****
23294 Wed May 20 11:27:31 2015
new/usr/src/cmd/make/bin/ar.cc
make: unifdef for other OSes (undefined)
****
_____unchanged_portion_omitted_

```
 57 #if defined(linux)
 58 #include <ctype.h>              /* isspace */
 59 #else
 57 #include <unistd.h>             /* close() */
 61 #endif


 60 /*
 61  * Defined macros
 62  */
 63 #ifndef S5EMUL
 64 #undef BITSPERBYTE
 65 #define BITSPERBYTE     8
 66 #endif

 68 /*
 69  * Defines for all the different archive formats.  See next comment
 70  * block for justification for not using <ar.h>s versions.
 71  */
 72 #define AR_5_MAGIC              "<ar>"          /* 5.0 format magic string */
 73 #define AR_5_MAGIC_LENGTH       4               /* 5.0 format string length */

 75 #define AR_PORT_MAGIC           "!<arch>\n"     /* Port. (6.0) magic string */
 76 #define AR_PORT_MAGIC_LENGTH    8               /* Port. (6.0) string length */
 77 #define AR_PORT_END_MAGIC       "'\n"           /* Port. (6.0) end of header */
 78 #define AR_PORT_WORD            4               /* Port. (6.0) 'word' length */

 80 /*
 81  * typedefs & structs
 82  */
 83 /*
 84  * These are the archive file headers for the formats.  Note
 85  * that it really doesnt matter if these structures are defined
 86  * here.  They are correct as of the respective archive format
 87  * releases.  If the archive format is changed, then since backwards
 88  * compatability is the desired behavior, a new structure is added
 89  * to the list.
 90  */
 91 typedef struct {        /* 5.0 ar header format: vax family; 3b family */
 92         char                    ar_magic[AR_5_MAGIC_LENGTH];    /* AR_5_MAGIC*/
 93         char                    ar_name[16];    /* Space terminated */
 94         char                    ar_date[AR_PORT_WORD];  /* sgetl() accessed */
 95         char                    ar_syms[AR_PORT_WORD];  /* sgetl() accessed */
 96 }                       Arh_5;
```

_____unchanged_portion_omitted_

```
**********************************************************
   15297 Wed May 20 11:27:32 2015
new/usr/src/cmd/make/bin/dist.cc
make: unifdef for other OSes (undefined)
**********************************************************
```

```
   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 #ifdef DISTRIBUTED
  27 /*
  28  *      dist.cc
  29  *
  30  *      Deal with the distributed processing
  31  */

  33 #include <avo/err.h>
  34 #include <avo/find_dir.h>
  35 #include <avo/util.h>
  36 #include <dm/Avo_AcknowledgeMsg.h>
  37 #include <dm/Avo_DoJobMsg.h>
  38 #include <dm/Avo_JobResultMsg.h>
  39 #include <mk/defs.h>
  40 #include <mksh/misc.h>          /* getmem() */
  41 #include <rw/pstream.h>
  42 #include <rw/queuecol.h>
  43 #include <rw/xdrstrea.h>
  44 #include <signal.h>
  45 #ifdef linux
  46 #include <sstream>
  47 using namespace std;
  48 #else
  45 #include <strstream.h>
  50 #endif
  46 #include <sys/stat.h>           /* stat() */
  47 #include <sys/types.h>
  48 #include <sys/wait.h>
  49 #include <unistd.h>
  50 #include <errno.h>

  52 /*
  53  * Defined macros
  54  */

  56 #define AVO_BLOCK_INTERUPTS sigfillset(&newset) ; \
```



```
  57         sigprocmask(SIG_SETMASK, &newset, &oldset)
  58
  59 #define AVO_UNBLOCK_INTERUPTS \
  60         sigprocmask(SIG_SETMASK, &oldset, &newset)


  63 /*
  64  * typedefs & structs
  65  */

  67 /*
  68  * Static variables
  69  */
  70 int             dmake_ifd;
  71 FILE*           dmake_ifp;
  72 XDR             xdrs_in;

  74 int             dmake_ofd;
  75 FILE*           dmake_ofp;
  76 XDR             xdrs_out;

  78 // These instances are required for the RWfactory.
  79 static Avo_JobResultMsg dummyJobResultMsg;
  80 static Avo_AcknowledgeMsg dummyAcknowledgeMsg;
  81 static int firstAcknowledgeReceived = 0;
  82
  83 int             rxmPid = 0;

  85 /*
  86  * File table of contents
  87  */
  88 static void     set_dmake_env_vars(void);

  90 /*
  91  * void
  92  * startup_rxm(void)
  93  *
  94  * When startup_rxm() is called, read_command_options() and
  95  * read_files_and_state() have already been read, so DMake
  96  * will now know what options to pass to rxm.
  97  *
  98  * rxm
  99  *      [ -k ] [ -n ]
 100  *      [ -c <dmake_rcfile> ]
 101  *      [ -g <dmake_group> ]
 102  *      [ -j <dmake_max_jobs> ]
 103  *      [ -m <dmake_mode> ]
 104  *      [ -o <dmake_odir> ]
 105  *     <read_fd> <write_fd>
 106  *
 107  * rxm will, among other things, read the rc file.
 108  *
 109  */
 110 void
 111 startup_rxm(void)
 112 {
 113         Name            dmake_name;
 114         Name            dmake_value;
 115         Avo_err         *find_run_dir_err;
 116         int             pipe1[2], pipe2[2];
 117         Property        prop;
 118         char            *run_dir;
 119         char            rxm_command[MAXPATHLEN];
 120         int             rxm_debug = 0;

 122         int             length;
```

```
123         char *          env;

125         firstAcknowledgeReceived = 0;
126         /*
127          * Create two pipes, one for dmake->rxm, and one for rxm->dmake.
128          * pipe1 is dmake->rxm,
129          * pipe2 is rxm->dmake.
130          */
131         if ((pipe(pipe1) < 0) || (pipe(pipe2) < 0)) {
132                 fatal(catgets(catd, 1, 245, "pipe() failed: %s"), errmsg(errno))
133         }

135         set_dmake_env_vars();

137         if ((rxmPid = fork()) < 0) {    /* error */
138                 fatal(catgets(catd, 1, 246, "fork() failed: %s"), errmsg(errno))
139         } else if (rxmPid > 0) {        /* parent, dmake */
140                 dmake_ofd = pipe1[1];   // write side of pipe
141                 if (!(dmake_ofp = fdopen(dmake_ofd, "a"))) {
142                         fatal(catgets(catd, 1, 247, "fdopen() failed: %s"), errm
143                 }
144                 xdrstdio_create(&xdrs_out, dmake_ofp, XDR_ENCODE);

146                 dmake_ifd = pipe2[0];   // read side of pipe
147                 if (!(dmake_ifp = fdopen(dmake_ifd, "r"))) {
148                         fatal(catgets(catd, 1, 248, "fdopen() failed: %s"), errm
149                 }
150                 xdrstdio_create(&xdrs_in, dmake_ifp, XDR_DECODE);

152                 close(pipe1[0]);        // read side
153                 close(pipe2[1]);        // write side
154         } else {                        /* child, rxm */
155                 close(pipe1[1]);        // write side
156                 close(pipe2[0]);        // read side

158                 /* Find the run directory of dmake, for rxm. */
159                 find_run_dir_err = avo_find_run_dir(&run_dir);
160                 if (find_run_dir_err) {
161                         delete find_run_dir_err;
162                         /* Use the path to find rxm. */
163                         (void) sprintf(rxm_command, NOCATGETS("rxm"));
164                 } else {
165                         /* Use the run dir of dmake for rxm. */
166                         (void) sprintf(rxm_command, NOCATGETS("%s/rxm"), run_dir
167                 }

169                 if (continue_after_error) {
170                         (void) strcat(rxm_command, NOCATGETS(" -k"));
171                 }
172                 if (do_not_exec_rule) {
173                         (void) strcat(rxm_command, NOCATGETS(" -n"));
174                 }
175                 if (rxm_debug) {
176                         (void) strcat(rxm_command, NOCATGETS(" -S"));
177                 }
178                 if (send_mtool_msgs) {
179                         (void) sprintf(&rxm_command[strlen(rxm_command)],
180                                         NOCATGETS(" -O %d"),
181                                         mtool_msgs_fd);
182                 }
183                 MBSTOWCS(wcs_buffer, NOCATGETS("DMAKE_RCFILE"));
184                 dmake_name = GETNAME(wcs_buffer, FIND_LENGTH);
185                 if (((prop = get_prop(dmake_name->prop, macro_prop)) != NULL) &&
186                     ((dmake_value = prop->body.macro.value) != NULL)) {
187                         (void) sprintf(&rxm_command[strlen(rxm_command)],
188                                         NOCATGETS(" -c %s"),
```



```
189                                         dmake_value->string_mb);
190                 } else {
191                         length = 2 + strlen(NOCATGETS("DMAKE_RCFILE"));
192                         env = getmem(length);
193                         (void) sprintf(env,
194                                         "%s=",
195                                         NOCATGETS("DMAKE_RCFILE"));
196                         (void) putenv(env);
197                 }
198                 MBSTOWCS(wcs_buffer, NOCATGETS("DMAKE_GROUP"));
199                 dmake_name = GETNAME(wcs_buffer, FIND_LENGTH);
200                 if (((prop = get_prop(dmake_name->prop, macro_prop)) != NULL) &&
201                     ((dmake_value = prop->body.macro.value) != NULL)) {
202                         (void) sprintf(&rxm_command[strlen(rxm_command)],
203                                         NOCATGETS(" -g %s"),
204                                         dmake_value->string_mb);
205                 } else {
206                         length = 2 + strlen(NOCATGETS("DMAKE_GROUP"));
207                         env = getmem(length);
208                         (void) sprintf(env,
209                                         "%s=",
210                                         NOCATGETS("DMAKE_GROUP"));
211                         (void) putenv(env);
212                 }
213                 MBSTOWCS(wcs_buffer, NOCATGETS("DMAKE_MAX_JOBS"));
214                 dmake_name = GETNAME(wcs_buffer, FIND_LENGTH);
215                 if (((prop = get_prop(dmake_name->prop, macro_prop)) != NULL) &&
216                     ((dmake_value = prop->body.macro.value) != NULL)) {
217                         (void) sprintf(&rxm_command[strlen(rxm_command)],
218                                         NOCATGETS(" -j %s"),
219                                         dmake_value->string_mb);
220                 } else {
221                         length = 2 + strlen(NOCATGETS("DMAKE_MAX_JOBS"));
222                         env = getmem(length);
223                         (void) sprintf(env,
224                                         "%s=",
225                                         NOCATGETS("DMAKE_MAX_JOBS"));
226                         (void) putenv(env);
227                 }
228                 MBSTOWCS(wcs_buffer, NOCATGETS("DMAKE_MODE"));
229                 dmake_name = GETNAME(wcs_buffer, FIND_LENGTH);
230                 if (((prop = get_prop(dmake_name->prop, macro_prop)) != NULL) &&
231                     ((dmake_value = prop->body.macro.value) != NULL)) {
232                         (void) sprintf(&rxm_command[strlen(rxm_command)],
233                                         NOCATGETS(" -m %s"),
234                                         dmake_value->string_mb);
235                 } else {
236                         length = 2 + strlen(NOCATGETS("DMAKE_MODE"));
237                         env = getmem(length);
238                         (void) sprintf(env,
239                                         "%s=",
240                                         NOCATGETS("DMAKE_MODE"));
241                         (void) putenv(env);
242                 }
243                 MBSTOWCS(wcs_buffer, NOCATGETS("DMAKE_ODIR"));
244                 dmake_name = GETNAME(wcs_buffer, FIND_LENGTH);
245                 if (((prop = get_prop(dmake_name->prop, macro_prop)) != NULL) &&
246                     ((dmake_value = prop->body.macro.value) != NULL)) {
247                         (void) sprintf(&rxm_command[strlen(rxm_command)],
248                                         NOCATGETS(" -o %s"),
249                                         dmake_value->string_mb);
250                 } else {
251                         length = 2 + strlen(NOCATGETS("DMAKE_ODIR"));
252                         env = getmem(length);
253                         (void) sprintf(env,
254                                         "%s=",
```

```
255                                         NOCATGETS("DMAKE_ODIR"));
256                         (void) putenv(env);
257                 }

259                 (void) sprintf(&rxm_command[strlen(rxm_command)],
260                                NOCATGETS(" %d %d"),
261                                pipe1[0], pipe2[1]);
267 #ifdef linux
268                 execl(NOCATGETS("/bin/sh"),
269 #else
262                 execl(NOCATGETS("/usr/bin/sh"),
271 #endif
263                       NOCATGETS("sh"),
264                       NOCATGETS("-c"),
265                       rxm_command,
266                       (char *)NULL);
267                 _exit(127);
268         }
269 }
```

_____unchanged_portion_omitted_

```
**********************************************************
  105078 Wed May 20 11:27:33 2015
new/usr/src/cmd/make/bin/doname.cc
make: unifdef for other OSes (undefined)
**********************************************************
```

```
   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 2006 Sun Microsystems, Inc. All rights reserved.
  23  * Use is subject to license terms.
  24  */

  26 /*
  27  *      doname.c
  28  *
  29  *      Figure out which targets are out of date and rebuild them
  30  */

  32 /*
  33  * Included files
  34  */
  35 #include <avo/avo_alloca.h>             /* alloca() */
  36 #if defined(TEAMWARE_MAKE_CMN)
  37 #include <avo/util.h>           /* avo_get_user(), avo_hostname() */
  38 #endif

  40 #if defined(DISTRIBUTED) || defined(MAKETOOL) /* tolik */
  41 #       include <avo/strings.h> /* AVO_STRDUP() */
  42 #       include <dm/Avo_MToolJobResultMsg.h>
  43 #       include <dm/Avo_MToolJobStartMsg.h>
  44 #       include <dm/Avo_MToolRsrcInfoMsg.h>
  45 #       include <dm/Avo_macro_defs.h> /* AVO_BLOCK_INTERUPTS & AVO_UNBLOCK_INTER
  46 #       include <dmthread/Avo_ServerState.h>
  47 #       include <rw/pstream.h>
  48 #       include <rw/xdrstrea.h>
  49 #endif

  51 #include <fcntl.h>
  52 #include <mk/defs.h>
  53 #include <mksh/i18n.h>          /* get_char_semantics_value() */
  54 #include <mksh/macro.h>         /* getvar(), expand_value() */
  55 #include <mksh/misc.h>          /* getmem() */
  56 #include <poll.h>

  58 #ifdef PARALLEL
  59 #       include <rx/api.h>
  60 #endif
```



```
  62 #include <signal.h>

  64 #ifndef HP_UX
  64 #       include <stropts.h>
  66 #endif

  66 #include <sys/errno.h>
  67 #include <sys/stat.h>
  68 #include <sys/types.h>
  69 #include <sys/utsname.h>        /* uname() */
  70 #include <sys/wait.h>
  71 #include <unistd.h>             /* close() */

  73 /*
  74  * Defined macros
  75  */
  76 #ifndef PARALLEL
  77 #       define LOCALHOST "localhost"
  78 #endif

  80 #define MAXRULES 100

  82 #if defined(DISTRIBUTED) || defined(MAKETOOL) /* tolik */
  83 #define SEND_MTOOL_MSG(cmds) \
  84         if (send_mtool_msgs) { \
  85                 cmds \
  86         }
  87 #else
  88 #define SEND_MTOOL_MSG(cmds)
  89 #endif

  91 // Sleep for .1 seconds between stat()'s
  92 const int       STAT_RETRY_SLEEP_TIME = 100000;

  94 /*
  95  * typedefs & structs
  96  */

  98 /*
  99  * Static variables
 100  */
 101 static char     hostName[MAXNAMELEN] = "";
 102 static char     userName[MAXNAMELEN] = "";

 104 #if defined(DISTRIBUTED) || defined(MAKETOOL) /* tolik */
 105         static FILE     *mtool_msgs_fp;
 106         static XDR      xdrs;
 107         static int      sent_rsrc_info_msg = 0;
 108 #endif

 110 static int      second_pass = 0;

 112 /*
 113  * File table of contents
 114  */
 115 extern  Doname          doname_check(register Name target, register Boolean do_g
 116 extern  Doname          doname(register Name target, register Boolean do_get, re
 117 static  Boolean         check_dependencies(Doname *result, Property line, Boolea
 118 void            dynamic_dependencies(Name target);
 119 static  Doname          run_command(register Property line, Boolean print_machin
 120 extern  Doname          execute_serial(Property line);
 121 extern  Name            vpath_translation(register Name cmd);
 122 extern  void            check_state(Name temp_file_name);
 123 static  void            read_dependency_file(register Name filename);
 124 static  void            check_read_state_file(void);
 125 static  void            do_assign(register Name line, register Name target);
```

```
126 static  void            build_command_strings(Name target, register Property lin
127 static  Doname          touch_command(register Property line, register Name targ
128 extern  void            update_target(Property line, Doname result);
129 static  Doname          sccs_get(register Name target, register Property *comman
130 extern  void            read_directory_of_file(register Name file);
131 static  void            add_pattern_conditionals(register Name target);
132 extern  void            set_locals(register Name target, register Property old_l
133 extern  void            reset_locals(register Name target, register Property old
134 extern  Boolean         check_auto_dependencies(Name target, int auto_count, Nam
135 static  void            delete_query_chain(Chain ch);

137 // From read2.cc
138 extern  Name            normalize_name(register wchar_t *name_string, register i


141 #if defined(DISTRIBUTED) || defined(MAKETOOL) /* tolik */
142         static void             append_job_result_msg(Avo_MToolJobResultMsg *job
143         static int              pollResults(char *outFn, char *errFn, char *host
144         static void             pollResultsAction(char *outFn, char *errFn);
145         static void             rxmGetNextResultsBlock(int fd);
146         static int              us_sleep(unsigned int nusecs);
147         extern "C" void         Avo_PollResultsAction_Sigusr1Handler(int foo);
148 #endif

150 /*
151  * DONE.
