) nul_char;
544                             left_head_len = percent-colon-1;
545                     } else {
546                             left_head = NULL;
547                             left_head_len = 0;
548                     }

550                     if (eq > percent+1) {
551                             tmp_len = eq - percent;
552                             if(left_tail) {
553                                     retmem(left_tail);
554                             }
555                             left_tail = ALLOC_WC(tmp_len);
556                             (void) wsncpy(left_tail,
557                                           percent + 1,
558                                           eq - percent - 1);
559                             left_tail[eq-percent-1] = (int) nul_char;
560                             left_tail_len = eq-percent-1;
561                     } else {
562                             left_tail = NULL;
563                             left_tail_len = 0;
564                     }

566                     if ((percent = (wchar_t *) wschr(++eq,
567                                                      (int) percent_char)) ==

569                             right_hand[0] = ALLOC_WC(wslen(eq) + 1);
570                             right_hand[1] = NULL;
571                             (void) wscpy(right_hand[0], eq);
572                     } else {
573                             i = 0;
574                             do {
575                                     right_hand[i] = ALLOC_WC(percent-eq+1);
576                                     (void) wsncpy(right_hand[i],
577                                                   eq,
578                                                   percent - eq);
579                                     right_hand[i][percent-eq] =
580                                       (int) nul_char;
581                                     if (i++ >= VSIZEOF(right_hand)) {
582                                             fatal_mksh(catgets(libmksdmsi18n
583                                     }
584                                     eq = percent + 1;
585                                     if (eq[0] == (int) nul_char) {
586                                             MBSTOWCS(wcs_buffer, "");
587                                             right_hand[i] = (wchar_t *) wsdu
588                                             i++;
589                                             break;
```



```
590                                     }
591                                     } while ((percent = (wchar_t *) wschr(eq, (int)
592                                     if (eq[0] != (int) nul_char) {
593                                             right_hand[i] = ALLOC_WC(wslen(eq) + 1);
594                                             (void) wscpy(right_hand[i], eq);
595                                             i++;
596                                     }
597                                     right_hand[i] = NULL;
598                             }
599                             replacement = pattern_replace;
600                     }
601             }
602             if (name == NULL) {
603                     /*
604                      * No translations found.
605                      * Use the whole string as the macro name.
606                      */
607                     name = GETNAME(string.buffer.start,
608                                    string.text.p - string.buffer.start);
609             }
610             if (string.free_after_use) {
611                     retmem(string.buffer.start);
612             }
613             if (name == make) {
614                     make_word_mentioned = true;
615             }
616             if (name == query) {
617                     query_mentioned = true;
618             }
619             if ((name == host_arch) || (name == target_arch)) {
620                     if (!init_arch_done) {
621                             init_arch_done = true;
622                             init_arch_macros();
623                     }
624             }
625             if ((name == host_mach) || (name == target_mach)) {
626                     if (!init_mach_done) {
627                             init_mach_done = true;
628                             init_mach_macros();
629                     }
630             }
631             /* Get the macro value. */
632             macro = get_prop(name->prop, macro_prop);
633 #ifdef NSE
634             if (nse_watch_vars && nse && macro != NULL) {
635                     if (macro->body.macro.imported) {
636                             nse_shell_var_used= name;
637                     }
638                     if (macro->body.macro.value != NULL){
639                           if (nse_backquotes(macro->body.macro.value->string)) {
640                                   nse_backquote_seen= name;
641                           }
642                        }
643             }
644 #endif
645             if ((macro != NULL) && macro->body.macro.is_conditional) {
646                     conditional_macro_used = true;
647                     /*
648                      * Add this conditional macro to the beginning of the
649                      * global list.
650                      */
651                     add_macro_to_global_list(name);
652                     if (makefile_type == reading_makefile) {
653                             warning_mksh(catgets(libmksdmsi18n_catd, 1, 164, "Condit
654                                          name->string_mb, file_being_read, line_n
655                     }
```