152  *
153  *      doname_check(target, do_get, implicit, automatic)
154  *
155  *      Will call doname() and then inspect the return value
156  *
157  *      Return value:
158  *                              Indication if the build failed or not
159  *
160  *      Parameters:
161  *              target          The target to build
162  *              do_get          Passed thru to doname()
163  *              implicit        Passed thru to doname()
164  *              automatic       Are we building a hidden dependency?
165  *
166  *      Global variables used:
167  *              build_failed_seen       Set if -k is on and error occurs
168  *              continue_after_error    Indicates that -k is on
169  *              report_dependencies     No error msg if -P is on
170  */
171 Doname
172 doname_check(register Name target, register Boolean do_get, register Boolean imp
173 {
174         int first_time = 1;
175         (void) fflush(stdout);
176 try_again:
177         switch (doname(target, do_get, implicit, automatic)) {
178         case build_ok:
179                 second_pass = 0;
180                 return build_ok;
181         case build_running:
182                 second_pass = 0;
183                 return build_running;
184         case build_failed:
185                 if (!continue_after_error) {
186                         fatal(catgets(catd, 1, 13, "Target `%s' not remade becau
187                               target->string_mb);
188                 }
189                 build_failed_seen = true;
190                 second_pass = 0;
191                 return build_failed;
```



```
192         case build_dont_know:
193                 /*
194                  * If we can't figure out how to build an automatic
195                  * (hidden) dependency, we just ignore it.
196                  * We later declare the target to be out of date just in
197                  * case something changed.
198                  * Also, don't complain if just reporting the dependencies
199                  * and not building anything.
200                  */
201                 if (automatic || (report_dependencies_level > 0)) {
202                         second_pass = 0;
203                         return build_dont_know;
204                 }
205                 if(first_time) {
206                         first_time = 0;
207                         second_pass = 1;
208                         goto try_again;
209                 }
210                 second_pass = 0;
211                 if (continue_after_error && !svr4) {
212                         warning(catgets(catd, 1, 14, "Don't know how to make tar
213                                 target->string_mb);
214                         build_failed_seen = true;
215                         return build_failed;
216                 }
217                 fatal(catgets(catd, 1, 15, "Don't know how to make target `%s'")
218                 break;
219         }
220 #ifdef lint
221         return build_failed;
222 #endif
223 }
______unchanged_portion_omitted_

3546 // Continuously poll and show the results of remotely executing a job,
3547 // i.e., output the stdout and stderr files.

3549 static int
3550 pollResults(char *outFn, char *errFn, char *hostNm)
3551 {
3552         int             child;

3554         child = fork();
3555         switch (child) {
3556         case -1:
3557                 break;
3558         case 0:
3559                 enable_interrupt((void (*) (int))SIG_DFL);
3562 #ifdef linux
3563                 (void) signal(SIGUSR1, Avo_PollResultsAction_Sigusr1Handler);
3564 #else
3560                 (void) sigset(SIGUSR1, Avo_PollResultsAction_Sigusr1Handler);
3566 #endif
3561                 pollResultsAction(outFn, errFn);

3563                 exit(0);
3564                 break;
3565         default:
3566                 break;
3567         }
3568         return child;
3569 }
______unchanged_portion_omitted_

3581 static void
3582 pollResultsAction(char *outFn, char *errFn)
```

```
3583 {
3584         int                     fd;
3585         time_t                  file_time = 0;
3586         long                    file_time_nsec = 0;
3587         struct stat             statbuf;
3588         int                     stat_rc;

3590         // Keep stat'ing until file exists.
3591         while (((stat_rc = stat(outFn, &statbuf)) != 0) &&
3592                (errno == ENOENT) &&
3593                !pollResultsActionTimeToFinish) {
3594                 us_sleep(STAT_RETRY_SLEEP_TIME);
3595         }
3596         // The previous stat() could be failed due to EINTR
3597         // So one more try is needed
3598         if (stat_rc != 0 && stat(outFn, &statbuf) != 0) {
3599                 // stat() failed
3600                 warning(NOCATGETS("Internal error: stat(\"%s\", ...) failed: %s\
3601                         outFn, strerror(errno));
3602                 exit(1);
3603         }

3605         if ((fd = open(outFn, O_RDONLY)) < 0
3606                 && (errno != EINTR || (fd = open(outFn, O_RDONLY)) < 0)) {
3607                 // open() failed
3608                 warning(NOCATGETS("Internal error: open(\"%s\", O_RDONLY) failed
3609                         outFn, strerror(errno));
3610                 exit(1);
3611         }

3613         while (!pollResultsActionTimeToFinish && stat(outFn, &statbuf) == 0) {
3620 #ifdef linux
3621                 if ((statbuf.st_mtime > file_time)
3622                    ) {
3623                         file_time = statbuf.st_mtime;
3624                         rxmGetNextResultsBlock(fd);
3625                 }
3626 #else
3614                 if ((statbuf.st_mtim.tv_sec > file_time) ||
3615                     ((statbuf.st_mtim.tv_sec == file_time) &&
3616                     (statbuf.st_mtim.tv_nsec > file_time_nsec))
3617                    ) {
3618                         file_time = statbuf.st_mtim.tv_sec;
3619                         file_time_nsec = statbuf.st_mtim.tv_nsec;
3620                         rxmGetNextResultsBlock(fd);
3621                 }
3635 #endif
3622                 us_sleep(STAT_RETRY_SLEEP_TIME);
3623         }
3624         // Check for the rest of output
3625         rxmGetNextResultsBlock(fd);

3627         (void) close(fd);
3628 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   18431 Wed May 20 11:27:33 2015
new/usr/src/cmd/make/bin/files.cc
make: unifdef for other OSes (undefined)
**********************************************************
```

```
/*
 * 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 2003 Sun Microsystems, Inc. All rights reserved.
 * Use is subject to license terms.
 */

/*
 *      files.c
 *
 *      Various file related routines:
 *              Figure out if file exists
 *              Wildcard resolution for directory reader
 *              Directory reader
 */



/*
 * Included files
 */
#include <dirent.h>             /* opendir() */
#include <errno.h>              /* errno */
#include <mk/defs.h>
#include <mksh/macro.h>         /* getvar() */
#include <mksh/misc.h>          /* get_prop(), append_prop() */
#include <sys/stat.h>           /* lstat() */

/*
 * Defined macros
 */

/*
 * typedefs & structs
 */

/*
 * Static variables
 */

/*
 * File table of contents
 */
extern  timestruc_t&    exists(register Name target);
```



```
extern  void            set_target_stat(register Name target, struct stat buf);
static  timestruc_t&    vpath_exists(register Name target);
static  Name            enter_file_name(wchar_t *name_string, wchar_t *library);
static  Boolean         star_match(register char *string, register char *pattern
static  Boolean         amatch(register wchar_t *string, register wchar_t *patte

/*
 *      exists(target)
 *
 *      Figure out the timestamp for one target.
 *
 *      Return value:
 *                              The time the target was created
 *
 *      Parameters:
 *              target          The target to check
 *
 *      Global variables used:
 *              debug_level     Should we trace the stat call?
 *              recursion_level Used for tracing
 *              vpath_defined   Was the variable VPATH defined in environment?
 */
timestruc_t&
exists(register Name target)
{
        struct stat             buf;
        register int            result;

        /* We cache stat information. */
        if (target->stat.time != file_no_time) {
                return target->stat.time;
        }

        /*
         * If the target is a member, we have to extract the time
         * from the archive.
         */
        if (target->is_member &&
            (get_prop(target->prop, member_prop) != NULL)) {
                return read_archive(target);
        }

        if (debug_level > 1) {
                (void) printf(NOCATGETS("%*sstat(%s)\n"),
                                recursion_level,
                                "",
                                target->string_mb);
        }

        result = lstat_vroot(target->string_mb, &buf, NULL, VROOT_DEFAULT);
        if ((result != -1) && ((buf.st_mode & S_IFMT) == S_IFLNK)) {
                /*
                 * If the file is a symbolic link, we remember that
                 * and then we get the status for the refd file.
                 */
                target->stat.is_sym_link = true;
                result = stat_vroot(target->string_mb, &buf, NULL, VROOT_DEFAULT
        } else {
                target->stat.is_sym_link = false;
        }

        if (result < 0) {
                target->stat.time = file_doesnt_exist;
                target->stat.stat_errno = errno;
                if ((errno == ENOENT) &&
                    vpath_defined &&
```

```
128 /* azv, fixing bug 1262942, VPATH works with a leaf name
129  * but not a directory name.
130  */
131                     (target->string_mb[0] != (int) slash_char) ) {
132 /* BID_1214655 */
133 /* azv */
134                         vpath_exists(target);
135                         // return vpath_exists(target);
136                 }
137         } else {
138                 /* Save all the information we need about the file */
139                 target->stat.stat_errno = 0;
140                 target->stat.is_file = true;
141                 target->stat.mode = buf.st_mode & 0777;
142                 target->stat.size = buf.st_size;
143                 target->stat.is_dir =
144                   BOOLEAN((buf.st_mode & S_IFMT) == S_IFDIR);
145                 if (target->stat.is_dir) {
146                         target->stat.time = file_is_dir;
147                 } else {
148                         /* target->stat.time = buf.st_mtime; */
149 /* BID_1129806 */
150 /* vis@nbsp.nsk.su */
151 #if defined(linux)
152                         timestruc_t ttime = { buf.st_mtime, 0 };
153                         target->stat.time = MAX(ttime, file_min_time);
154 #else
151                         target->stat.time = MAX(buf.st_mtim, file_min_time);
156 #endif
152                 }
153         }
154         if ((target->colon_splits > 0) &&
155             (get_prop(target->prop, time_prop) == NULL)) {
156                 append_prop(target, time_prop)->body.time.time =
157                   target->stat.time;
158         }
159         return target->stat.time;
160 }

162 /*
163  *      set_target_stat( target, buf)
164  *
165  *      Called by exists() to set some stat fields in the Name structure
166  *      to those read by the stat_vroot() call (from disk).
167  *
168  *      Parameters:
169  *              target          The target whose stat field is set
170  *              buf             stat values (on disk) of the file
171  *                              represented by target.
172  */
173 void
174 set_target_stat(register Name target, struct stat buf)
175 {
176         target->stat.stat_errno = 0;
177         target->stat.is_file = true;
178         target->stat.mode = buf.st_mode & 0777;
179         target->stat.size = buf.st_size;
180         target->stat.is_dir =
181           BOOLEAN((buf.st_mode & S_IFMT) == S_IFDIR);
182         if (target->stat.is_dir) {
183                 target->stat.time = file_is_dir;
184         } else {
185                 /* target->stat.time = buf.st_mtime; */
186 /* BID_1129806 */
187 /* vis@nbsp.nsk.su */
193 #if defined(linux)
```



```
194                 timestruc_t ttime = { buf.st_mtime, 0 };
195                 target->stat.time = ttime;
196 #else
188                 target->stat.time = MAX(buf.st_mtim, file_min_time);
198 #endif
189         }
190 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
    5326 Wed May 20 11:27:34 2015
new/usr/src/cmd/make/bin/globals.cc
make: unifdef for other OSes (undefined)
**********************************************************
```

```
/*
 * 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;
```

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

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



```
191         Name            temp_file_name;
192         short           temp_file_number;
193         time_t          timing_start;
194         wchar_t         *top_level_target;
195         Boolean         touch;                          /* '-t' */
196         Boolean         trace_reader;                   /* '-D' */
197         Boolean         build_unconditional;            /* '-u' */
198         pathpt          vroot_path = VROOT_DEFAULT;
199         Name            wait_name;
200         wchar_t         wcs_buffer2[MAXPATHLEN];
201         wchar_t         *wcs_ptr;
202         wchar_t         *wcs_ptr2;
203         nl_catd         catd;
204         long int        hostid;

206 /*
207  * File table of contents
208  */
```

```
**********************************************************
  101286 Wed May 20 11:27:34 2015
new/usr/src/cmd/make/bin/main.cc
make: unifdef for other OSes (undefined)
**********************************************************
```

```
   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 2006 Sun Microsystems, Inc. All rights reserved.
  23  * Use is subject to license terms.
  24  */

  26 /*
  27  *      main.cc
  28  *
  29  *      make program main routine plus some helper routines
  30  */
  31
  32 /*
  33  * Included files
  34  */
  35 #if defined(TEAMWARE_MAKE_CMN)
  36 #       include <avo/intl.h>
  37 #       include <avo/libcli.h>          /* libcli_init() */
  38 #       include <avo/cli_license.h>     /* avo_cli_get_license() */
  39 #       include <avo/find_dir.h>        /* avo_find_run_dir() */
  40 #       include <avo/version_string.h>
  41 #       include <avo/util.h>            /* avo_init() */
  42 #ifdef USE_DMS_CCR
  43 #       include <avo/usage_tracking.h>
  44 #else
  45 #       include <avo/cleanup.h>
  46 #endif
  47 #endif

  49 #if defined(TEAMWARE_MAKE_CMN)
  50 /* This is for dmake only (not for Solaris make).
  51  * Include code to check updates (dmake patches)
  52  */
  53 #ifdef _CHECK_UPDATE_H
  54 #include <libAU.h>
  55 #endif
  56 #endif

  58 #include <bsd/bsd.h>            /* bsd_signal() */

  60 #ifdef DISTRIBUTED
  61 #       include <dm/Avo_AcknowledgeMsg.h>
```



```
  62 #       include <rw/xdrstrea.h>
  63 #       include <dmrc/dmrc.h> /* dmakerc file processing */
  64 #endif

  66 #include <locale.h>             /* setlocale() */
  67 #include <mk/defs.h>
  68 #include <mksdmsi18n/mksdmsi18n.h>      /* libmksdmsi18n_init() */
  69 #include <mksh/macro.h>         /* getvar() */
  70 #include <mksh/misc.h>          /* getmem(), setup_char_semantics() */

  72 #if defined(TEAMWARE_MAKE_CMN)
  73 #ifdef USE_DMS_CCR
  74 #       include <pubdmsi18n/pubdmsi18n.h>       /* libpubdmsi18n_init() */
  75 #endif
  76 #endif

  78 #include <pwd.h>                /* getpwnam() */
  79 #include <setjmp.h>
  80 #include <signal.h>
  81 #include <stdlib.h>
  82 #include <sys/errno.h>          /* ENOENT */
  83 #include <sys/stat.h>           /* fstat() */
  84 #include <fcntl.h>              /* open() */

  86 #       include <sys/systeminfo.h>      /* sysinfo() */

  88 #include <sys/types.h>          /* stat() */
  89 #include <sys/wait.h>           /* wait() */
  90 #include <unistd.h>             /* execv(), unlink(), access() */
  91 #include <vroot/report.h>       /* report_dependency(), get_report_file() */

  93 // From read2.cc
  94 extern  Name            normalize_name(register wchar_t *name_string, register i

  96 // From parallel.cc
  97 #if defined(TEAMWARE_MAKE_CMN)
  98 #define MAXJOBS_ADJUST_RFE4694000

 100 #ifdef MAXJOBS_ADJUST_RFE4694000
 101 extern void job_adjust_fini();
 102 #endif /* MAXJOBS_ADJUST_RFE4694000 */
 103 #endif /* TEAMWARE_MAKE_CMN */

 105 #if defined(linux)
 106 #include <ctype.h>
 107 #endif

 106 /*
 107  * Defined macros
 108  */
 109 #define LD_SUPPORT_ENV_VAR      NOCATGETS("SGS_SUPPORT")
 110 #define LD_SUPPORT_MAKE_LIB     NOCATGETS("libmakestate.so.1")

 112 /*
 113  * typedefs & structs
 114  */

 116 /*
 117  * Static variables
 118  */
 119 static  char            *argv_zero_string;
 120 static  Boolean         build_failed_ever_seen;
 121 static  Boolean         continue_after_error_ever_seen; /* '-k' */
 122 static  Boolean         dmake_group_specified;          /* '-g' */
 123 static  Boolean         dmake_max_jobs_specified;       /* '-j' */
 124 static  Boolean         dmake_mode_specified;           /* '-m' */
```

```
125 static  Boolean         dmake_add_mode_specified;       /* '-x' */
126 static  Boolean         dmake_output_mode_specified;    /* '-x DMAKE_OUTPUT_MODE
127 static  Boolean         dmake_compat_mode_specified;    /* '-x SUN_MAKE_COMPAT_M
128 static  Boolean         dmake_odir_specified;           /* '-o' */
129 static  Boolean         dmake_rcfile_specified;         /* '-c' */
130 static  Boolean         env_wins;                       /* '-e' */
131 static  Boolean         ignore_default_mk;              /* '-r' */
132 static  Boolean         list_all_targets;               /* '-T' */
133 static  int             mf_argc;
134 static  char            **mf_argv;
135 static  Dependency_rec  not_auto_depen_struct;
136 static  Dependency      not_auto_depen = &not_auto_depen_struct;
137 static  Boolean         pmake_cap_r_specified;          /* '-R' */
138 static  Boolean         pmake_machinesfile_specified;   /* '-M' */
139 static  Boolean         stop_after_error_ever_seen;     /* '-S' */
140 static  Boolean         trace_status;                   /* '-p' */

142 #ifdef DMAKE_STATISTICS
143 static  Boolean         getname_stat = false;
144 #endif

146 #if defined(TEAMWARE_MAKE_CMN)
147         static  time_t          start_time;
148         static  int             g_argc;
149         static  char            **g_argv;
150 #ifdef USE_DMS_CCR
151         static  Avo_usage_tracking *usageTracking = NULL;
152 #else
153         static  Avo_cleanup     *cleanup = NULL;
154 #endif
155 #endif

157 /*
158  * File table of contents
159  */
160         extern "C" void         cleanup_after_exit(void);

162 #ifdef TEAMWARE_MAKE_CMN
163 extern "C" {
164         extern  void            dmake_exit_callback(void);
165         extern  void            dmake_message_callback(char *);
166 }
167 #endif

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

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

173 static  void            append_makeflags_string(Name, String);
174 static  void            doalarm(int);
175 static  void            enter_argv_values(int , char **, ASCII_Dyn_Array *);
176 static  void            make_targets(int, char **, Boolean);
177 static  int             parse_command_option(char);
178 static  void            read_command_options(int, char **);
179 static  void            read_environment(Boolean);
180 static  void            read_files_and_state(int, char **);
181 static  Boolean         read_makefile(Name, Boolean, Boolean, Boolean);
182 static  void            report_recursion(Name);
183 static  void            set_sgs_support(void);
184 static  void            setup_for_projectdir(void);
185 static  void            setup_makeflags_argv(void);
186 static  void            report_dir_enter_leave(Boolean entering);

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

190 #ifdef DISTRIBUTED
```



```
191         extern  int             dmake_ofd;
192         extern  FILE*           dmake_ofp;
193         extern  int             rxmPid;
194         extern  XDR             xdrs_out;
195 #endif
196 #ifdef TEAMWARE_MAKE_CMN
197         extern  char            verstring[];
198 #endif

200 jmp_buf jmpbuffer;
204 #if !defined(linux)
201 extern nl_catd catd;
206 #endif

203 /*
204  *      main(argc, argv)
205  *
206  *      Parameters:
207  *              argc                    You know what this is
208  *              argv                    You know what this is
209  *
210  *      Static variables used:
211  *              list_all_targets        make -T seen
212  *              trace_status            make -p seen
213  *
214  *      Global variables used:
215  *              debug_level             Should we trace make actions?