```
656         }
657         /* Macro name read and parsed. Expand the value. */
658         if ((macro == NULL) || (macro->body.macro.value == NULL)) {
659                 /* If the value is empty, we just get out of here. */
660                 goto exit;
661         }
662         if (replacement == sh_replace) {
663                 /* If we should do a :sh transform, we expand the command
664                  * and process it.
665                  */
666                 INIT_STRING_FROM_STACK(string, buffer);
667                 /* Expand the value into a local string buffer and run cmd. */
668                 expand_value_with_daemon(name, macro, &string, cmd);
669                 sh_command2string(&string, destination);
670         } else if ((replacement != no_replace) || (extraction != no_extract)) {
671                 /*
672                  * If there were any transforms specified in the macro
673                  * name, we deal with them here.
674                  */
675                 INIT_STRING_FROM_STACK(string, buffer);
676                 /* Expand the value into a local string buffer. */
677                 expand_value_with_daemon(name, macro, &string, cmd);
678                 /* Scan the expanded string. */
679                 p = string.buffer.start;
680                 while (*p != (int) nul_char) {
681                         wchar_t         chr;
682
683                         /*
684                          * First skip over any white space and append
685                          * that to the destination string.
686                          */
687                         block_start = p;
688                         while ((*p != (int) nul_char) && iswspace(*p)) {
689                                 p++;
690                         }
691                         append_string(block_start,
692                                       destination,
693                                       p - block_start);
694                         /* Then find the end of the next word. */
695                         block_start = p;
696                         while ((*p != (int) nul_char) && !iswspace(*p)) {
697                                 p++;
698                         }
699                         /* If we cant find another word we are done */
700                         if (block_start == p) {
701                                 break;
702                         }
703                         /* Then apply the transforms to the word */
704                         INIT_STRING_FROM_STACK(extracted, extracted_string);
705                         switch (extraction) {
706                         case dir_extract:
707                                 /*
708                                  * $(@D) type transform. Extract the
709                                  * path from the word. Deliver "." if
710                                  * none is found.
711                                  */
712                                 if (p != NULL) {
713                                         chr = *p;
714                                         *p = (int) nul_char;
715                                 }
716                                 eq = (wchar_t *) wsrchr(block_start, (int) slash
717                                 if (p != NULL) {
718                                         *p = chr;
719                                 }
720                                 if ((eq == NULL) || (eq > p)) {
721                                         MBSTOWCS(wcs_buffer, ".");
```



```
722                                         append_string(wcs_buffer, &extracted, 1)
723                                 } else {
724                                         append_string(block_start,
725                                                       &extracted,
726                                                       eq - block_start);
727                                 }
728                                 break;
729                         case file_extract:
730                                 /*
731                                  * $(@F) type transform. Remove the path
732                                  * from the word if any.
733                                  */
734                                 if (p != NULL) {
735                                         chr = *p;
736                                         *p = (int) nul_char;
737                                 }
738                                 eq = (wchar_t *) wsrchr(block_start, (int) slash
739                                 if (p != NULL) {
740                                         *p = chr;
741                                 }
742                                 if ((eq == NULL) || (eq > p)) {
743                                         append_string(block_start,
744                                                       &extracted,
745                                                       p - block_start);
746                                 } else {
747                                         append_string(eq + 1,
748                                                       &extracted,
749                                                       p - eq - 1);
750                                 }
751                                 break;
752                         case no_extract:
753                                 append_string(block_start,
754                                               &extracted,
755                                               p - block_start);
756                                 break;
757                         }
758                         switch (replacement) {
759                         case suffix_replace:
760                                 /*
761                                  * $(FOO:.o=.c) type transform.
762                                  * Maybe replace the tail of the word.
763                                  */
764                                 if (((extracted.text.p -
765                                       extracted.buffer.start) >=
766                                      left_tail_len) &&
767                                     IS_WEQUALN(extracted.text.p - left_tail_len,
768                                                left_tail,
769                                                left_tail_len)) {
770                                         append_string(extracted.buffer.start,
771                                                       destination,
772                                                       (extracted.text.p -
773                                                        extracted.buffer.start)
774                                                       - left_tail_len);
775                                         append_string(right_tail,
776                                                       destination,
777                                                       FIND_LENGTH);
778                                 } else {
779                                         append_string(extracted.buffer.start,
780                                                       destination,
781                                                       FIND_LENGTH);
782                                 }
783                                 break;
784                         case pattern_replace:
785                                 /* $(X:a%b=c%d) type transform. */
786                                 if (((extracted.text.p -
787                                       extracted.buffer.start) >=
```