216  *              keep_state              Set if .KEEP_STATE seen
217  *              makeflags               The Name "MAKEFLAGS", used to get macro
218  *              remote_command_name     Name of remote invocation cmd ("on")
219  *              running_list            List of parallel running processes
220  *              stdout_stderr_same      true if stdout and stderr are the same
221  *              auto_dependencies       The Name "SUNPRO_DEPENDENCIES"
222  *              temp_file_directory     Set to the dir where we create tmp file
223  *              trace_reader            Set to reflect tracing status
224  *              working_on_targets      Set when building user targets
225  */
226 int
227 main(int argc, char *argv[])
228 {
229         /*
230          * cp is a -> to the value of the MAKEFLAGS env var,
231          * which has to be regular chars.
232          */
233         register char           *cp;
234         char                    make_state_dir[MAXPATHLEN];
235         Boolean                 parallel_flag = false;
236         char                    *prognameptr;
237         char                    *slash_ptr;
238         mode_t                  um;
239         int                     i;
240 #ifdef TEAMWARE_MAKE_CMN
241         struct itimerval        value;
242         char                    def_dmakerc_path[MAXPATHLEN];
243         Name                    dmake_name, dmake_name2;
244         Name                    dmake_value, dmake_value2;
245         Property                prop, prop2;
246         struct stat             statbuf;
247         int                     statval;
248 #endif

250 #ifndef PARALLEL
251         struct stat             out_stat, err_stat;
252 #endif
253         hostid = gethostid();
254 #ifdef TEAMWARE_MAKE_CMN
```

```
255         avo_get_user(NULL, NULL); // Initialize user name
256 #endif
257         bsd_signals();

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

266 #if defined (HP_UX) || defined(linux)
267        /* HP-UX users typically will not have NLSPATH set, and this binary
268         * requires that it be set.  On HP-UX 9.0x, /usr/lib/nls/%L/%N.cat is
269         * the path to set it to.
270         */

272        if (getenv(NOCATGETS("NLSPATH")) == NULL) {
273          putenv(NOCATGETS("NLSPATH=/usr/lib/nls/%L/%N.cat"));
274         }
275 #endif

262 #ifdef DMAKE_STATISTICS
263         if (getenv(NOCATGETS("DMAKE_STATISTICS"))) {
264                 getname_stat = true;
265         }
266 #endif



269         /*
270          * avo_init() sets the umask to 0.  Save it here and restore
271          * it after the avo_init() call.
272          */
273 #if defined(TEAMWARE_MAKE_CMN) || defined(MAKETOOL)
274         um = umask(0);
275         avo_init(argv[0]);
276         umask(um);

278 #ifdef USE_DMS_CCR
279         usageTracking = new Avo_usage_tracking(NOCATGETS("dmake"), argc, argv);
280 #else
281         cleanup = new Avo_cleanup(NOCATGETS("dmake"), argc, argv);
282 #endif
283 #endif

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

289 #ifdef _CHECK_UPDATE_H
290         /* This is for dmake only (not for Solaris make).
291          * Check (in background) if there is an update (dmake patch)
292          * and inform user
293          */
294         {
295                 Avo_err         *err;
296                 char            *dir;
297                 err = avo_find_run_dir(&dir);
298                 if (AVO_OK == err) {
299                         AU_check_update_service(NOCATGETS("Dmake"), dir);
300                 }
301         }
302 #endif /* _CHECK_UPDATE_H */
303 #endif

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


308 #if defined(TEAMWARE_MAKE_CMN) || defined(MAKETOOL)
309 /*
310  * I put libmksdmsi18n_init() under #ifdef because it requires avo_i18n_init()
```



```
311  * from avo_util library.
312  */
313         libmksdmsi18n_init();
314 #ifdef USE_DMS_CCR
315         libpubdmsi18n_init();
316 #endif
317 #endif


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

324 #ifdef TEAMWARE_MAKE_CMN
325         g_argc = argc;
326         g_argv = (char **) malloc((g_argc + 1) * sizeof(char *));
327         for (i = 0; i < argc; i++) {
328                 g_argv[i] = argv[i];
329         }
330         g_argv[i] = NULL;
331 #endif /* TEAMWARE_MAKE_CMN */

333         /*
334          * Set argv_zero_string to some form of argv[0] for
335          * recursive MAKE builds.
336          */

338         if (*argv[0] == (int) slash_char) {
339                 /* argv[0] starts with a slash */
340                 argv_zero_string = strdup(argv[0]);
341         } else if (strchr(argv[0], (int) slash_char) == NULL) {
342                 /* argv[0] contains no slashes */
343                 argv_zero_string = strdup(argv[0]);
344         } else {
345                 /*
346                  * argv[0] contains at least one slash,
347                  * but doesn't start with a slash
348                  */
349                 char    *tmp_current_path;
350                 char    *tmp_string;

352                 tmp_current_path = get_current_path();
353                 tmp_string = getmem(strlen(tmp_current_path) + 1 +
354                                     strlen(argv[0]) + 1);
355                 (void) sprintf(tmp_string,
356                                "%s/%s",
357                                tmp_current_path,
358                                argv[0]);
359                 argv_zero_string = strdup(tmp_string);
360                 retmem_mb(tmp_string);
361         }

363         /*
364          * The following flags are reset if we don't have the
365          * (.nse_depinfo or .make.state) files locked and only set
366          * AFTER the file has been locked. This ensures that if the user
367          * interrupts the program while file_lock() is waiting to lock
368          * the file, the interrupt handler doesn't remove a lock
369          * that doesn't belong to us.
370          */
371         make_state_lockfile = NULL;
372         make_state_locked = false;

374 #ifdef NSE
375         nse_depinfo_lockfile[0] = '\0';
376         nse_depinfo_locked = false;
```

```
377 #endif

379         /*
380          * look for last slash char in the path to look at the binary
381          * name. This is to resolve the hard link and invoke make
382          * in svr4 mode.
383          */

385         /* Sun OS make standart */
386         svr4 = false;
387         posix = false;
388         if(!strcmp(argv_zero_string, NOCATGETS("/usr/xpg4/bin/make"))) {
389                 svr4 = false;
390                 posix = true;
391         } else {
392                 prognameptr = strrchr(argv[0], '/');
393                 if(prognameptr) {
394                         prognameptr++;
395                 } else {
396                         prognameptr = argv[0];
397                 }
398                 if(!strcmp(prognameptr, NOCATGETS("svr4.make"))) {
399                         svr4 = true;
400                         posix = false;
401                 }
402         }
418 #if !defined(HP_UX) && !defined(linux)
403         if (getenv(USE_SVR4_MAKE) || getenv(NOCATGETS("USE_SVID"))){
404            svr4 = true;
405            posix = false;
406         }
423 #endif

408         /*
409          * Find the dmake_compat_mode: posix, sun, svr4, or gnu_style, .
410          */
411         char * dmake_compat_mode_var = getenv(NOCATGETS("SUN_MAKE_COMPAT_MODE"))
412         if (dmake_compat_mode_var != NULL) {
413                 if (0 == strcasecmp(dmake_compat_mode_var, NOCATGETS("GNU"))) {
414                         gnu_style = true;
415                 }
416                 //svr4 = false;
417                 //posix = false;
418         }

420         /*
421          * Temporary directory set up.
422          */
423         char * tmpdir_var = getenv(NOCATGETS("TMPDIR"));
424         if (tmpdir_var != NULL && *tmpdir_var == '/' && strlen(tmpdir_var) < MAX
425                 strcpy(mbs_buffer, tmpdir_var);
426                 for (tmpdir_var = mbs_buffer+strlen(mbs_buffer);
427                         *(--tmpdir_var) == '/' && tmpdir_var > mbs_buffer;
428                         *tmpdir_var = '\0');
429                 if (strlen(mbs_buffer) + 32 < MAXPATHLEN) { /* 32 = strlen("/dma
430                         sprintf(mbs_buffer2, NOCATGETS("%s/dmake.tst.%d.XXXXXX")
431                                 mbs_buffer, getpid());
432                         int fd = mkstemp(mbs_buffer2);
433                         if (fd >= 0) {
434                                 close(fd);
435                                 unlink(mbs_buffer2);
436                                 tmpdir = strdup(mbs_buffer);
437                         }
438                 }
439         }
```



```
441 #ifndef PARALLEL
442         /* find out if stdout and stderr point to the same place */
443         if (fstat(1, &out_stat) < 0) {
444                 fatal(catgets(catd, 1, 165, "fstat of standard out failed: %s"),
445         }
446         if (fstat(2, &err_stat) < 0) {
447                 fatal(catgets(catd, 1, 166, "fstat of standard error failed: %s"
448         }
449         if ((out_stat.st_dev == err_stat.st_dev) &&
450             (out_stat.st_ino == err_stat.st_ino)) {
451                 stdout_stderr_same = true;
452         } else {
453                 stdout_stderr_same = false;
454         }
455 #else
456         stdout_stderr_same = false;
457 #endif
458         /* Make the vroot package scan the path using shell semantics */
459         set_path_style(0);

461         setup_char_semantics();

463         setup_for_projectdir();

465         /*
466          * If running with .KEEP_STATE, curdir will be set with
467          * the connected directory.
468          */
469         (void) atexit(cleanup_after_exit);

471         load_cached_names();

473 /*
474  *      Set command line flags
475  */
476         setup_makeflags_argv();
477         read_command_options(mf_argc, mf_argv);
478         read_command_options(argc, argv);
479         if (debug_level > 0) {
480                 cp = getenv(makeflags->string_mb);
481                 (void) printf(catgets(catd, 1, 167, "MAKEFLAGS value: %s\n"), cp
482         }

484         setup_interrupt(handle_interrupt);

486         read_files_and_state(argc, argv);

488 #ifdef TEAMWARE_MAKE_CMN
489         /*
490          * Find the dmake_output_mode: TXT1, TXT2 or HTML1.
491          */
492         MBSTOWCS(wcs_buffer, NOCATGETS("DMAKE_OUTPUT_MODE"));
493         dmake_name2 = GETNAME(wcs_buffer, FIND_LENGTH);
494         prop2 = get_prop(dmake_name2->prop, macro_prop);
495         if (prop2 == NULL) {
496                 /* DMAKE_OUTPUT_MODE not defined, default to TXT1 mode */
497                 output_mode = txt1_mode;
498         } else {
499                 dmake_value2 = prop2->body.macro.value;
500                 if ((dmake_value2 == NULL) ||
501                     (IS_EQUAL(dmake_value2->string_mb, NOCATGETS("TXT1")))) {
502                         output_mode = txt1_mode;
503                 } else if (IS_EQUAL(dmake_value2->string_mb, NOCATGETS("TXT2")))
504                         output_mode = txt2_mode;
505                 } else if (IS_EQUAL(dmake_value2->string_mb, NOCATGETS("HTML1"))
506                         output_mode = html1_mode;
```

```
507                 } else {
508                         warning(catgets(catd, 1, 352, "Unsupported value ‘%s’ fo
509                               dmake_value2->string_mb);
510                 }
511         }
512         /*
513          * Find the dmake_mode: distributed, parallel, or serial.
514          */
515     if ((!pmake_cap_r_specified) &&
516         (!pmake_machinesfile_specified)) {
517         MBSTOWCS(wcs_buffer, NOCATGETS("DMAKE_MODE"));
518         dmake_name2 = GETNAME(wcs_buffer, FIND_LENGTH);
519         prop2 = get_prop(dmake_name2->prop, macro_prop);
520         if (prop2 == NULL) {
521                 /* DMAKE_MODE not defined, default to distributed mode */
522                 dmake_mode_type = distributed_mode;
523                 no_parallel = false;
524         } else {
525                 dmake_value2 = prop2->body.macro.value;
526                 if ((dmake_value2 == NULL) ||
527                     (IS_EQUAL(dmake_value2->string_mb, NOCATGETS("distributed"))
528                         dmake_mode_type = distributed_mode;
529                         no_parallel = false;
530                 } else if (IS_EQUAL(dmake_value2->string_mb, NOCATGETS("parallel
531                         dmake_mode_type = parallel_mode;
532                         no_parallel = false;
533 #ifdef SGE_SUPPORT
534                         grid = false;
535                 } else if (IS_EQUAL(dmake_value2->string_mb, NOCATGETS("grid")))
536                         dmake_mode_type = parallel_mode;
537                         no_parallel = false;
538                         grid = true;
539 #endif
540                 } else if (IS_EQUAL(dmake_value2->string_mb, NOCATGETS("serial")
541                         dmake_mode_type = serial_mode;
542                         no_parallel = true;
543                 } else {
544                         fatal(catgets(catd, 1, 307, "Unknown dmake mode argument
545                 }
546         }

548         if ((!list_all_targets) &&
549             (report_dependencies_level == 0)) {
550                 /*
551                  * Check to see if either DMAKE_RCFILE or DMAKE_MODE is defined.
552                  * They could be defined in the env, in the makefile, or on the
553                  * command line.
554                  * If neither is defined, and $(HOME)/.dmakerc does not exists,
555                  * then print a message, and default to parallel mode.
556                  */
557 #ifdef DISTRIBUTED
558                 MBSTOWCS(wcs_buffer, NOCATGETS("DMAKE_RCFILE"));
559                 dmake_name = GETNAME(wcs_buffer, FIND_LENGTH);
560                 MBSTOWCS(wcs_buffer, NOCATGETS("DMAKE_MODE"));
561                 dmake_name2 = GETNAME(wcs_buffer, FIND_LENGTH);
562                 if ((((prop = get_prop(dmake_name->prop, macro_prop)) == NULL) |
563                      ((dmake_value = prop->body.macro.value) == NULL)) &&
564                     (((prop2 = get_prop(dmake_name2->prop, macro_prop)) == NULL)
565                      ((dmake_value2 = prop2->body.macro.value) == NULL))) {
566                         Boolean empty_dmakerc = true;
567                         char *homedir = getenv(NOCATGETS("HOME"));
568                         if ((homedir != NULL) && (strlen(homedir) < (sizeof(def_
569                                 sprintf(def_dmakerc_path, NOCATGETS("%s/.dmakerc
570                                 if ((((statval = stat(def_dmakerc_path, &statbuf
571                                         ((statval == 0) && (statbuf.st_size == 0
572                                 } else {
```



```
573                                         Avo_dmakerc     *rcfile = new Avo_dmaker
574                                         Avo_err         *err = rcfile->read(def_
575                                         if (err) {
576                                                 fatal(err->str);
577                                         }
578                                         empty_dmakerc = rcfile->was_empty();
579                                         delete rcfile;
580                                 }
581                         }
582                         if (empty_dmakerc) {
583                                 if (getenv(NOCATGETS("DMAKE_DEF_PRINTED")) == NU
584                                         putenv(NOCATGETS("DMAKE_DEF_PRINTED=TRUE
585                                         (void) fprintf(stdout, catgets(catd, 1,
586                                         (void) fprintf(stdout, catgets(catd, 1,
587                                 }
588                                 dmake_mode_type = parallel_mode;
589                                 no_parallel = false;
590                         }
591                 }
592 #else
593                 if(dmake_mode_type == distributed_mode) {
594                         (void) fprintf(stdout, NOCATGETS("dmake: Distributed mod
595                         (void) fprintf(stdout, NOCATGETS("       Defaulting to p
596                         dmake_mode_type = parallel_mode;
597                         no_parallel = false;
598                 }
599 #endif  /* DISTRIBUTED */
600         }
601     }
602 #endif

604 #ifdef TEAMWARE_MAKE_CMN
605         parallel_flag = true;
606         /* XXX - This is a major hack for DMake/Licensing. */
607         if (getenv(NOCATGETS("DMAKE_CHILD")) == NULL) {
608                 if (!avo_cli_search_license(argv[0], dmake_exit_callback, TRUE,
609                         /*
610                          * If the user can not get a TeamWare license,
611                          * default to serial mode.
612                          */
613                         dmake_mode_type = serial_mode;
614                         no_parallel = true;
615                 } else {
616                         putenv(NOCATGETS("DMAKE_CHILD=TRUE"));
617                 }
618                 start_time = time(NULL);
619                 /*
620                  * XXX - Hack to disable SIGALRM’s from licensing library’s
621                  *       setitimer().
622                  */
623                 value.it_interval.tv_sec = 0;
624                 value.it_interval.tv_usec = 0;
625                 value.it_value.tv_sec = 0;
626                 value.it_value.tv_usec = 0;
627                 (void) setitimer(ITIMER_REAL, &value, NULL);
628         }

630 //
631 // If dmake is running with -t option, set dmake_mode_type to serial.
632 // This is done because doname() calls touch_command() that runs serially.
633 // If we do not do that, maketool will have problems.