```
788                                      left_head_len+left_tail_len) &&
789                                     IS_WEQUALN(left_head,
790                                              extracted.buffer.start,
791                                              left_head_len) &&
792                                     IS_WEQUALN(left_tail,
793                                              extracted.text.p - left_tail_len,
794                                              left_tail_len)) {
795                                         i = 0;
796                                         while (right_hand[i] != NULL) {
797                                                 append_string(right_hand[i],
798                                                               destination,
799                                                               FIND_LENGTH);
800                                                 i++;
801                                                 if (right_hand[i] != NULL) {
802                                                         append_string(extracted.
803                                                                       start +
804                                                                       left_head_
805                                                                       destinatio
806                                                                       (extracted
807                                                 }
808                                         }
809                                 } else {
810                                         append_string(extracted.buffer.start,
811                                                       destination,
812                                                       FIND_LENGTH);
813                                 }
814                                 break;
815                         case no_replace:
816                                 append_string(extracted.buffer.start,
817                                               destination,
818                                               FIND_LENGTH);
819                                 break;
820                         case sh_replace:
821                                 break;
822                             }
823                 }
824                 if (string.free_after_use) {
825                         retmem(string.buffer.start);
826                 }
827         } else {
828                 /*
829                  * This is for the case when the macro name did not
830                  * specify transforms.
831                  */
832                 if (!strncmp(name->string_mb, NOCATGETS("GET"), 3)) {
833                         dollarget_seen = true;
834                 }
835                 dollarless_flag = false;
836                 if (!strncmp(name->string_mb, "<", 1) &&
837                     dollarget_seen) {
838                         dollarless_flag = true;
839                         dollarget_seen = false;
840                 }
841                 expand_value_with_daemon(name, macro, destination, cmd);
842         }
843 exit:
844         if(left_tail) {
845                 retmem(left_tail);
846         }
847         if(right_tail) {
848                 retmem(right_tail);
849         }
850         if(left_head) {
851                 retmem(left_head);
852         }
853         i = 0;
```



```
854         while (right_hand[i] != NULL) {
855                 retmem(right_hand[i]);
856                 i++;
857         }
858         *destination->text.p = (int) nul_char;
859         destination->text.end = destination->text.p;
860 }

862 static void
863 add_macro_to_global_list(Name macro_to_add)
864 {
865         Macro_list      new_macro;
866         Macro_list      macro_on_list;
867         char            *name_on_list = (char*)NULL;
868         char            *name_to_add = macro_to_add->string_mb;
869         char            *value_on_list = (char*)NULL;
870         const char      *value_to_add = (char*)NULL;
 43         char            *value_to_add = (char*)NULL;

872         if (macro_to_add->prop->body.macro.value != NULL) {
873                 value_to_add = macro_to_add->prop->body.macro.value->string_mb;
874         } else {
875                 value_to_add = "";
876         }

878         /*
879          * Check if this macro is already on list, if so, do nothing
880          */
881         for (macro_on_list = cond_macro_list;
882              macro_on_list != NULL;
883              macro_on_list = macro_on_list->next) {

885                 name_on_list = macro_on_list->macro_name;
886                 value_on_list = macro_on_list->value;

888                 if (IS_EQUAL(name_on_list, name_to_add)) {
889                         if (IS_EQUAL(value_on_list, value_to_add)) {
890                                 return;
891                         }
892                 }
893         }
894         new_macro = (Macro_list) malloc(sizeof(Macro_list_rec));
895         new_macro->macro_name = strdup(name_to_add);
896         new_macro->value = strdup(value_to_add);
897         new_macro->next = cond_macro_list;
898         cond_macro_list = new_macro;
899 }
```

_____*unchanged_portion_omitted_*

```
**********************************************************
   25884 Wed May 20 11:22:29 2015
new/usr/src/cmd/make/lib/mksh/misc.cc
make: fix GCC warnings
**********************************************************
```

```
/*
 * 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 2004 Sun Microsystems, Inc. All rights reserved.
 * Use is subject to license terms.
 */


/*
 *      misc.cc
 *
 *      This file contains various unclassified routines. Some main groups:
 *              getname
 *              Memory allocation
 *              String handling
 *              Property handling
 *              Error message handling
 *              Make internal state dumping
 *              main routine support
 */

/*
 * Included files
 */
#include <bsd/bsd.h>            /* bsd_signal() */
#include <mksh/i18n.h>          /* get_char_semantics_value() */
#include <mksh/misc.h>
#include <mksdmsi18n/mksdmsi18n.h>
#include <stdarg.h>             /* va_list, va_start(), va_end() */
#include <stdlib.h>             /* mbstowcs() */
#include <sys/signal.h>         /* SIG_DFL */
#include <sys/wait.h>           /* wait() */

#ifdef SUN5_0
#include <string.h>             /* strerror() */
#endif

#if defined (HP_UX) || defined (linux)
#include <unistd.h>
#endif

/*
 * Defined macros
```



```
 */

/*
 * typedefs & structs
 */

/*
 * Static variables
 */
#ifdef SUN5_0
extern "C" {
        void            (*sigivalue)(int) = SIG_DFL;
        void            (*sigqvalue)(int) = SIG_DFL;
        void            (*sigtvalue)(int) = SIG_DFL;
        void            (*sighvalue)(int) = SIG_DFL;
}
extern "C" void         (*sigivalue)(int) = SIG_DFL;
extern "C" void         (*sigqvalue)(int) = SIG_DFL;
extern "C" void         (*sigtvalue)(int) = SIG_DFL;
extern "C" void         (*sighvalue)(int) = SIG_DFL;
#else
static  void            (*sigivalue)(int) = (void (*) (int)) SIG_DFL;
static  void            (*sigqvalue)(int) = (void (*) (int)) SIG_DFL;
static  void            (*sigtvalue)(int) = (void (*) (int)) SIG_DFL;
static  void            (*sighvalue)(int) = (void (*) (int)) SIG_DFL;
#endif

long    getname_bytes_count = 0;
long    getname_names_count = 0;
long    getname_struct_count = 0;

long    freename_bytes_count = 0;
long    freename_names_count = 0;
long    freename_struct_count = 0;

long    expandstring_count = 0;
long    getwstring_count = 0;

/*
 * File table of contents
 */
static void     expand_string(register String string, register int length);

#define FATAL_ERROR_MSG_SIZE 200

/*
 *      getmem(size)
 *
 *      malloc() version that checks the returned value.
 *
 *      Return value:
 *                              The memory chunk we allocated
 *
 *      Parameters:
 *              size            The size of the chunk we need
 *
 *      Global variables used:
 */
char *
getmem(register int size)
{
        register char           *result = (char *) malloc((unsigned) size);
        if (result == NULL) {
                char buf[FATAL_ERROR_MSG_SIZE];
                sprintf(buf, NOCATGETS("*** Error: malloc(%d) failed: %s\n"), si
                strcat(buf, catgets(libmksdmsi18n_catd, 1, 126, "mksh: Fatal err
```

```
124                 fputs(buf, stderr);
125 #ifdef SUN5_0
126                 exit_status = 1;
127 #endif
128                 exit(1);
129         }
130         return result;
131 }
```

*unchanged_portion_omitted_*