634 //
635         if(touch) {
636                 dmake_mode_type = serial_mode;
637                 no_parallel = true;
638         }
```

```
#else
        parallel_flag = false;
#endif

#if defined (TEAMWARE_MAKE_CMN) && defined(REDIRECT_ERR)
        /*
         * Check whether stdout and stderr are physically same.
         * This is in order to decide whether we need to redirect
         * stderr separately from stdout.
         * This check is performed only if __DMAKE_SEPARATE_STDERR
         * is not set. This variable may be used in order to preserve
         * the 'old' behaviour.
         */
        out_err_same = true;
        char * dmake_sep_var = getenv(NOCATGETS("__DMAKE_SEPARATE_STDERR"));
        if (dmake_sep_var == NULL || (0 != strcasecmp(dmake_sep_var, NOCATGETS("
                struct stat stdout_stat;
                struct stat stderr_stat;
                if( (fstat(1, &stdout_stat) == 0)
                 && (fstat(2, &stderr_stat) == 0) )
                {
                        if( (stdout_stat.st_dev != stderr_stat.st_dev)
                         || (stdout_stat.st_ino != stderr_stat.st_ino) )
                        {
                                out_err_same = false;
                        }
                }
        }
#endif

#ifdef DISTRIBUTED
        /*
         * At this point, DMake should startup an rxm with any and all
         * DMake command line options. Rxm will, among other things,
         * read the rc file.
         */
        if ((!list_all_targets) &&
            (report_dependencies_level == 0) &&
            (dmake_mode_type == distributed_mode)) {
                startup_rxm();
        }
#endif


/*
 *      Enable interrupt handler for alarms
 */
        (void) bsd_signal(SIGALRM, (SIG_PF)doalarm);

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

                report_dependency("");
                report_file = get_report_file();
                if ((report_file != NULL) && (report_file != (FILE*)-1)) {
                        (void) fprintf(report_file, "\n");
                }
        }

/*
 *      Make sure SUNPRO_DEPENDENCIES is exported (or not) properly
 *      and NSE_DEP.
 */
        if (keep_state) {
```



```
                maybe_append_prop(sunpro_dependencies, macro_prop)->
                  body.macro.exported = true;
#ifdef NSE
                (void) setenv(NOCATGETS("NSE_DEP"), get_current_path());
#endif
        } else {
                maybe_append_prop(sunpro_dependencies, macro_prop)->
                  body.macro.exported = false;
        }

        working_on_targets = true;
        if (trace_status) {
                dump_make_state();
                fclose(stdout);
                fclose(stderr);
                exit_status = 0;
                exit(0);
        }
        if (list_all_targets) {
                dump_target_list();
                fclose(stdout);
                fclose(stderr);
                exit_status = 0;
                exit(0);
        }
        trace_reader = false;

        /*
         * Set temp_file_directory to the directory the .make.state
         * file is written to.
         */
        if ((slash_ptr = strrchr(make_state->string_mb, (int) slash_char)) == NU
                temp_file_directory = strdup(get_current_path());
        } else {
                *slash_ptr = (int) nul_char;
                (void) strcpy(make_state_dir, make_state->string_mb);
                *slash_ptr = (int) slash_char;
                   /* when there is only one slash and it's the first
                   ** character, make_state_dir should point to '/'.
                   */
                if(make_state_dir[0] == '\0') {
                   make_state_dir[0] = '/';
                   make_state_dir[1] = '\0';
                }
                if (make_state_dir[0] == (int) slash_char) {
                        temp_file_directory = strdup(make_state_dir);
                } else {
                        char    tmp_current_path2[MAXPATHLEN];

                        (void) sprintf(tmp_current_path2,
                                       "%s/%s",
                                       get_current_path(),
                                       make_state_dir);
                        temp_file_directory = strdup(tmp_current_path2);
                }
        }

#ifdef DISTRIBUTED
        building_serial = false;
#endif

        report_dir_enter_leave(true);

        make_targets(argc, argv, parallel_flag);

        report_dir_enter_leave(false);
```

```
 772 #ifdef NSE
 773         exit(nse_exit_status());
 774 #else
 775         if (build_failed_ever_seen) {
 776                 if (posix) {
 777                         exit_status = 1;
 778                 }
 779                 exit(1);
 780         }
 781         exit_status = 0;
 782         exit(0);
 783 #endif
 784         /* NOTREACHED */
 785 }
```

unchanged_portion_omitted_

```
1202 /*
1203  *      read_command_options(argc, argv)
1204  *
1205  *      Scan the cmd line options and process the ones that start with "-"
1206  *
1207  *      Return value:
1208  *                              -M argument, if any
1209  *
1210  *      Parameters:
1211  *              argc            You know what this is
1212  *              argv            You know what this is
1213  *
1214  *      Global variables used:
1215  */
1216 static void
1217 read_command_options(register int argc, register char **argv)
1218 {
1219         register int            ch;
1220         int                     current_optind = 1;
1221         int                     last_optind_with_double_hyphen = 0;
1222         int                     last_optind;
1223         int                     last_current_optind;
1224         register int            i;
1225         register int            j;
1226         register int            k;
1227         register int            makefile_next = 0; /*
1228                                                     * flag to note options:
1229                                                     * -c, f, g, j, m, o
1230                                                     */
1231         const char              *tptr;
1232         const char              *CMD_OPTS;

1234         extern char             *optarg;
1235         extern int              optind, opterr, optopt;

1237 #define SUNPRO_CMD_OPTS "-~Bbc:Ddef:g:ij:K:kM:m:NnO:o:PpqRrSsTtuVvwx:"

1239 #ifdef TEAMWARE_MAKE_CMN
1240 #       define SVR4_CMD_OPTS   "-c:ef:g:ij:km:nO:o:pqrsTtVv"
1241 #else
1242 #       define SVR4_CMD_OPTS   "-ef:iknpqrstV"
1243 #endif

1245         /*
1246          * Added V in SVR4_CMD_OPTS also, which is going to be a hidden
1247          * option, just to make sure that the getopt doesn't fail when some
1248          * users leave their USE_SVR4_MAKE set and try to use the makefiles
1249          * that are designed to issue commands like $(MAKE) -V. Anyway it
```



```
1250          * sets the same flag but ensures that getopt doesn't fail.
1251          */

1253         opterr = 0;
1254         optind = 1;
1255         while (1) {
1256                 last_optind=optind;                     /* Save optind and curre
1257                 last_current_optind=current_optind;     /* in case we have to re
1258                 if (svr4) {
1259                         CMD_OPTS=SVR4_CMD_OPTS;
1260                         ch = getopt(argc, argv, SVR4_CMD_OPTS);
1261                 } else {
1262                         CMD_OPTS=SUNPRO_CMD_OPTS;
1263                         ch = getopt(argc, argv, SUNPRO_CMD_OPTS);
1264                 }
1265                 if (ch == EOF) {
1266                         if(optind < argc) {
1267                                 /*
1268                                  * Fixing bug 4102537:
1269                                  *    Strange behaviour of command make using --
1270                                  * Not all argv have been processed
1271                                  * Skip non-flag argv and continue processing.
1272                                  */
1273                                 optind++;
1274                                 current_optind++;
1275                                 continue;
1276                         } else {
1277                                 break;
1278                         }

1280                 }
1281                 if (ch == '?') {
1282                         if (optopt == '-') {
1283                                 /* Bug 5060758: getopt() changed behavior (s10_6
1284                                  * and now we have to deal with cases when optio
1285                                  * with double hyphen appear here, from -$(MAKEF
1286                                  */
1287                                 i = current_optind;
1288                                 if (argv[i][0] == '-') {
1289                                   if (argv[i][1] == '-') {
1290                                     if (argv[i][2] != '\0') {
1291                                       /* Check if this option is allowed */
1292                                       tptr = strchr(CMD_OPTS, argv[i][2]);
1293                                       if (tptr) {
1294                                         if (last_optind_with_double_hyphen != cu
1295                                           /* This is first time we are trying to
1296                                            * problem with this option. If we com
1297                                            * time, we will go to fatal error.
1298                                            */
1299                                           last_optind_with_double_hyphen = curre
1300
1301                                           /* Eliminate first hyphen character */
1302                                           for (j=0; argv[i][j] != '\0'; j++) {
1303                                             argv[i][j] = argv[i][j+1];
1304                                           }
1305
1306                                           /* Repeat the processing of this argum
1307                                           optind=last_optind;
1308                                           current_optind=last_current_optind;
1309                                           continue;
1310                                         }
1311                                       }
1312                                     }
1313                                   }
1314                                 }
1315                         }
```

```
1316                }

1318                if (ch == '?') {
1319                        if (svr4) {
1320 #ifdef TEAMWARE_MAKE_CMN
1321                                fprintf(stderr,
1322                                        catgets(catd, 1, 267, "Usage : dmake [ -
1323                                fprintf(stderr,
1324                                        catgets(catd, 1, 268, "              [ -
1325                                fprintf(stderr,
1326                                        catgets(catd, 1, 269, "              [ -
1327 #else
1328                                fprintf(stderr,
1329                                        catgets(catd, 1, 270, "Usage : make [ -f
1330                                fprintf(stderr,
1331                                        catgets(catd, 1, 271, "             [ -s
1332 #endif
1333                                tptr = strchr(SVR4_CMD_OPTS, optopt);
1334                        } else {
1335 #ifdef TEAMWARE_MAKE_CMN
1336                                fprintf(stderr,
1337                                        catgets(catd, 1, 272, "Usage : dmake [ -
1338                                fprintf(stderr,
1339                                        catgets(catd, 1, 273, "              [ -
1340                                fprintf(stderr,
1341                                        catgets(catd, 1, 274, "              [ -
1342                                fprintf(stderr,
1343                                        catgets(catd, 1, 275, "              [ -
1344 #else
1345                                fprintf(stderr,
1346                                        catgets(catd, 1, 276, "Usage : make [ -f
1347                                fprintf(stderr,
1348                                        catgets(catd, 1, 277, "             [ -e
1349                                fprintf(stderr,
1350                                        catgets(catd, 1, 278, "             [ -u
1351 #endif
1352                                tptr = strchr(SUNPRO_CMD_OPTS, optopt);
1353                        }
1354                        if (!tptr) {
1355                                fatal(catgets(catd, 1, 279, "Unknown option '-%c
1356                        } else {
1357                                fatal(catgets(catd, 1, 280, "Missing argument af
1358                        }
1359                }

1378 #if defined(linux)
1379                if (ch == 1) {
1380                        if(optind < argc) {
1381                                //optind++;
1382                                //current_optind++;
1383                        makefile_next = 0;
1384                        current_optind = optind;
1385                                continue;
1386                        } else {
1387                                break;
1388                        }
1389                }
1390 #endif


1363                makefile_next |= parse_command_option(ch);
1364                /*
1365                 * If we're done processing all of the options of
1366                 * ONE argument string...
1367                 */
1368                if (current_optind < optind) {
```



```
1369                        i = current_optind;
1370                        k = 0;
1371                        /* If there's an argument for an option... */
1372                        if ((optind - current_optind) > 1) {
1373                                k = i + 1;
1374                        }
1375                        switch (makefile_next) {
1376                        case 0:
1377                                argv[i] = NULL;
1378                                /* This shouldn't happen */
1379                                if (k) {
1380                                        argv[k] = NULL;
1381                                }
1382                                break;
1383                        case 1: /* -f seen */
1384                                argv[i] = (char *)NOCATGETS("-f");
1385                                break;
1386                        case 2: /* -c seen */
1387                                argv[i] = (char *)NOCATGETS("-c");
1388 #ifndef TEAMWARE_MAKE_CMN
1389                                warning(catgets(catd, 1, 281, "Ignoring Distribu
1390 #endif
1391                                break;
1392                        case 4: /* -g seen */
1393                                argv[i] = (char *)NOCATGETS("-g");
1394 #ifndef TEAMWARE_MAKE_CMN
1395                                warning(catgets(catd, 1, 282, "Ignoring Distribu
1396 #endif
1397                                break;
1398                        case 8: /* -j seen */
1399                                argv[i] = (char *)NOCATGETS("-j");
1400 #ifndef TEAMWARE_MAKE_CMN
1401                                warning(catgets(catd, 1, 283, "Ignoring Distribu
1402 #endif
1403                                break;
1404                        case 16: /* -M seen */
1405                                argv[i] = (char *)NOCATGETS("-M");
1406 #ifndef TEAMWARE_MAKE_CMN
1407                                warning(catgets(catd, 1, 284, "Ignoring Parallel
1408 #endif
1409                                break;
1410                        case 32: /* -m seen */
1411                                argv[i] = (char *)NOCATGETS("-m");
1412 #ifndef TEAMWARE_MAKE_CMN
1413                                warning(catgets(catd, 1, 285, "Ignoring Distribu
1414 #endif
1415                                break;
1416 #ifndef PARALLEL
1417                        case 128: /* -O seen */
1418                                argv[i] = (char *)NOCATGETS("-O");
1419                                break;
1420 #endif
1421                        case 256: /* -K seen */
1422                                argv[i] = (char *)NOCATGETS("-K");
1423                                break;
1424                        case 512:       /* -o seen */
1425                                argv[i] = (char *)NOCATGETS("-o");
1426 #ifndef TEAMWARE_MAKE_CMN
1427                                warning(catgets(catd, 1, 311, "Ignoring Distribu
1428 #endif
1429                                break;
1430                        case 1024: /* -x seen */
1431                                argv[i] = (char *)NOCATGETS("-x");
1432 #ifndef TEAMWARE_MAKE_CMN
1433                                warning(catgets(catd, 1, 353, "Ignoring Distribu
1434 #endif
```

```
1435                                    break;
1436                            default: /* > 1 of -c, f, g, j, K, M, m, O, o, x seen */
1437                                    fatal(catgets(catd, 1, 286, "Illegal command lin
1438                            }
1439
1440                            makefile_next = 0;
1441                            current_optind = optind;
1442                    }
1443            }
1444 }
_____unchanged_portion_omitted_

1626 /*
1627  *      parse_command_option(ch)
1628  *
1629  *      Parse make command line options.