```
317 /*
318  *      setup_char_semantics()
319  *
320  *      Load the vector char_semantics[] with lexical markers
321  *
322  *      Parameters:
323  *
324  *      Global variables used:
325  *              char_semantics  The vector of character semantics that we set
326  */
327 void
328 setup_char_semantics(void)
329 {
330         const char      *s;
328         char            *s;
331         wchar_t         wc_buffer[1];
332         int             entry;

334         if (svr4) {
335                 s = "@-";
336         } else {
337                 s = "=@-?!+";
338         }
339         for (s; MBTOWC(wc_buffer, s); s++) {
340                 entry = get_char_semantics_entry(*wc_buffer);
341                 char_semantics[entry] |= (int) command_prefix_sem;
342         }
343         char_semantics[dollar_char_entry] |= (int) dollar_sem;
344         for (s = "#|=^();&<>*?[]:$‘’\"\\\n"; MBTOWC(wc_buffer, s); s++) {
345                 entry = get_char_semantics_entry(*wc_buffer);
346                 char_semantics[entry] |= (int) meta_sem;
347         }
348         char_semantics[percent_char_entry] |= (int) percent_sem;
349         for (s = "@*<%?^"; MBTOWC(wc_buffer, s); s++) {
350                 entry = get_char_semantics_entry(*wc_buffer);
351                 char_semantics[entry] |= (int) special_macro_sem;
352         }
353         for (s = "?[*"; MBTOWC(wc_buffer, s); s++) {
354                 entry = get_char_semantics_entry(*wc_buffer);
355                 char_semantics[entry] |= (int) wildcard_sem;
356         }
357         char_semantics[colon_char_entry] |= (int) colon_sem;
358         char_semantics[parenleft_char_entry] |= (int) parenleft_sem;
359 }
```

*unchanged_portion_omitted_*

```
405 static char static_buf[MAXPATHLEN*3];

407 /*
408  *      fatal_mksh(format, args...)
409  *
410  *      Print a message and die
411  *
412  *      Parameters:
413  *              format          printf type format string
414  *              args            Arguments to match the format
```



```
415  */
416 /*VARARGS*/
417 void
418 fatal_mksh(const char *message, ...)
416 fatal_mksh(char * message, ...)
419 {
420         va_list args;
421         char    *buf = static_buf;
422         char    *mksh_fat_err = catgets(libmksdmsi18n_catd, 1, 128, "mksh: Fatal
423         char    *cur_wrk_dir = catgets(libmksdmsi18n_catd, 1, 129, "Current work
424         int     mksh_fat_err_len = strlen(mksh_fat_err);

426         va_start(args, message);
427         (void) fflush(stdout);
428         (void) strcpy(buf, mksh_fat_err);
429         size_t buf_len = vsnprintf(static_buf + mksh_fat_err_len,
430                                    sizeof(static_buf) - mksh_fat_err_len,
431                                    message, args)
432                         + mksh_fat_err_len
433                         + strlen(cur_wrk_dir)
434                         + strlen(get_current_path_mksh())
435                         + 3; // "\n\n"
436         va_end(args);
437         if (buf_len >= sizeof(static_buf)) {
438                 buf = getmem(buf_len);
439                 (void) strcpy(buf, mksh_fat_err);
440                 va_start(args, message);
441                 (void) vsprintf(buf + mksh_fat_err_len, message, args);
442                 va_end(args);
443         }
444         (void) strcat(buf, "\n");
445 /*
446         if (report_pwd) {
447  */
448         if (1) {
449                 (void) strcat(buf, cur_wrk_dir);
450                 (void) strcat(buf, get_current_path_mksh());
451                 (void) strcat(buf, "\n");
452         }
453         (void) fputs(buf, stderr);
454         (void) fflush(stderr);
455         if (buf != static_buf) {
456                 retmem_mb(buf);
457         }
458 #ifdef SUN5_0
459         exit_status = 1;
460 #endif
461         exit(1);
462 }

464 /*
465  *      fatal_reader_mksh(format, args...)
466  *
467  *      Parameters:
468  *              format          printf style format string
469  *              args            arguments to match the format
470  */
471 /*VARARGS*/
472 void
473 fatal_reader_mksh(const char * pattern, ...)
471 fatal_reader_mksh(char * pattern, ...)
474 {
475         va_list args;
476         char    message[1000];

478         va_start(args, pattern);
```

```
479 /*
480         if (file_being_read != NULL) {
481                 WCSTOMBS(mbs_buffer, file_being_read);
482                 if (line_number != 0) {
483                         (void) sprintf(message,
484                                         catgets(libmksdmsi18n_catd, 1, 130, "%s,
485                                         mbs_buffer,
486                                         line_number,
487                                         pattern);
488                 } else {
489                         (void) sprintf(message,
490                                         "%s: %s",
491                                         mbs_buffer,
492                                         pattern);
493                 }
494                 pattern = message;
495         }
496  */

498         (void) fflush(stdout);
499         (void) fprintf(stderr, catgets(libmksdmsi18n_catd, 1, 131, "mksh: Fatal
500         (void) vfprintf(stderr, pattern, args);
501         (void) fprintf(stderr, "\n");
502         va_end(args);

504 /*
505         if (temp_file_name != NULL) {
506                 (void) fprintf(stderr,
507                                 catgets(libmksdmsi18n_catd, 1, 132, "mksh: Temp-f
508                                 temp_file_name->string_mb);
509                 temp_file_name = NULL;
510         }
511  */

513 /*
514         if (report_pwd) {
515  */
516         if (1) {
517                 (void) fprintf(stderr,
518                                 catgets(libmksdmsi18n_catd, 1, 133, "Current work
519                                 get_current_path_mksh());
520         }
521         (void) fflush(stderr);
522 #ifdef SUN5_0
523         exit_status = 1;
524 #endif
525         exit(1);
526 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
    5519 Wed May 20 11:22:30 2015
new/usr/src/cmd/make/lib/vroot/lock.cc
make: fix GCC warnings
**********************************************************
```

```
   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 2004 Sun Microsystems, Inc. All rights reserved.
  23  * Use is subject to license terms.
  24  */

  26 #include <avo/intl.h>   /* for NOCATGETS */
  27 #include <stdio.h>
  28 #include <stdlib.h>
  29 #include <string.h>
  30 #include <sys/errno.h>
  31 #include <sys/param.h>
  32 #include <sys/stat.h>
  33 #include <sys/types.h>
  34 #include <unistd.h>
  35 #include <vroot/vroot.h>
  36 #include <mksdmsi18n/mksdmsi18n.h>
  37 #include <signal.h>
  38 #include <errno.h>                      /* errno */

  40 #if !defined(linux)
  41 extern  char            *sys_errlist[];
  42 extern  int             sys_nerr;
  43 #endif

  45 static  void            file_lock_error(char *msg, char *file, char *str, int ar

  47 #define BLOCK_INTERUPTS sigfillset(&newset) ; \
  48         sigprocmask(SIG_SETMASK, &newset, &oldset)

  50 #define UNBLOCK_INTERUPTS \
  51         sigprocmask(SIG_SETMASK, &oldset, &newset)

  53 /*
  54  * This code stolen from the NSE library and changed to not depend
  55  * upon any NSE routines or header files.
  56  *
  57  * Simple file locking.
  58  * Create a symlink to a file.  The "test and set" will be
  59  * atomic as creating the symlink provides both functions.
  60  *
  61  * The timeout value specifies how long to wait for stale locks
```