1630  *
1631  *      Return value:
1632  *                              Indicates if any -f -c or -M were seen
1633  *
1634  *      Parameters:
1635  *              ch              The character to parse
1636  *
1637  *      Static variables used:
1638  *              dmake_group_specified   Set for make -g
1639  *              dmake_max_jobs_specified        Set for make -j
1640  *              dmake_mode_specified    Set for make -m
1641  *              dmake_add_mode_specified        Set for make -x
1642  *              dmake_compat_mode_specified     Set for make -x SUN_MAKE_COMPAT_
1643  *              dmake_output_mode_specified     Set for make -x DMAKE_OUTPUT_MOD
1644  *              dmake_odir_specified    Set for make -o
1645  *              dmake_rcfile_specified  Set for make -c
1646  *              env_wins                Set for make -e
1647  *              ignore_default_mk       Set for make -r
1648  *              trace_status            Set for make -p
1649  *
1650  *      Global variables used:
1651  *              .make.state path & name set for make -K
1652  *              continue_after_error    Set for make -k
1653  *              debug_level             Set for make -d
1654  *              do_not_exec_rule        Set for make -n
1655  *              filter_stderr           Set for make -X
1656  *              ignore_errors_all       Set for make -i
1657  *              no_parallel             Set for make -R
1658  *              quest                   Set for make -q
1659  *              read_trace_level        Set for make -D
1660  *              report_dependencies     Set for make -P
1661  *              send_mtool_msgs         Set for make -K
1662  *              silent_all              Set for make -s
1663  *              touch                   Set for make -t
1664  */
1665 static int
1666 parse_command_option(register char ch)
1667 {
1668         static int              invert_next = 0;
1669         int                     invert_this = invert_next;

1671         invert_next = 0;
1672         switch (ch) {
1673         case '-':                       /* Ignore "--" */
1674                 return 0;
1675         case '~':                       /* Invert next option */
1676                 invert_next = 1;
1677                 return 0;
1678         case 'B':                       /* Obsolete */
1679                 return 0;
```



```
1680         case 'b':                       /* Obsolete */
1681                 return 0;
1682         case 'c':                       /* Read alternative dmakerc file */
1683                 if (invert_this) {
1684                         dmake_rcfile_specified = false;
1685                 } else {
1686                         dmake_rcfile_specified = true;
1687                 }
1688                 return 2;
1689         case 'D':                       /* Show lines read */
1690                 if (invert_this) {
1691                         read_trace_level--;
1692                 } else {
1693                         read_trace_level++;
1694                 }
1695                 return 0;
1696         case 'd':                       /* Debug flag */
1697                 if (invert_this) {
1698                         debug_level--;
1699                 } else {
1730 #if defined( HP_UX) || defined(linux)
1731                         if (debug_level < 2)  /* Fixes a bug on HP-UX */
1732 #endif
1700                         debug_level++;
1701                 }
1702                 return 0;
1703 #ifdef NSE
1704         case 'E':
1705                 if (invert_this) {
1706                         nse = false;
1707                 } else {
1708                         nse = true;
1709                 }
1710                 nse_init_source_suffixes();
1711                 return 0;
1712 #endif
1713         case 'e':                       /* Environment override flag */
1714                 if (invert_this) {
1715                         env_wins = false;
1716                 } else {
1717                         env_wins = true;
1718                 }
1719                 return 0;
1720         case 'f':                       /* Read alternative makefile(s) */
1721                 return 1;
1722         case 'g':                       /* Use alternative DMake group */
1723                 if (invert_this) {
1724                         dmake_group_specified = false;
1725                 } else {
1726                         dmake_group_specified = true;
1727                 }
1728                 return 4;
1729         case 'i':                       /* Ignore errors */
1730                 if (invert_this) {
1731                         ignore_errors_all = false;
1732                 } else {
1733                         ignore_errors_all = true;
1734                 }
1735                 return 0;
1736         case 'j':                       /* Use alternative DMake max jobs */
1737                 if (invert_this) {
1738                         dmake_max_jobs_specified = false;
1739                 } else {
1740                         dmake_max_jobs_specified = true;
1741                 }
1742                 return 8;
```

```
1743         case 'K':                          /* Read alternative .make.state */
1744                 return 256;
1745         case 'k':                          /* Keep making even after errors */
1746                 if (invert_this) {
1747                         continue_after_error = false;
1748                 } else {
1749                         continue_after_error = true;
1750                         continue_after_error_ever_seen = true;
1751                 }
1752                 return 0;
1753         case 'M':                          /* Read alternative make.machines file
1754                 if (invert_this) {
1755                         pmake_machinesfile_specified = false;
1756                 } else {
1757                         pmake_machinesfile_specified = true;
1758                         dmake_mode_type = parallel_mode;
1759                         no_parallel = false;
1760                 }
1761                 return 16;
1762         case 'm':                          /* Use alternative DMake build mode */
1763                 if (invert_this) {
1764                         dmake_mode_specified = false;
1765                 } else {
1766                         dmake_mode_specified = true;
1767                 }
1768                 return 32;
1769         case 'x':                          /* Use alternative DMake mode */
1770                 if (invert_this) {
1771                         dmake_add_mode_specified = false;
1772                 } else {
1773                         dmake_add_mode_specified = true;
1774                 }
1775                 return 1024;
1776         case 'N':                          /* Reverse -n */
1777                 if (invert_this) {
1778                         do_not_exec_rule = true;
1779                 } else {
1780                         do_not_exec_rule = false;
1781                 }
1782                 return 0;
1783         case 'n':                          /* Print, not exec commands */
1784                 if (invert_this) {
1785                         do_not_exec_rule = false;
1786                 } else {
1787                         do_not_exec_rule = true;
1788                 }
1789                 return 0;
1790 #ifndef PARALLEL
1791         case 'O':                          /* Send job start & result msgs */
1792                 if (invert_this) {
1793                         send_mtool_msgs = false;
1794                 } else {
1795 #ifdef DISTRIBUTED
1796                         send_mtool_msgs = true;
1797 #endif
1798                 }
1799                 return 128;
1800 #endif
1801         case 'o':                          /* Use alternative dmake output dir */
1802                 if (invert_this) {
1803                         dmake_odir_specified = false;
1804                 } else {
1805                         dmake_odir_specified = true;
1806                 }
1807                 return 512;
1808         case 'P':                          /* Print for selected targets */
```



```
1809                 if (invert_this) {
1810                         report_dependencies_level--;
1811                 } else {
1812                         report_dependencies_level++;
1813                 }
1814                 return 0;
1815         case 'p':                          /* Print description */
1816                 if (invert_this) {
1817                         trace_status = false;
1818                         do_not_exec_rule = false;
1819                 } else {
1820                         trace_status = true;
1821                         do_not_exec_rule = true;
1822                 }
1823                 return 0;
1824         case 'q':                          /* Question flag */
1825                 if (invert_this) {
1826                         quest = false;
1827                 } else {
1828                         quest = true;
1829                 }
1830                 return 0;
1831         case 'R':                          /* Don't run in parallel */
1832 #ifdef TEAMWARE_MAKE_CMN
1833                 if (invert_this) {
1834                         pmake_cap_r_specified = false;
1835                         no_parallel = false;
1836                 } else {
1837                         pmake_cap_r_specified = true;
1838                         dmake_mode_type = serial_mode;
1839                         no_parallel = true;
1840                 }
1841 #else
1842                 warning(catgets(catd, 1, 182, "Ignoring ParallelMake -R option")
1843 #endif
1844                 return 0;
1845         case 'r':                          /* Turn off internal rules */
1846                 if (invert_this) {
1847                         ignore_default_mk = false;
1848                 } else {
1849                         ignore_default_mk = true;
1850                 }
1851                 return 0;
1852         case 'S':                          /* Reverse -k */
1853                 if (invert_this) {
1854                         continue_after_error = true;
1855                 } else {
1856                         continue_after_error = false;
1857                         stop_after_error_ever_seen = true;
1858                 }
1859                 return 0;
1860         case 's':                          /* Silent flag */
1861                 if (invert_this) {
1862                         silent_all = false;
1863                 } else {
1864                         silent_all = true;
1865                 }
1866                 return 0;
1867         case 'T':                          /* Print target list */
1868                 if (invert_this) {
1869                         list_all_targets = false;
1870                         do_not_exec_rule = false;
1871                 } else {
1872                         list_all_targets = true;
1873                         do_not_exec_rule = true;
1874                 }
```

```
1875                 return 0;
1876         case 't':                          /* Touch flag */
1877                 if (invert_this) {
1878                         touch = false;
1879                 } else {
1880                         touch = true;
1881                 }
1882                 return 0;
1883         case 'u':                          /* Unconditional flag */
1884                 if (invert_this) {
1885                         build_unconditional = false;
1886                 } else {
1887                         build_unconditional = true;
1888                 }
1889                 return 0;
1890         case 'V':                         /* SVR4 mode */
1891                 svr4 = true;
1892                 return 0;
1893         case 'v':                         /* Version flag */
1894                 if (invert_this) {
1895                 } else {
1896 #ifdef TEAMWARE_MAKE_CMN
1897                         fprintf(stdout, NOCATGETS("dmake: %s\n"), verstring);
1898                         exit_status = 0;
1899                         exit(0);
1900 #else
1901                         warning(catgets(catd, 1, 324, "Ignoring DistributedMake
1902 #endif
1903                 }
1904                 return 0;
1905         case 'w':                          /* Unconditional flag */
1906                 if (invert_this) {
1907                         report_cwd = false;
1908                 } else {
1909                         report_cwd = true;
1910                 }
1911                 return 0;
1912 #if 0
1913         case 'X':                         /* Filter stdout */
1914                 if (invert_this) {
1915                         filter_stderr = false;
1916                 } else {
1917                         filter_stderr = true;
1918                 }
1919                 return 0;
1920 #endif
1921         default:
1922                 break;
1923         }
1924         return 0;
1925 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   26368 Wed May 20 11:27:35 2015
new/usr/src/cmd/make/bin/misc.cc
make: unifdef for other OSes (undefined)
**********************************************************
```

```
   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 2005 Sun Microsystems, Inc. All rights reserved.
  23  * Use is subject to license terms.
  24  */

  26 /*
  27  *      misc.cc
  28  *
  29  *      This file contains various unclassified routines. Some main groups:
  30  *              getname
  31  *              Memory allocation
  32  *              String handling
  33  *              Property handling
  34  *              Error message handling
  35  *              Make internal state dumping
  36  *              main routine support
  37  */

  39 /*
  40  * Included files
  41  */
  42 #include <errno.h>
  43 #include <mk/defs.h>
  44 #include <mksh/macro.h>         /* SETVAR() */
  45 #include <mksh/misc.h>          /* enable_interrupt() */
  46 #include <stdarg.h>             /* va_list, va_start(), va_end() */
  47 #include <vroot/report.h>       /* SUNPRO_DEPENDENCIES */

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

  50 #ifdef TEAMWARE_MAKE_CMN
  51 #define MAXJOBS_ADJUST_RFE4694000

  53 #ifdef MAXJOBS_ADJUST_RFE4694000
  54 extern void job_adjust_fini();
  55 #endif /* MAXJOBS_ADJUST_RFE4694000 */
  56 #endif /* TEAMWARE_MAKE_CMN */

  61 #if defined(linux)
```



```
  62 #include <time.h>              /* localtime() */
  63 #endif

  59 /*
  60  * Defined macros
  61  */

  63 /*
  64  * typedefs & structs
  65  */

  67 /*
  68  * Static variables
  69  */

  71 /*
  72  * File table of contents
  73  */
  74 static  void            print_rule(register Name target);
  75 static  void            print_target_n_deps(register Name target);

  77 /*************************************
  78  *
  79  *      getname
  80  */

  82 /*************************************
  83  *
  84  *      Memory allocation
  85  */

  87 /*
  88  *      free_chain()
  89  *
  90  *      frees a chain of Name_vector's
  91  *
  92  *      Parameters:
  93  *              ptr             Pointer to the first element in the chain
  94  *                              to be freed.
  95  *
  96  *      Global variables used:
  97  */
  98 void
  99 free_chain(Name_vector ptr)
 100 {
 101         if (ptr != NULL) {
 102                 if (ptr->next != NULL) {
 103                         free_chain(ptr->next);
 104                 }
 105                 free((char *) ptr);
 106         }
 107 }
_____unchanged_portion_omitted_

 568 /*************************************
 569  *
 570  *      main() support
 571  */

 573 /*
 574  *      load_cached_names()
 575  *
 576  *      Load the vector of cached names
 577  *
 578  *      Parameters:
 579  *
```

```
 *      Global variables used:
 *              Many many pointers to Name blocks.
 */
void
load_cached_names(void)
{
        char            *cp;
        Name            dollar;

        /* Load the cached_names struct */
        MBSTOWCS(wcs_buffer, NOCATGETS(".BUILT_LAST_MAKE_RUN"));
        built_last_make_run = GETNAME(wcs_buffer, FIND_LENGTH);
        MBSTOWCS(wcs_buffer, NOCATGETS("@"));
        c_at = GETNAME(wcs_buffer, FIND_LENGTH);
        MBSTOWCS(wcs_buffer, NOCATGETS(" *conditionals* "));
        conditionals = GETNAME(wcs_buffer, FIND_LENGTH);
        /*
         * A version of make was released with NSE 1.0 that used
         * VERSION-1.1 but this version is identical to VERSION-1.0.
         * The version mismatch code makes a special case for this
         * situation.  If the version number is changed from 1.0
         * it should go to 1.2.
         */
        MBSTOWCS(wcs_buffer, NOCATGETS("VERSION-1.0"));
        current_make_version = GETNAME(wcs_buffer, FIND_LENGTH);
        MBSTOWCS(wcs_buffer, NOCATGETS(".SVR4"));
        svr4_name = GETNAME(wcs_buffer, FIND_LENGTH);
        MBSTOWCS(wcs_buffer, NOCATGETS(".POSIX"));
        posix_name = GETNAME(wcs_buffer, FIND_LENGTH);
        MBSTOWCS(wcs_buffer, NOCATGETS(".DEFAULT"));
        default_rule_name = GETNAME(wcs_buffer, FIND_LENGTH);
#ifdef NSE
        MBSTOWCS(wcs_buffer, NOCATGETS(".DERIVED_SRC"));
        derived_src= GETNAME(wcs_buffer, FIND_LENGTH);
#endif
        MBSTOWCS(wcs_buffer, NOCATGETS("$"));
        dollar = GETNAME(wcs_buffer, FIND_LENGTH);
        MBSTOWCS(wcs_buffer, NOCATGETS(".DONE"));
        done = GETNAME(wcs_buffer, FIND_LENGTH);
        MBSTOWCS(wcs_buffer, NOCATGETS("."));
        dot = GETNAME(wcs_buffer, FIND_LENGTH);
        MBSTOWCS(wcs_buffer, NOCATGETS(".KEEP_STATE"));
        dot_keep_state = GETNAME(wcs_buffer, FIND_LENGTH);
        MBSTOWCS(wcs_buffer, NOCATGETS(".KEEP_STATE_FILE"));
        dot_keep_state_file = GETNAME(wcs_buffer, FIND_LENGTH);
        MBSTOWCS(wcs_buffer, NOCATGETS(""));
        empty_name = GETNAME(wcs_buffer, FIND_LENGTH);
        MBSTOWCS(wcs_buffer, NOCATGETS(" FORCE"));
        force = GETNAME(wcs_buffer, FIND_LENGTH);
        MBSTOWCS(wcs_buffer, NOCATGETS("HOST_ARCH"));
        host_arch = GETNAME(wcs_buffer, FIND_LENGTH);
        MBSTOWCS(wcs_buffer, NOCATGETS("HOST_MACH"));
        host_mach = GETNAME(wcs_buffer, FIND_LENGTH);
        MBSTOWCS(wcs_buffer, NOCATGETS(".IGNORE"));
        ignore_name = GETNAME(wcs_buffer, FIND_LENGTH);
        MBSTOWCS(wcs_buffer, NOCATGETS(".INIT"));
        init = GETNAME(wcs_buffer, FIND_LENGTH);
        MBSTOWCS(wcs_buffer, NOCATGETS(".LOCAL"));
        localhost_name = GETNAME(wcs_buffer, FIND_LENGTH);
        MBSTOWCS(wcs_buffer, NOCATGETS(".make.state"));
        make_state = GETNAME(wcs_buffer, FIND_LENGTH);
        MBSTOWCS(wcs_buffer, NOCATGETS("MAKEFLAGS"));
        makeflags = GETNAME(wcs_buffer, FIND_LENGTH);
        MBSTOWCS(wcs_buffer, NOCATGETS(".MAKE_VERSION"));
        make_version = GETNAME(wcs_buffer, FIND_LENGTH);
        MBSTOWCS(wcs_buffer, NOCATGETS(".NO_PARALLEL"));
```



```
        no_parallel_name = GETNAME(wcs_buffer, FIND_LENGTH);
        MBSTOWCS(wcs_buffer, NOCATGETS(".NOT_AUTO"));
        not_auto = GETNAME(wcs_buffer, FIND_LENGTH);
        MBSTOWCS(wcs_buffer, NOCATGETS(".PARALLEL"));
        parallel_name = GETNAME(wcs_buffer, FIND_LENGTH);
        MBSTOWCS(wcs_buffer, NOCATGETS("PATH"));
        path_name = GETNAME(wcs_buffer, FIND_LENGTH);
        MBSTOWCS(wcs_buffer, NOCATGETS("+"));
        plus = GETNAME(wcs_buffer, FIND_LENGTH);
        MBSTOWCS(wcs_buffer, NOCATGETS(".PRECIOUS"));
        precious = GETNAME(wcs_buffer, FIND_LENGTH);
        MBSTOWCS(wcs_buffer, NOCATGETS("?"));
        query = GETNAME(wcs_buffer, FIND_LENGTH);
        MBSTOWCS(wcs_buffer, NOCATGETS("^"));
        hat = GETNAME(wcs_buffer, FIND_LENGTH);
        MBSTOWCS(wcs_buffer, NOCATGETS(".RECURSIVE"));
        recursive_name = GETNAME(wcs_buffer, FIND_LENGTH);
        MBSTOWCS(wcs_buffer, NOCATGETS(".SCCS_GET"));
        sccs_get_name = GETNAME(wcs_buffer, FIND_LENGTH);
        MBSTOWCS(wcs_buffer, NOCATGETS(".SCCS_GET_POSIX"));
        sccs_get_posix_name = GETNAME(wcs_buffer, FIND_LENGTH);
        MBSTOWCS(wcs_buffer, NOCATGETS(".GET"));
        get_name = GETNAME(wcs_buffer, FIND_LENGTH);
        MBSTOWCS(wcs_buffer, NOCATGETS(".GET_POSIX"));
        get_posix_name = GETNAME(wcs_buffer, FIND_LENGTH);
        MBSTOWCS(wcs_buffer, NOCATGETS("SHELL"));
        shell_name = GETNAME(wcs_buffer, FIND_LENGTH);
        MBSTOWCS(wcs_buffer, NOCATGETS(".SILENT"));
        silent_name = GETNAME(wcs_buffer, FIND_LENGTH);
        MBSTOWCS(wcs_buffer, NOCATGETS(".SUFFIXES"));
        suffixes_name = GETNAME(wcs_buffer, FIND_LENGTH);
        MBSTOWCS(wcs_buffer, SUNPRO_DEPENDENCIES);
        sunpro_dependencies = GETNAME(wcs_buffer, FIND_LENGTH);
        MBSTOWCS(wcs_buffer, NOCATGETS("TARGET_ARCH"));
        target_arch = GETNAME(wcs_buffer, FIND_LENGTH);
        MBSTOWCS(wcs_buffer, NOCATGETS("TARGET_MACH"));
        target_mach = GETNAME(wcs_buffer, FIND_LENGTH);
        MBSTOWCS(wcs_buffer, NOCATGETS("VIRTUAL_ROOT"));
        virtual_root = GETNAME(wcs_buffer, FIND_LENGTH);
        MBSTOWCS(wcs_buffer, NOCATGETS("VPATH"));
        vpath_name = GETNAME(wcs_buffer, FIND_LENGTH);
        MBSTOWCS(wcs_buffer, NOCATGETS(".WAIT"));
        wait_name = GETNAME(wcs_buffer, FIND_LENGTH);

        wait_name->state = build_ok;

        /* Mark special targets so that the reader treats them properly */
        svr4_name->special_reader = svr4_special;
        posix_name->special_reader = posix_special;
        built_last_make_run->special_reader = built_last_make_run_special;
        default_rule_name->special_reader = default_special;
#ifdef NSE
        derived_src->special_reader= derived_src_special;
#endif
        dot_keep_state->special_reader = keep_state_special;
        dot_keep_state_file->special_reader = keep_state_file_special;
        ignore_name->special_reader = ignore_special;
        make_version->special_reader = make_version_special;
        no_parallel_name->special_reader = no_parallel_special;
        parallel_name->special_reader = parallel_special;
        localhost_name->special_reader = localhost_special;
        precious->special_reader = precious_special;
        sccs_get_name->special_reader = sccs_get_special;
        sccs_get_posix_name->special_reader = sccs_get_posix_special;
        get_name->special_reader = get_special;
        get_posix_name->special_reader = get_posix_special;
```

```
712         silent_name->special_reader = silent_special;
713         suffixes_name->special_reader = suffixes_special;

715         /* The value of $$ is $ */
716         (void) SETVAR(dollar, dollar, false);
717         dollar->dollar = false;

719         /* Set the value of $(SHELL) */
726         #ifdef HP_UX
727         MBSTOWCS(wcs_buffer, NOCATGETS("/bin/posix/sh"));
728         #else
720         if (posix) {
721           MBSTOWCS(wcs_buffer, NOCATGETS("/usr/xpg4/bin/sh"));
722         } else {
723           MBSTOWCS(wcs_buffer, NOCATGETS("/bin/sh"));
724         }
734         #endif
725         (void) SETVAR(shell_name, GETNAME(wcs_buffer, FIND_LENGTH), false);

727         /*
728          * Use " FORCE" to simulate a FRC dependency for :: type
729          * targets with no dependencies.
730          */
731         (void) append_prop(force, line_prop);
732         force->stat.time = file_max_time;

734         /* Make sure VPATH is defined before current dir is read */
735         if ((cp = getenv(vpath_name->string_mb)) != NULL) {
736                 MBSTOWCS(wcs_buffer, cp);
737                 (void) SETVAR(vpath_name,
738                               GETNAME(wcs_buffer, FIND_LENGTH),
739                               false);
740         }

742         /* Check if there is NO PATH variable. If not we construct one. */
743         if (getenv(path_name->string_mb) == NULL) {
744                 vroot_path = NULL;
745                 add_dir_to_path(NOCATGETS("."), &vroot_path, -1);
746                 add_dir_to_path(NOCATGETS("/bin"), &vroot_path, -1);
747                 add_dir_to_path(NOCATGETS("/usr/bin"), &vroot_path, -1);
748         }
749 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   61268 Wed May 20 11:27:35 2015
new/usr/src/cmd/make/bin/parallel.cc
make: unifdef for other OSes (undefined)
**********************************************************
______unchanged_portion_omitted_
 434 #endif

 436 #ifdef TEAMWARE_MAKE_CMN
 437 #define MAXJOBS_ADJUST_RFE4694000

 439 #ifdef MAXJOBS_ADJUST_RFE4694000

 441 #include <unistd.h>     /* sysconf(_SC_NPROCESSORS_ONLN) */
 442 #include <sys/ipc.h>            /* ftok() */
 443 #include <sys/shm.h>            /* shmget(), shmat(), shmdt(), shmctl() */
 444 #include <semaphore.h>          /* sem_init(), sem_trywait(), sem_post(), sem_de
 445 #if defined(linux)
 446 #define LOADAVG_1MIN    0
 447 #else
 445 #include <sys/loadavg.h>        /* getloadavg() */
 449 #endif /* linux */

 447 /*
 448  *      adjust_pmake_max_jobs (int pmake_max_jobs)
 449  *
 450  *      Parameters:
 451  *              pmake_max_jobs  - max jobs limit set by user
 452  *
 453  *      External functions used:
 454  *              sysconf()
 455  *              getloadavg()
 456  */
 457 static int
 458 adjust_pmake_max_jobs (int pmake_max_jobs)
 459 {
 460         static int      ncpu = 0;
 461         double          loadavg[3];
 462         int             adjustment;
 463         int             adjusted_max_jobs;

 465         if (ncpu <= 0) {
 466                 if ((ncpu = sysconf(_SC_NPROCESSORS_ONLN)) <= 0) {
 467                         ncpu = 1;
 468                 }
 469         }
 470         if (getloadavg(loadavg, 3) != 3) return(pmake_max_jobs);
 471         adjustment = ((int)loadavg[LOADAVG_1MIN]);
 472         if (adjustment < 2) return(pmake_max_jobs);
 473         if (ncpu > 1) {
 474                 adjustment = adjustment / ncpu;
 475         }
 476         adjusted_max_jobs = pmake_max_jobs - adjustment;
 477         if (adjusted_max_jobs < 1) adjusted_max_jobs = 1;
 478         return(adjusted_max_jobs);
 479 }
______unchanged_portion_omitted_

2049 /*
2050  * This function replaces the makesh binary.
2051  */
2052
2053 #ifdef SGE_SUPPORT
2054 #define DO_CHECK(f)     if (f <= 0) { \
2055                                 fprintf(stderr, \
2056                                         catgets(catd, 1, 347, "Could not write t
```



```
2057                                                 script_file, errmsg(errno)); \
2058                                 _exit(1); \
2059                         }
2060 #endif /* SGE_SUPPORT */

2062 static pid_t
2063 run_rule_commands(char *host, char **commands)
2064 {
2065         Boolean         always_exec;
2066         Name            command;
2067         Boolean         ignore;
2068         int             length;
2069         Doname          result;
2070         Boolean         silent_flag;
2071 #ifdef SGE_SUPPORT
2072         wchar_t         *wcmd, *tmp_wcs_buffer = NULL;
2073         char            *cmd, *tmp_mbs_buffer = NULL;
2074         FILE            *scrfp;
2075         Name            shell = getvar(shell_name);
2076 #else
2077         wchar_t         *tmp_wcs_buffer;
2078 #endif /* SGE_SUPPORT */

2080         childPid = fork();
2081         switch (childPid) {
2082         case -1:        /* Error */
2083                 fatal(catgets(catd, 1, 337, "Could not fork child process for dm
2084                       errmsg(errno));
2085                 break;
2086         case 0:         /* Child */
2087                 /* To control the processed targets list is not the child's busi
2088                 running_list = NULL;
2089 #if defined(REDIRECT_ERR)
2090                 if(out_err_same) {
2091                         redirect_io(stdout_file, (char*)NULL);
2092                 } else {
2093                         redirect_io(stdout_file, stderr_file);
2094                 }
2095 #else
2096                 redirect_io(stdout_file, (char*)NULL);
2097 #endif
2098 #ifdef SGE_SUPPORT
2099                 if (grid) {
2100                         int fdes = mkstemp(script_file);
2101                         if ((fdes < 0) || (scrfp = fdopen(fdes, "w")) == NULL) {
2102                                 fprintf(stderr,
2103                                         catgets(catd, 1, 341, "Could not create
2104                                         script_file, errmsg(errno));
2105                                 _exit(1);
2106                         }
2107                         if (IS_EQUAL(shell->string_mb, "")) {
2108                                 shell = shell_name;
2109                         }
2110                 }
2111 #endif /* SGE_SUPPORT */
2112                 for (commands = commands;
2113                      (*commands != (char *)NULL);
2114                      commands++) {
2115                         silent_flag = silent;
2116                         ignore = false;
2117                         always_exec = false;
2118                         while ((**commands == (int) at_char) ||
2119                                (**commands == (int) hyphen_char) ||
2120                                (**commands == (int) plus_char)) {
2121                                 if (**commands == (int) at_char) {
2122                                         silent_flag = true;
```

```
2123                                    }
2124                                    if (**commands == (int) hyphen_char) {
2125                                            ignore = true;
2126                                    }
2127                                    if (**commands == (int) plus_char) {
2128                                            always_exec = true;
2129                                    }
2130                                    (*commands)++;
2131                            }
2132 #ifdef SGE_SUPPORT
2133                            if (grid) {
2134                                    if ((length = strlen(*commands)) >= MAXPATHLEN /
2135                                            wcmd = tmp_wcs_buffer = ALLOC_WC(length
2136                                            (void) mbstowcs(tmp_wcs_buffer, *command
2137                                    } else {
2138                                            MBSTOWCS(wcs_buffer, *commands);
2139                                            wcmd = wcs_buffer;
2140                                            cmd = mbs_buffer;
2141                                    }
2142                                    wchar_t *from = wcmd + wslen(wcmd);
2143                                    wchar_t *to = from + (from - wcmd);
2144                                    *to = (int) nul_char;
2145                                    while (from > wcmd) {
2146                                            *--to = *--from;
2147                                            if (*from == (int) newline_char) { // ne
2148                                                    *--to = *--from;
2149                                            } else if (wschr(char_semantics_char, *f
2150                                                    *--to = (int) backslash_char;
2151                                            }
2152                                    }
2153                                    if (length >= MAXPATHLEN*MB_LEN_MAX/2) { // size
2154                                            cmd = tmp_mbs_buffer = getmem((length *
2155                                            (void) wcstombs(tmp_mbs_buffer, to, (len
2156                                    } else {
2157                                            WCSTOMBS(mbs_buffer, to);
2158                                            cmd = mbs_buffer;
2159                                    }
2160                                    char *mbst, *mbend;
2161                                    if ((length > 0) &&
2162                                        !silent_flag) {
2163                                            for (mbst = cmd; (mbend = strstr(mbst, "
2164                                                    *mbend = '\0';
2165                                                    DO_CHECK(fprintf(scrfp, NOCATGET
2166                                                    *mbend = '\\';
2167                                            }
2168                                            DO_CHECK(fprintf(scrfp, NOCATGETS("/usr/
2169                                    }
2170                                    if (!do_not_exec_rule ||
2171                                        !working_on_targets ||
2172                                        always_exec) {
2177 #if defined(linux)
2178                                            if (0 != strcmp(shell->string_mb, (char*
2179                                                    DO_CHECK(fprintf(scrfp, NOCATGET
2180                                            } else
2181 #endif
2173                                            DO_CHECK(fprintf(scrfp, NOCATGETS("%s -c
2174                                            DO_CHECK(fputs(NOCATGETS("__DMAKECMDEXIT
2175                                            if (ignore) {
2176                                                    DO_CHECK(fprintf(scrfp, NOCATGET
2177                                                            catgets(catd, 1, 343, "\
2178                                                            catgets(catd, 1, 344, "(
2179                                            } else {
2180                                                    DO_CHECK(fprintf(scrfp, NOCATGET
2181                                                            catgets(catd, 1, 342, "\
2182                                            }
2183                                            if (silent_flag) {
```



```
2184                                                    DO_CHECK(fprintf(scrfp, NOCATGET
2185                                                            catgets(catd, 1, 345, "T
2186                                                    for (mbst = cmd; (mbend = strstr
2187                                                            *mbend = '\0';
2188                                                            DO_CHECK(fprintf(scrfp,
2189                                                            *mbend = '\\';
2190                                                    }
2191                                                    DO_CHECK(fprintf(scrfp, NOCATGET
2192                                            }
2193                                            if (!ignore) {
2194                                                    DO_CHECK(fputs(NOCATGETS("\texit
2195                                            }
2196                                            DO_CHECK(fputs(NOCATGETS("fi\n"), scrfp)
2197                                    }
2198                                    if (tmp_wcs_buffer) {
2199                                            retmem_mb(tmp_mbs_buffer);
2200                                            tmp_mbs_buffer = NULL;
2201                                    }
2202                                    if (tmp_wcs_buffer) {
2203                                            retmem(tmp_wcs_buffer);
2204                                            tmp_wcs_buffer = NULL;
2205                                    }
2206                                    continue;
2207                            }
2208 #endif /* SGE_SUPPORT */
2209                            if ((length = strlen(*commands)) >= MAXPATHLEN) {
2210                                    tmp_wcs_buffer = ALLOC_WC(length + 1);
2211                                    (void) mbstowcs(tmp_wcs_buffer, *commands, lengt
2212                                    command = GETNAME(tmp_wcs_buffer, FIND_LENGTH);
2213                                    retmem(tmp_wcs_buffer);
2214                            } else {
2215                                    MBSTOWCS(wcs_buffer, *commands);
2216                                    command = GETNAME(wcs_buffer, FIND_LENGTH);
2217                            }
2218                            if ((command->hash.length > 0) &&
2219                                !silent_flag) {
2220                                    (void) printf("%s\n", command->string_mb);
2221                            }
2222                            result = dosys(command,
2223                                           ignore,
2224                                           false,
2225                                           false, /* bugs #4085164 & #4990057 */
2226                                           /* BOOLEAN(silent_flag && ignore), */
2227                                           always_exec,
2228                                           (Name) NULL,
2229                                           false);
2230                            if (result == build_failed) {
2231                                    if (silent_flag) {
2232                                            (void) printf(catgets(catd, 1, 152, "The
2233                                    }
2234                                    if (!ignore) {
2235                                            _exit(1);
2236                                    }
2237                            }
2238                    }
2239 #ifndef SGE_SUPPORT
2240                    _exit(0);
2241 #else
2242                    if (!grid) {
2243                            _exit(0);
2244                    }
2245                    DO_CHECK(fputs(NOCATGETS("exit 0\n"), scrfp));
2246                    if (fclose(scrfp) != 0) {
2247                            fprintf(stderr,
2248                                    catgets(catd, 1, 346, "Could not close file: %s:
2249                                    script_file, errmsg(errno));
```

```
2250                            _exit(1);
2251                    }
2252                    {

2254 #define DEFAULT_QRSH_TRIES_NUMBER       1
2255 #define DEFAULT_QRSH_TIMEOUT           0

2257                    static char     *sge_env_var = NULL;
2258                    static int      qrsh_tries_number = DEFAULT_QRSH_TRIES_NUMBER;
2259                    static int      qrsh_timeout = DEFAULT_QRSH_TIMEOUT;
2260 #define SGE_DEBUG
2261 #ifdef SGE_DEBUG
2262                    static  Boolean do_not_remove = false;
2263 #endif /* SGE_DEBUG */
2264                    if (sge_env_var == NULL) {
2265                            sge_env_var = getenv(NOCATGETS("__SPRO_DMAKE_SGE_TRIES")
2266                            if (sge_env_var != NULL) {
2267                                    qrsh_tries_number = atoi(sge_env_var);
2268                                    if (qrsh_tries_number < 1 || qrsh_tries_number >
2269                                            qrsh_tries_number = DEFAULT_QRSH_TRIES_N
2270                                    }
2271                            }
2272                            sge_env_var = getenv(NOCATGETS("__SPRO_DMAKE_SGE_TIMEOUT
2273                            if (sge_env_var != NULL) {
2274                                    qrsh_timeout = atoi(sge_env_var);
2275                                    if (qrsh_timeout <= 0) {
2276                                            qrsh_timeout = DEFAULT_QRSH_TIMEOUT;
2277                                    }
2278                            } else {
2279                                    sge_env_var = "";
2280                            }
2281 #ifdef SGE_DEBUG
2282                            sge_env_var = getenv(NOCATGETS("__SPRO_DMAKE_SGE_DEBUG")
2283                            if (sge_env_var == NULL) {
2284                                    sge_env_var = "";
2285                            }
2286                            if (strstr(sge_env_var, NOCATGETS("noqrsh")) != NULL)
2287                                    qrsh_tries_number = 0;
2288                            if (strstr(sge_env_var, NOCATGETS("donotremove")) != NUL
2289                                    do_not_remove = true;
2290 #endif /* SGE_DEBUG */
2291                    }
2292                    for (int i = qrsh_tries_number; ; i--)
2293                    if ((childPid = fork()) < 0) {
2294                            fatal(catgets(catd, 1, 348, "Could not fork child proces
2295                                    errmsg(errno));
2296                            _exit(1);
2297                    } else if (childPid == 0) {
2298                            enable_interrupt((void (*) (int))SIG_DFL);
2299                            if (i > 0) {
2300                                    static char qrsh_cmd[50+MAXPATHLEN] = NOCATGETS(
2301                                    static char *fname_ptr = NULL;
2302                                    static char *argv[] = { NOCATGETS("sh"),
2303                                                            NOCATGETS("-fce"),
2304                                                            qrsh_cmd,
2305                                                            NULL};
2306                                    if (fname_ptr == NULL) {
2307                                            fname_ptr = qrsh_cmd + strlen(qrsh_cmd);
2308                                    }
2309                                    strcpy(fname_ptr, script_file);
2310                                    (void) execve(NOCATGETS("/bin/sh"), argv, enviro
2311                            } else {
2312                                    static char *argv[] = { NOCATGETS("sh"),
2313                                                            script_file,
2314                                                            NULL};
2315                                    (void) execve(NOCATGETS("/bin/sh"), argv, enviro
```



```
2316                            }
2317                            fprintf(stderr,
2318                                    catgets(catd, 1, 349, "Could not load ‘qrsh’: %s
2319                                    errmsg(errno));
2320                            _exit(1);
2321                    } else {
2322                            int             status;
2323                            pid_t pid;
2324                            while ((pid = wait(&status)) != childPid) {
2325                                    if (pid == -1) {
2326                                            fprintf(stderr,
2327                                                    catgets(catd, 1, 350, "wait() fa
2328                                                    errmsg(errno));
2329                                            _exit(1);
2330                                    }
2331                            }
2332                            if (status != 0 && i > 0) {
2333                                    if (i > 1) {
2334                                            sleep(qrsh_timeout);
2335                                    }
2336                                    continue;
2337                            }
2338 #ifdef SGE_DEBUG
2339                            if (do_not_remove) {
2340                                    if (status) {
2341                                            fprintf(stderr,
2342                                                    NOCATGETS("SGE script failed: %s
2343                                                    script_file);
2344                                    }
2345                                    _exit(status ? 1 : 0);
2346                            }
2347 #endif /* SGE_DEBUG */
2348                            (void) unlink(script_file);
2349                            _exit(status ? 1 : 0);
2350                    }
2351                    }
2352 #endif /* SGE_SUPPORT */
2353                    break;
2354            default:
2355                    break;
2356            }
2357            return childPid;
2358 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   11231 Wed May 20 11:27:37 2015
new/usr/src/cmd/make/bin/pmake.cc
make: unifdef for other OSes (undefined)
**********************************************************
```

```
   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 #ifdef TEAMWARE_MAKE_CMN

  28 /*
  29  * Included files
  30  */
  31 #include <arpa/inet.h>
  32 #include <mk/defs.h>
  33 #include <mksh/misc.h>
  34 #include <netdb.h>
  35 #include <netinet/in.h>
  36 #include <sys/socket.h>
  37 #include <sys/stat.h>
  38 #include <sys/types.h>
  39 #include <sys/utsname.h>
  40 #include <rpc/rpc.h>           /* host2netname(), netname2host() */
  41 #ifdef linux
  42 #       include <unistd.h>     /* getdomainname() */
  43 #endif

  42 /*
  43  * Defined macros
  44  */

  46 /*
  47  * typedefs & structs
  48  */

  50 /*
  51  * Static variables
  52  */

  54 /*
  55  * File table of contents
  56  */
  57 static int             get_max(wchar_t **ms_address, wchar_t *hostname);
  58 static Boolean         pskip_comment(wchar_t **cp_address);
```



```
  59 static void            pskip_till_next_word(wchar_t **cp);
  60 static Boolean         pskip_white_space(wchar_t **cp_address);


  63 /*
  64  *      read_make_machines(Name make_machines_name)
  65  *
  66  *      For backwards compatibility w/ PMake 1.x, when DMake 2.x is
  67  *      being run in parallel mode, DMake should parse the PMake startup
  68  *      file $(HOME)/.make.machines to get the PMake max jobs.