```
  62  * to disappear.  If the lock is more than 'timeout' seconds old
  63  * then it is ok to blow it away.  This part has a small window
  64  * of vunerability as the operations of testing the time,
  65  * removing the lock and creating a new one are not atomic.
  66  * It would be possible for two processes to both decide to blow
  67  * away the lock and then have process A remove the lock and establish
  68  * its own, and then then have process B remove the lock which accidentily
  69  * removes A's lock rather than the stale one.
  70  *
  71  * A further complication is with the NFS.  If the file in question is
  72  * being served by an NFS server, then its time is set by that server.
  73  * We can not use the time on the client machine to check for a stale
  74  * lock.  Therefore, a temp file on the server is created to get
  75  * the servers current time.
  76  *
  77  * Returns an error message.  NULL return means the lock was obtained.
  78  *
  79  * 12/6/91 Added the parameter "file_locked".  Before this parameter
  80  * was added, the calling procedure would have to wait for file_lock()
  81  * to return before it sets the flag. If the user interrupted "make"
  82  * between the time the lock was acquired and the time file_lock()
  83  * returns, make wouldn't know that the file has been locked, and therefore
  84  * it wouldn' remove the lock. Setting the flag right after locking the file
  85  * makes this window much smaller.
  86  */

  88 int
  89 file_lock(char *name, char *lockname, int *file_locked, int timeout)
  90 {
  91         int             counter = 0;
  92         static char     msg[MAXPATHLEN+1];
  93         int             printed_warning = 0;
  94         int             r;
  95         struct stat     statb;
  96         sigset_t newset;
  97         sigset_t oldset;

  99         *file_locked = 0;
 100         if (timeout <= 0) {
 101                 timeout = 120;
 102         }
 103         for (;;) {
 104                 BLOCK_INTERUPTS;
 105                 r = symlink(name, lockname);
 106                 if (r == 0) {
 107                         *file_locked = 1;
 108                         UNBLOCK_INTERUPTS;
 109                         return 0; /* success */
 110                 }
 111                 UNBLOCK_INTERUPTS;

 113                 if (errno != EEXIST) {
 114                         file_lock_error(msg, name, (char *)NOCATGETS("symlink(%s
 114                         file_lock_error(msg, name, NOCATGETS("symlink(%s, %s)"),
 115                             (int) name, (int) lockname);
 116                         fprintf(stderr, "%s", msg);
 117                         return errno;
 118                 }

 120                 counter = 0;
 121                 for (;;) {
 122                         sleep(1);
 123                         r = lstat(lockname, &statb);
 124                         if (r == -1) {
 125                                 /*
 126                                  * The lock must have just gone away - try
```

```
 127                                     * again.
 128                                     */
 129                                    break;
 130                            }

 132                            if ((counter > 5) && (!printed_warning)) {
 133                                    /* Print waiting message after 5 secs */
 134 #if defined(SUN5_0) || defined(HP_UX) || defined(linux)
 135                                    (void) getcwd(msg, MAXPATHLEN);
 136 #else
 137                                    (void) getwd(msg);
 138 #endif
 139                                    fprintf(stderr,
 140                                            catgets(libmksdmsi18n_catd, 1, 162, "fil
 141                                            name);
 142                                    fprintf(stderr,
 143                                            catgets(libmksdmsi18n_catd, 1, 163, "fil
 144                                            lockname);
 145                                    fprintf(stderr,
 146                                            catgets(libmksdmsi18n_catd, 1, 144, "Cur
 147                                            msg);

 149                                    printed_warning = 1;
 150                            }

 152                            if (++counter > timeout ) {
 153                                    /*
 154                                     * Waited enough - return an error..
 155                                     */
 156                                    return EEXIST;
 157                            }
 158                    }
 159            }
 160            /* NOTREACHED */
 161 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
    9525 Wed May 20 11:22:30 2015
new/usr/src/cmd/make/lib/vroot/report.cc
make: fix GCC warnings
**********************************************************
______unchanged_portion_omitted_

 308 void
 309 report_dependency(const char *name)
 309 report_dependency(register char *name)
 310 {
 311         register char   *filename;
 312         char            buffer[MAXPATHLEN+1];
 313         register char   *p;
 314         register char   *p2;
 315         char            nse_depinfo_file[MAXPATHLEN];