  69  *
  70  *      Return value:
  71  *              int of PMake max jobs
  72  *
  73  *      Parameters:
  74  *              make_machines_name      Name of .make.machines file
  75  *
  76  */
  77 int
  78 read_make_machines(Name make_machines_name)
  79 {
  80         wchar_t                 c;
  81         Boolean                 default_make_machines;
  82         struct hostent          *hp;
  83         wchar_t                 local_host[MAX_HOSTNAMELEN + 1];
  84         char                    local_host_mb[MAX_HOSTNAMELEN + 1] = "";
  85         int                     local_host_wslen;
  86         wchar_t                 full_host[MAXNETNAMELEN + 1];
  87         int                     full_host_wslen = 0;
  88         char                    *homedir;
  89         Name                    MAKE_MACHINES;
  90         struct stat             make_machines_buf;
  91         FILE                    *make_machines_file;
  92         wchar_t                 *make_machines_list = NULL;
  93         char                    *make_machines_list_mb = NULL;
  94         wchar_t                 make_machines_path[MAXPATHLEN];
  95         char                    mb_make_machines_path[MAXPATHLEN];
  96         wchar_t                 *mp;
  97         wchar_t                 *ms;
  98         int                     pmake_max_jobs = 0;
  99         struct utsname          uts_info;


 102         MBSTOWCS(wcs_buffer, NOCATGETS("MAKE_MACHINES"));
 103         MAKE_MACHINES = GETNAME(wcs_buffer, FIND_LENGTH);
 104         /* Did the user specify a .make.machines file on the command line? */
 105         default_make_machines = false;
 106         if (make_machines_name == NULL) {
 107                 /* Try reading the default .make.machines file, in $(HOME). */
 108                 homedir = getenv(NOCATGETS("HOME"));
 109                 if ((homedir != NULL) && (strlen(homedir) < (sizeof(mb_make_mach
 110                         sprintf(mb_make_machines_path,
 111                          NOCATGETS("%s/.make.machines"), homedir);
 112                         MBSTOWCS(make_machines_path, mb_make_machines_path);
 113                         make_machines_name = GETNAME(make_machines_path, FIND_LE
 114                         default_make_machines = true;
 115                 }
 116                 if (make_machines_name == NULL) {
 117                         /*
 118                          * No $(HOME)/.make.machines file.
 119                          * Return 0 for PMake max jobs.
 120                          */
 121                         return(0);
 122                 }
 123         }
 124 /*
```

```
125         make_machines_list_mb = getenv(MAKE_MACHINES->string_mb);
126   */
127         /* Open the .make.machines file. */
128         if ((make_machines_file = fopen(make_machines_name->string_mb, "r")) ==
129                 if (!default_make_machines) {
130                         /* Error opening .make.machines file. */
131                         fatal(catgets(catd, 1, 314, "Open of %s failed: %s"),
132                               make_machines_name->string_mb,
133                               errmsg(errno));
134                 } else {
135                         /*
136                          * No $(HOME)/.make.machines file.
137                          * Return 0 for PMake max jobs.
138                          */
139                         return(0);
140                 }
141         /* Stat the .make.machines file to get the size of the file.  */
142         } else if (fstat(fileno(make_machines_file), &make_machines_buf) < 0) {
143                 /* Error stat'ing .make.machines file. */
144                 fatal(catgets(catd, 1, 315, "Stat of %s failed: %s"),
145                       make_machines_name->string_mb,
146                       errmsg(errno));
147         } else {
148                 /* Allocate memory for "MAKE_MACHINES=<contents of .m.m>" */
149                 make_machines_list_mb =
150                   (char *) getmem((int) (strlen(MAKE_MACHINES->string_mb) +
151                                          2 +
152                                          make_machines_buf.st_size));
153                 sprintf(make_machines_list_mb,
154                         "%s=",
155                         MAKE_MACHINES->string_mb);
156                 /* Read in the .make.machines file. */
157                 if (fread(make_machines_list_mb + strlen(MAKE_MACHINES->string_m
158                           sizeof(char),
159                           (int) make_machines_buf.st_size,
160                           make_machines_file) != make_machines_buf.st_size) {
161                         /*
162                          * Error reading .make.machines file.
163                          * Return 0 for PMake max jobs.
164                          */
165                         warning(catgets(catd, 1, 316, "Unable to read %s"),
166                                 make_machines_name->string_mb);
167                         (void) fclose(make_machines_file);
168                         retmem_mb((caddr_t) make_machines_list_mb);
169                         return(0);
170                 } else {
171                         (void) fclose(make_machines_file);
172                         /* putenv "MAKE_MACHINES=<contents of .m.m>" */
173                         *(make_machines_list_mb +
174                           strlen(MAKE_MACHINES->string_mb) +
175                           1 +
176                           make_machines_buf.st_size) = (int) nul_char;
177                         if (putenv(make_machines_list_mb) != 0) {
178                                 warning(catgets(catd, 1, 317, "Couldn't put cont
179                                         make_machines_name->string_mb);
180                         } else {
181                                 make_machines_list_mb += strlen(MAKE_MACHINES->s
182                                 make_machines_list = ALLOC_WC(strlen(make_machin
183                                 (void) mbstowcs(make_machines_list,
184                                                 make_machines_list_mb,
185                                                 (strlen(make_machines_list_mb) +
186                         }
187                 }
188         }

190         uname(&uts_info);
```



```
191         strcpy(local_host_mb, &uts_info.nodename[0]);
192         MBSTOWCS(local_host, local_host_mb);
193         local_host_wslen = wslen(local_host);

195         // There is no getdomainname() function on Solaris.
196         // And netname2host() function does not work on Linux.
197         // So we have to use different APIs.
201 #ifdef linux
202         if (getdomainname(mbs_buffer, MAXNETNAMELEN+1) == 0) {
203                 sprintf(mbs_buffer2, "%s.%s", local_host_mb, mbs_buffer);
204 #else
198         if (host2netname(mbs_buffer, NULL, NULL) &&
199             netname2host(mbs_buffer, mbs_buffer2, MAXNETNAMELEN+1)) {
207 #endif
200                 MBSTOWCS(full_host, mbs_buffer2);
201                 full_host_wslen = wslen(full_host);
202         }

204         for (ms = make_machines_list;
205              (ms) && (*ms );
206              ) {
207                 /*
208                  * Skip white space and comments till you reach
209                  * a machine name.
210                  */
211                 pskip_till_next_word(&ms);

213                 /*
214                  * If we haven't reached the end of file, process the
215                  * machine name.
216                  */
217                 if (*ms) {
218                         /*
219                          * If invalid machine name decrement counter
220                          * and skip line.
221                          */
222                         mp = ms;
223                         SKIPWORD(ms);
224                         c = *ms;
225                         *ms++ = '\0'; /* Append null to machine name. */
226                         /*
227                          * If this was the beginning of a comment
228                          * (we overwrote a # sign) and it's not
229                          * end of line yet, shift the # sign.
230                          */
231                         if ((c == '#') && (*ms != '\n') && (*ms)) {
232                                 *ms = '#';
233                         }
234                         WCSTOMBS(mbs_buffer, mp);
235                         /*
236                          * Print "Ignoring unknown host" if:
237                          * 1) hostname is longer than MAX_HOSTNAMELEN, or
238                          * 2) hostname is unknown
239                          */
240                         if ((wslen(mp) > MAX_HOSTNAMELEN) ||
241                             ((hp = gethostbyname(mbs_buffer)) == NULL)) {
242                                 warning(catgets(catd, 1, 318, "Ignoring unknown
243                                         mbs_buffer);
244                                 SKIPTOEND(ms);
245                                 /* Increment ptr if not end of file. */
246                                 if (*ms) {
247                                         ms++;
248                                 }
249                         } else {
250                                 /* Compare current hostname with local_host. */
251                                 if (wslen(mp) == local_host_wslen &&
```

```
252                                     IS_WEQUALN(mp, local_host, local_host_wslen)
253                                         /*
254                                          * Bingo, local_host is in .make.machine
255                                          * Continue reading.
256                                          */
257                                         pmake_max_jobs = PMAKE_DEF_MAX_JOBS;
258                                 /* Compare current hostname with full_host. */
259                                 } else if (wslen(mp) == full_host_wslen &&
260                                           IS_WEQUALN(mp, full_host, full_host_w
261                                         /*
262                                          * Bingo, full_host is in .make.machines
263                                          * Continue reading.
264                                          */
265                                         pmake_max_jobs = PMAKE_DEF_MAX_JOBS;
266                                 } else {
267                                         if (c != '\n') {
268                                             SKIPTOEND(ms);
269                                             if (*ms) {
270                                                 ms++;
271                                             }
272                                         }
273                                         continue;
274                                 }
275                                 /* If we get here, local_host is in .make.machin
276                                 if (c != '\n')  {
277                                         /* Now look for keyword 'max'. */
278                                         MBSTOWCS(wcs_buffer, NOCATGETS("max"));
279                                         SKIPSPACE(ms);
280                                         while ((*ms != '\n') && (*ms)) {
281                                                 if (*ms == '#') {
282                                                         pskip_comment(&ms);
283                                                 } else if (IS_WEQUALN(ms, wcs_bu
284                                                         /* Skip "max". */
285                                                         ms += 3;
286                                                         pmake_max_jobs = get_max
287                                                         SKIPSPACE(ms);
288                                                 } else {
289                                                         warning(catgets(catd, 1,
290                                                         SKIPTOEND(ms);
291                                                         break;
292                                                 }
293                                         }
294                                 }
295                                 break; /* out of outermost for() loop. */
296                         }
297                 }
298         }
299         retmem(make_machines_list);
300         return(pmake_max_jobs);
301 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   57159 Wed May 20 11:27:37 2015
new/usr/src/cmd/make/bin/read.cc
make: unifdef for other OSes (undefined)
**********************************************************
```

```
   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 2006 Sun Microsystems, Inc. All rights reserved.
  23  * Use is subject to license terms.
  24  */

  26 /*
  27  *      read.c
  28  *
  29  *      This file contains the makefile reader.
  30  */

  32 /*
  33  * Included files
  34  */
  35 #include <avo/avo_alloca.h>             /* alloca() */
  36 #include <errno.h>              /* errno */
  37 #include <fcntl.h>              /* fcntl() */
  38 #include <mk/defs.h>
  39 #include <mksh/macro.h>         /* expand_value(), expand_macro() */
  40 #include <mksh/misc.h>          /* getmem() */
  41 #include <mksh/read.h>          /* get_next_block_fn() */
  42 #include <sys/uio.h>            /* read() */
  43 #include <unistd.h>             /* read(), unlink() */

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

  46 /*
  47  * typedefs & structs
  48  */

  50 /*
  51  * Static variables
  52  */

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

  56 /*
  57  * File table of contents
```



```
  58  */
  59 static  void            parse_makefile(register Name true_makefile_name, registe
  60 static  Source          push_macro_value(register Source bp, register wchar_t *b
  61 extern  void            enter_target_groups_and_dependencies(Name_vector target,
  62 extern  Name            normalize_name(register wchar_t *name_string, register i

  64 /*
  65  *      read_simple_file(makefile_name, chase_path, doname_it,
  66  *              complain, must_exist, report_file, lock_makefile)
  67  *
  68  *      Make the makefile and setup to read it. Actually read it if it is stdio
  69  *
  70  *      Return value:
  71  *                              false if the read failed
  72  *
  73  *      Parameters:
  74  *              makefile_name   Name of the file to read
  75  *              chase_path      Use the makefile path when opening file
  76  *              doname_it       Call doname() to build the file first
  77  *              complain        Print message if doname/open fails
  78  *              must_exist      Generate fatal if file is missing
  79  *              report_file     Report file when running -P
  80  *              lock_makefile   Lock the makefile when reading
  81  *
  82  *      Static variables used:
  83  *
  84  *      Global variables used:
  85  *              do_not_exec_rule Is -n on?
  86  *              file_being_read Set to the name of the new file
  87  *              line_number     The number of the current makefile line
  88  *              makefiles_used  A list of all makefiles used, appended to
  89  */


  92 Boolean
  93 read_simple_file(register Name makefile_name, register Boolean chase_path, regis
  94 {
  95         static short            max_include_depth;
  96         register Property       makefile = maybe_append_prop(makefile_name,
  97                                                              makefile_prop);
  98         Boolean                 forget_after_parse = false;
  99         static pathpt           makefile_path;
 100         register int            n;
 101         char                    *path;
 102         register Source         source = ALLOC(Source);
 103         Property                orig_makefile = makefile;
 104         Dependency              *dpp;
 105         Dependency              dp;
 106         register int            length;
 107         wchar_t                 *previous_file_being_read = file_being_read;
 108         int                     previous_line_number = line_number;
 109         wchar_t                 previous_current_makefile[MAXPATHLEN];
 110         Makefile_type           save_makefile_type;
 111         Name                    normalized_makefile_name;
 112         register wchar_t        *string_start;
 113         register wchar_t        *string_end;


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

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

 119 #ifdef NSE
```

```
120         if (report_file){
121                 wscpy(previous_current_makefile, current_makefile);
122                 wscpy(current_makefile, wcb);
123         }
124 #endif
125         if (max_include_depth++ >= 40) {
126                 fatal(catgets(catd, 1, 66, "Too many nested include statements")
127         }
128         if (makefile->body.makefile.contents != NULL) {
129                 retmem(makefile->body.makefile.contents);
130         }
131         source->inp_buf =
132           source->inp_buf_ptr =
133             source->inp_buf_end = NULL;
134         source->error_converting = false;
135         makefile->body.makefile.contents = NULL;
136         makefile->body.makefile.size = 0;
137         if ((makefile_name->hash.length != 1) ||
138             (wcb[0] != (int) hyphen_char)) {
139                 if ((makefile->body.makefile.contents == NULL) &&
140                     (doname_it)) {
141                         if (makefile_path == NULL) {
142                                 add_dir_to_path(".",
143                                                 &makefile_path,
144                                                 -1);
145                                 add_dir_to_path(NOCATGETS("/usr/share/lib/make")
146                                                 &makefile_path,
147                                                 -1);
148                                 add_dir_to_path(NOCATGETS("/etc/default"),
149                                                 &makefile_path,
150                                                 -1);
151                         }
152                         save_makefile_type = makefile_type;
153                         makefile_type = reading_nothing;
154                         if (doname(makefile_name, true, false) == build_dont_kno
155                                 /* Try normalized filename */
156                                 string_start=get_wstring(makefile_name->string_m
157                                 for (string_end=string_start+1; *string_end != L
158                                 normalized_makefile_name=normalize_name(string_s
159                                 if ((strcmp(makefile_name->string_mb, normalized
160                                         (doname(normalized_makefile_name, true,
161                                         n = access_vroot(makefile_name->string_m
162                                                  4,
163                                                  chase_path ?
164                                                  makefile_path : NULL,
165                                                  VROOT_DEFAULT);
166                                         if (n == 0) {
167                                                 get_vroot_path((char **) NULL,
168                                                        &path,
169                                                        (char **) NULL);
170                                                 if ((path[0] == (int) period_cha
171                                                     (path[1] == (int) slash_char
172                                                         path += 2;
173                                                 }
174                                                 MBSTOWCS(wcs_buffer, path);
175                                                 makefile_name = GETNAME(wcs_buff
176                                                                 FIND_LENGTH);
177                                         }
178                                 }
179                                 retmem(string_start);
180                                 /*
181                                  * Commented out: retmem_mb(normalized_makefile_
182                                  * We have to return this memory, but it seems t
183                                  * in dmake or in Sun C++ 5.7 compiler (it works
184                                  * is compiled using Sun C++ 5.6).
185                                  */
```



```
186                                 // retmem_mb(normalized_makefile_name->string_mb
187                         }
188                         makefile_type = save_makefile_type;
189                 }
190                 source->string.free_after_use = false;
191                 source->previous = NULL;
192                 source->already_expanded = false;
193                 /* Lock the file for read, but not when -n. */
194                 if (lock_makefile &&
195                     !do_not_exec_rule) {
196
197                         make_state_lockfile = getmem(strlen(make_state->string_
198                         (void) sprintf(make_state_lockfile,
199                                        NOCATGETS("%s.lock"),
200                                        make_state->string_mb);
201                         (void) file_lock(make_state->string_mb,
202                                          make_state_lockfile,
203                                          (int *) &make_state_locked,
204                                          0);
205                         if(!make_state_locked) {
206                                 printf(NOCATGETS("-- NO LOCKING for read\n"));
207                                 retmem_mb(make_state_lockfile);
208                                 make_state_lockfile = 0;
209                                 return failed;
210                         }
211                 }
212                 if (makefile->body.makefile.contents == NULL) {
213                         save_makefile_type = makefile_type;
214                         makefile_type = reading_nothing;
215                         if ((doname_it) &&
216                             (doname(makefile_name, true, false) == build_failed)
217                                 if (complain) {
218                                         (void) fprintf(stderr,
219 #ifdef DISTRIBUTED
220                                                        catgets(catd, 1, 67, "dma
221 #else
222                                                        catgets(catd, 1, 237, "ma
223 #endif
224                                                        makefile_name->string_mb)
225                                 }
226                                 max_include_depth--;
227                                 makefile_type = save_makefile_type;
228                                 return failed;
229                         }
230                         makefile_type = save_makefile_type;
231                         //
232                         // Before calling exists() make sure that we have the ri
233                         //
234                         makefile_name->stat.time = file_no_time;
235
236                         if (exists(makefile_name) == file_doesnt_exist) {
237                                 if (complain ||
238                                     (makefile_name->stat.stat_errno != ENOENT))
239                                         if (must_exist) {
240                                                 fatal(catgets(catd, 1, 68, "Can'
241                                                       makefile_name->string_mb,
242                                                       errmsg(makefile_name->
243                                                              stat.stat_errno));
244                                         } else {
245                                                 warning(catgets(catd, 1, 69, "Ca
246                                                         makefile_name->string_mb
247                                                         errmsg(makefile_name->
248                                                                stat.stat_errno))
249                                         }
250                                 }
251                                 max_include_depth--;
```

```
252                             if(make_state_locked && (make_state_lockfile !=
253                                     (void) unlink(make_state_lockfile);
254                                     retmem_mb(make_state_lockfile);
255                                     make_state_lockfile = NULL;
256                                     make_state_locked = false;
257                             }
258                             retmem(wcb);
259                             retmem_mb((char *)source);
260                             return failed;
261                     }
262                     /*
263                      * These values are the size and bytes of
264                      * the MULTI-BYTE makefile.