 317         if (report_file == NULL) {
 318                 if ((filename= getenv(SUNPRO_DEPENDENCIES)) == NULL) {
 319                         report_file = (FILE *)-1;
 320                         return;
 321                 }
 322                 if (strlen(filename) == 0) {
 323                         report_file = (FILE *)-1;
 324                         return;
 325                 }
 326                 (void)strcpy(buffer, name);
 327                 name = buffer;
 328                 p = strchr(filename, ' ');
 329                 if(p) {
 330                         *p= 0;
 331                 } else {
 332                         report_file = (FILE *)-1;
 333                         return;
 334                 }
 335                 if ((report_file= fopen(filename, "a")) == NULL) {
 336                         if ((report_file= fopen(filename, "w")) == NULL) {
 337                                 report_file= (FILE *)-1;
 338                                 return;
 339                         }
 340                 }
 341 #if defined(SUN5_0) || defined(HP_UX) || defined(linux)
 342                 atexit(close_report_file);
 343 #else
 344                 (void)on_exit(close_report_file, (char *)report_file);
 345 #endif
 346                 if ((p2= strchr(p+1, ' ')) != NULL)
 347                         *p2= 0;
 348                 target_being_reported_for= (char *)malloc((unsigned)(strlen(p+1)
 349                 (void)strcpy(target_being_reported_for, p+1);
 350                 (void)fputs(p+1, report_file);
 351                 (void)fputs(":", report_file);
 352                 *p= ' ';
 353                 if (p2 != NULL)
 354                         *p2= ' ';
 355         }
 356         if (report_file == (FILE *)-1)
 357                 return;
 358         (void)fputs(name, report_file);
 359         (void)fputs(" ", report_file);
 360 }
______unchanged_portion_omitted_
```

```
**********************************************************
    9177 Wed May 20 11:22:30 2015
new/usr/src/cmd/make/lib/vroot/vroot.cc
make: fix GCC warnings
**********************************************************
```

```
/*
 * 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 2004 Sun Microsystems, Inc. All rights reserved.
 * Use is subject to license terms.
 */


#include <stdlib.h>
#include <string.h>

#include <vroot/vroot.h>
#include <vroot/args.h>

#include <string.h>
#include <sys/param.h>
#include <sys/file.h>

#include <avo/intl.h>   /* for NOCATGETS */

typedef struct {
        short           init;
        pathpt          vector;
        const char      *env_var;
        char            *env_var;
} vroot_patht;
```

_____unchanged_portion_omitted_

```
static vroot_datat      vroot_data= {
        { 0, NULL, NOCATGETS("VIRTUAL_ROOT")},
        { 0, NULL, NOCATGETS("PATH")},
        "", NULL, NULL, NULL, 0, 1};

void
add_dir_to_path(const char *path, register pathpt *pointer, register int positio
add_dir_to_path(register char *path, register pathpt *pointer, register int posi
{
        register int            size= 0;
        register int            length;
        register char           *name;
        register pathcellpt     p;
        pathpt                  new_path;
```



```
        if (*pointer != NULL) {
                for (p= &((*pointer)[0]); p->path != NULL; p++, size++);
                if (position < 0)
                        position= size;}
        else
                if (position < 0)
                        position= 0;
        if (position >= size) {
                new_path= (pathpt)calloc((unsigned)(position+2), sizeof(pathcell
                if (*pointer != NULL) {
                        memcpy((char *)new_path,(char *)(*pointer),  size*sizeof
                        free((char *)(*pointer));};
                *pointer= new_path;};
        length= strlen(path);
        name= (char *)malloc((unsigned)(length+1));
        (void)strcpy(name, path);
        if ((*pointer)[position].path != NULL)
                free((*pointer)[position].path);
        (*pointer)[position].path= name;
        (*pointer)[position].length= length;
}
```

_____unchanged_portion_omitted_

```
const char *
char *
get_vroot_name(void)
{
        return(vroot_data.vroot.env_var);
}

const char *
char *
get_path_name(void)
{
        return(vroot_data.path.env_var);
}
```

_____unchanged_portion_omitted_