265                      */
266                     orig_makefile->body.makefile.size =
267                       makefile->body.makefile.size =
268                         source->bytes_left_in_file =
269                           makefile_name->stat.size;
270                     if (report_file) {
271                             for (dpp = &makefiles_used;
272                                  *dpp != NULL;
273                                  dpp = &(*dpp)->next);
274                             dp = ALLOC(Dependency);
275                             dp->next = NULL;
276                             dp->name = makefile_name;
277                             dp->automatic = false;
278                             dp->stale = false;
279                             dp->built = false;
280                             *dpp = dp;
281                     }
282                     source->fd = open_vroot(makefile_name->string_mb,
283                                             O_RDONLY,
284                                             0,
285                                             NULL,
286                                             VROOT_DEFAULT);
287                     if (source->fd < 0) {
288                             if (complain || (errno != ENOENT)) {
289                                     if (must_exist) {
290                                             fatal(catgets(catd, 1, 70, "Can'
291                                                   makefile_name->string_mb,
292                                                   errmsg(errno));
293                                     } else {
294                                             warning(catgets(catd, 1, 71, "Ca
295                                                     makefile_name->string_mb
296                                                     errmsg(errno));
297                                     }
298                             }
299                             max_include_depth--;
300                             return failed;
301                     }
302                     (void) fcntl(source->fd, F_SETFD, 1);
303                     orig_makefile->body.makefile.contents =
304                       makefile->body.makefile.contents =
305                         source->string.text.p =
306                           source->string.buffer.start =
307                             ALLOC_WC((int) (makefile_name->stat.size + 2));
308                     if (makefile_type == reading_cpp_file) {
309                             forget_after_parse = true;
310                     }
311                     source->string.text.end = source->string.text.p;
312                     source->string.buffer.end =
313                       source->string.text.p + makefile_name->stat.size;
314             } else {
315                     /* Do we ever reach here? */
316                     source->fd = -1;
317                     source->string.text.p =
```



```
318                       source->string.buffer.start =
319                         makefile->body.makefile.contents;
320                     source->string.text.end =
321                       source->string.buffer.end =
322                         source->string.text.p + makefile->body.makefile.size
323                     source->bytes_left_in_file =
324                       makefile->body.makefile.size;
325             }
326             file_being_read = wcb;
327     } else {
328             char            *stdin_text_p;
329             char            *stdin_text_end;
330             char            *stdin_buffer_start;
331             char            *stdin_buffer_end;
332             char            *p_mb;
333             int             num_mb_chars;
334             size_t          num_wc_chars;
335
336             MBSTOWCS(wcs_buffer, NOCATGETS("Standard in"));
337             makefile_name = GETNAME(wcs_buffer, FIND_LENGTH);
338             /*
339              * Memory to read standard in, then convert it
340              * to wide char strings.
341              */
342             stdin_buffer_start =
343               stdin_text_p = getmem(length = 1024);
344             stdin_buffer_end = stdin_text_p + length;
345             MBSTOWCS(wcs_buffer, NOCATGETS("standard input"));
346             file_being_read = (wchar_t *) wsdup(wcs_buffer);
347             line_number = 0;
348             while ((n = read(fileno(stdin),
349                              stdin_text_p,
350                              length)) > 0) {
351                     length -= n;
352                     stdin_text_p += n;
353                     if (length == 0) {
354                             p_mb = getmem(length = 1024 +
355                                           (stdin_buffer_end -
356                                            stdin_buffer_start));
357                             (void) strncpy(p_mb,
358                                            stdin_buffer_start,
359                                            (stdin_buffer_end -
360                                             stdin_buffer_start));
361                             retmem_mb(stdin_buffer_start);
362                             stdin_text_p = p_mb +
363                               (stdin_buffer_end - stdin_buffer_start);
364                             stdin_buffer_start = p_mb;
365                             stdin_buffer_end =
366                               stdin_buffer_start + length;
367                             length = 1024;
368                     }
369             }
370             if (n < 0) {
371                     fatal(catgets(catd, 1, 72, "Error reading standard input
372                           errmsg(errno));
373             }
374             stdin_text_p = stdin_buffer_start;
375             stdin_text_end = stdin_buffer_end - length;
376             num_mb_chars = stdin_text_end - stdin_text_p;
377
378             /*
379              * Now, convert the sequence of multibyte chars into
380              * a sequence of corresponding wide character codes.
381              */
382             source->string.free_after_use = false;
383             source->previous = NULL;
```

```
 384                 source->bytes_left_in_file = 0;
 385                 source->fd = -1;
 386                 source->already_expanded = false;
 387                 source->string.buffer.start =
 388                   source->string.text.p = ALLOC_WC(num_mb_chars + 1);
 389                 source->string.buffer.end =
 390                     source->string.text.p + num_mb_chars;
 391                 num_wc_chars = mbstowcs(source->string.text.p,
 392                                         stdin_text_p,
 393                                         num_mb_chars);
 394                 if ((int) num_wc_chars >= 0) {
 395                         source->string.text.end =
 396                           source->string.text.p + num_wc_chars;
 397                 }
 398                 (void) retmem_mb(stdin_text_p);
 399         }
 400         line_number = 1;
 401         if (trace_reader) {
 402                 (void) printf(catgets(catd, 1, 73, ">>>>>>>>>>>>>>>> Reading mak
 403                               makefile_name->string_mb);
 404         }
 405         parse_makefile(makefile_name, source);
 406         if (trace_reader) {
 407                 (void) printf(catgets(catd, 1, 74, ">>>>>>>>>>>>>>>> End of make
 408                               makefile_name->string_mb);
 409         }
 410 #ifdef NSE
 411         if (report_file && (previous_current_makefile[0] != NULL)) {
 412                 wscpy(current_makefile, previous_current_makefile);
 413         }
 414 #endif
 415         if(file_being_read) {
 416                 retmem(file_being_read);
 417         }
 418         file_being_read = previous_file_being_read;
 419         line_number = previous_line_number;
 420         makefile_type = reading_nothing;
 421         max_include_depth--;
 422         if (make_state_locked) {
 423                 /* Unlock .make.state. */
 424                 unlink(make_state_lockfile);
 425                 make_state_locked = false;
 426                 retmem_mb(make_state_lockfile);
 427         }
 428         if (forget_after_parse) {
 429                 retmem(makefile->body.makefile.contents);
 430                 makefile->body.makefile.contents = NULL;
 431         }
 432         retmem_mb((char *)source);
 433         return succeeded;
 434 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
    1641 Wed May 20 11:27:38 2015
new/usr/src/cmd/make/include/avo/intl.h
make: unifdef for other OSes (undefined)
**********************************************************
```

```
   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 2001 Sun Microsystems, Inc. All rights reserved.
  23  * Use is subject to license terms.
  24  */

  26 #ifndef _AVO_INTL_H
  27 #define _AVO_INTL_H

  29 #if defined(SUN4_x) || defined(HP_UX)
  30 #include <avo/widefake.h>
  31 #endif

  30 /*
  31  * For catgets
  32  */
  33 #include <nl_types.h>

  38 #ifdef HP_UX
  39 #ifdef __cplusplus
  40 #ifndef _STDLIB_INCLUDED
  41 #include <stdlib.h>             /* for wchar_t definition and HP-UX - */
  42 #endif                          /* wide character function prototypes. */
  43 extern "C" {
  44 char *gettext(char *msg);
  45 char *dgettext(const char *, const char *);
  46 char *bindtextdomain(const char *, const char *);
  47 char *textdomain(char *);
  48 }
  49 #endif /* __cplusplus */
  50 #endif

  36 /*
  37  * NOCATGETS is a dummy macro that returns it argument.
  38  * It is used to identify strings that we consciously do not
  39  * want to apply catgets() to.  We have tools that check the
  40  * sources for strings that are not catgets'd and the tools
  41  * ignore strings that are NOCATGETS'd.
  42  */
  43 #define NOCATGETS(str)  (str)

  45 /*
```



```
  46  * Define the various text domains
  47  */
  48 #define AVO_DOMAIN_CODEMGR      "codemgr"
  49 #define AVO_DOMAIN_VERTOOL      "vertool"
  50 #define AVO_DOMAIN_FILEMERGE    "filemerge"
  51 #define AVO_DOMAIN_DMAKE        "dmake"
  52 #define AVO_DOMAIN_PMAKE        "pmake"
  53 #define AVO_DOMAIN_FREEZEPOINT  "freezept"
  54 #define AVO_DOMAIN_MAKETOOL     "maketool"

  56 #endif
```

```
**********************************************************
    1302 Wed May 20 11:27:38 2015
new/usr/src/cmd/make/include/bsd/bsd.h
make: unifdef for other OSes (undefined)
**********************************************************
```

```
/*
 * 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 2003 Sun Microsystems, Inc. All rights reserved.
 * Use is subject to license terms.
 */

/*
 * bsd/bsd.h: Interface definitions to BSD compatibility functions for SVR4.
 */

#ifndef _BSD_BSD_H
#define _BSD_BSD_H

#include <signal.h>

#if defined (HP_UX) || defined (linux)
typedef void SIG_FUNC_TYP(int);
typedef SIG_FUNC_TYP *SIG_TYP;
#define SIG_PF SIG_TYP
#endif

#ifndef __cplusplus
typedef void (*SIG_PF) (int);
#endif

#ifdef __cplusplus
extern "C" SIG_PF bsd_signal(int a, SIG_PF b);
#else
extern void (*bsd_signal(int, void (*) (int))) (int);
#endif
extern void bsd_signals(void);

#endif
```

****
16494 Wed May 20 11:27:38 2015
new/usr/src/cmd/make/include/mk/defs.h
make: unifdef for other OSes (undefined)
****
_____***unchanged_portion_omitted_***

```
177 /*
178  * Typedefs for all structs
179  */
180 typedef struct _Cmd_line        *Cmd_line, Cmd_line_rec;
181 typedef struct _Dependency      *Dependency, Dependency_rec;
182 typedef struct _Macro           *Macro, Macro_rec;
183 typedef struct _Name_vector     *Name_vector, Name_vector_rec;
184 typedef struct _Percent         *Percent, Percent_rec;
185 typedef struct _Dyntarget       *Dyntarget;
186 typedef struct _Recursive_make  *Recursive_make, Recursive_make_rec;
187 typedef struct _Running         *Running, Running_rec;


190 /*
191  *      extern declarations for all global variables.
192  *      The actual declarations are in globals.cc
193  */
194 extern  Boolean         allrules_read;
195 extern  Name            posix_name;
196 extern  Name            svr4_name;
197 extern  Boolean         sdot_target;
198 extern  Boolean         all_parallel;
199 extern  Boolean         assign_done;
200 extern  Boolean         build_failed_seen;
201 #ifdef DISTRIBUTED
202 extern  Boolean         building_serial;
203 #endif
204 extern  Name            built_last_make_run;
205 extern  Name            c_at;
206 #ifdef DISTRIBUTED
207 extern  Boolean         called_make;
208 #endif
209 extern  Boolean         command_changed;
210 extern  Boolean         commands_done;
211 extern  Chain           conditional_targets;
212 extern  Name            conditionals;
213 extern  Boolean         continue_after_error;
214 extern  Property        current_line;
215 extern  Name            current_make_version;
216 extern  Name            current_target;
217 extern  short           debug_level;
218 extern  Cmd_line        default_rule;
219 extern  Name            default_rule_name;
220 extern  Name            default_target_to_build;
221 extern  Boolean         depinfo_already_read;
222 extern  Name            dmake_group;
223 extern  Name            dmake_max_jobs;
224 extern  Name            dmake_mode;
225 extern  DMake_mode      dmake_mode_type;
226 extern  Name            dmake_output_mode;
227 extern  DMake_output_mode       output_mode;
228 extern  Name            dmake_odir;
229 extern  Name            dmake_rcfile;
230 extern  Name            done;
231 extern  Name            dot;
232 extern  Name            dot_keep_state;
233 extern  Name            dot_keep_state_file;
234 extern  Name            empty_name;
```



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

```
#ifdef SGE_SUPPORT
extern  char            script_file[];
#endif
extern  Boolean         stdout_stderr_same;
extern  Dependency      suffixes;
extern  Name            suffixes_name;
extern  Name            sunpro_dependencies;
extern  Boolean         target_variants;
extern  const char      *tmpdir;
extern  const char      *temp_file_directory;
extern  Name            temp_file_name;
extern  short           temp_file_number;
extern  wchar_t         *top_level_target;
extern  Boolean         touch;
extern  Boolean         trace_reader;
extern  Boolean         build_unconditional;
extern  pathpt          vroot_path;
extern  Name            wait_name;
extern  wchar_t         wcs_buffer2[];
extern  wchar_t         *wcs_ptr;
extern  wchar_t         *wcs_ptr2;
extern  nl_catd         catd;
extern  long int        hostid;

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

/*
 * Declarations of system supplied functions
 */
extern  int             file_lock(char *, char *, int *, int);

/*
 * Declarations of functions declared and used by make
 */
extern  void            add_pending(Name target, int recursion_level, Boolean do
extern  void            add_running(Name target, Name true_target, Property comm
extern  void            add_serial(Name target, int recursion_level, Boolean do_
extern  void            add_subtree(Name target, int recursion_level, Boolean do
extern  void            append_or_replace_macro_in_dyn_array(ASCII_Dyn_Array *Ar
#ifdef DISTRIBUTED
extern  Doname          await_dist(Boolean waitflg);
#endif
#ifdef TEAMWARE_MAKE_CMN
extern  void            await_parallel(Boolean waitflg);
#endif
extern  void            build_suffix_list(Name target_suffix);
extern  Boolean         check_auto_dependencies(Name target, int auto_count, Nam
extern  void            check_state(Name temp_file_name);
extern  void            cond_macros_into_string(Name np, String_rec *buffer);
extern  void            construct_target_string();
extern  void            create_xdrs_ptr(void);
extern  void            depvar_add_to_list (Name name, Boolean cmdline);
#ifdef DISTRIBUTED
extern  void            distribute_rxm(Avo_DoJobMsg *dmake_job_msg);
extern  int             getRxmMessage(void);
extern  Avo_JobResultMsg* getJobResultMsg(void);
extern  Avo_AcknowledgeMsg* getAcknowledgeMsg(void);
#endif
extern  Doname          doname(register Name target, register Boolean do_get, re
extern  Doname          doname_check(register Name target, register Boolean do_g
```



```
extern  Doname          doname_parallel(Name target, Boolean do_get, Boolean imp
extern  Doname          dosys(register Name command, register Boolean ignore_err
extern  void            dump_make_state(void);
extern  void            dump_target_list(void);
extern  void            enter_conditional(register Name target, Name name, Name
extern  void            enter_dependencies(register Name target, Chain target_gr
extern  void            enter_dependency(Property line, register Name depe, Bool
extern  void            enter_equal(Name name, Name value, register Boolean appe
extern  Percent         enter_percent(register Name target, Chain target_group,
extern  Dyntarget       enter_dyntarget(register Name target);
extern  Name_vector     enter_name(String string, Boolean tail_present, register
extern  Boolean         exec_vp(register char *name, register char **argv, char
extern  Doname          execute_parallel(Property line, Boolean waitflg, Boolean
extern  Doname          execute_serial(Property line);
extern  timestruc_t&    exists(register Name target);
extern  void            fatal(char *, ...);
extern  void            fatal_reader(char *, ...);
extern  Doname          find_ar_suffix_rule(register Name target, Name true_targ
extern  Doname          find_double_suffix_rule(register Name target, Property *
extern  Doname          find_percent_rule(register Name target, Property *comman
extern  int             find_run_directory (char *cmd, char *cwd, char *dir, cha
extern  Doname          find_suffix_rule(Name target, Name target_body, Name tar
extern  Chain           find_target_groups(register Name_vector target_list, reg
extern  void            finish_children(Boolean docheck);
extern  void            finish_running(void);
extern  void            free_chain(Name_vector ptr);
extern  void            gather_recursive_deps(void);
extern  char            *get_current_path(void);
extern  int             get_job_msg_id(void);
extern  FILE            *get_mtool_msgs_fp(void);
#ifdef DISTRIBUTED
extern  Boolean         get_dmake_group_specified(void);
extern  Boolean         get_dmake_max_jobs_specified(void);
extern  Boolean         get_dmake_mode_specified(void);
extern  Boolean         get_dmake_odir_specified(void);
extern  Boolean         get_dmake_rcfile_specified(void);
extern  Boolean         get_pmake_machinesfile_specified(void);
#endif
#if defined(DISTRIBUTED) || defined(MAKETOOL) /* tolik */
extern  XDR             *get_xdrs_ptr(void);
#endif
extern  wchar_t         *getmem_wc(register int size);
#if !defined(linux)
/* On linux getwd(char *) is defined in 'unistd.h' */
#ifdef __cplusplus
extern "C" {
#endif
extern  char            *getwd(char *);
#ifdef __cplusplus
}
#endif
#endif
extern  void            handle_interrupt(int);
extern  Boolean         is_running(Name target);
extern  void            load_cached_names(void);
extern  Boolean         parallel_ok(Name target, Boolean line_prop_must_exists);
extern  void            print_dependencies(register Name target, register Proper
extern  void            send_job_start_msg(Property line);
extern  void            send_rsrc_info_msg(int max_jobs, char *hostname, char *u
extern  void            print_value(register Name value, Daemon daemon);
extern  timestruc_t&    read_archive(register Name target);
extern  int             read_dir(Name dir, wchar_t *pattern, Property line, wcha
extern  void            read_directory_of_file(register Name file);
extern  int             read_make_machines(Name make_machines_name);
extern  Boolean         read_simple_file(register Name makefile_name, register B
extern  void            remove_recursive_dep(Name target);
```

```
429 extern  void            report_recursive_dep(Name target, char *line);
430 extern  void            report_recursive_done(void);
431 extern  void            report_recursive_init(void);
432 extern  Recursive_make  find_recursive_target(Name target);
433 extern  void            reset_locals(register Name target, register Property old
434 extern  void            set_locals(register Name target, register Property old_l
435 extern  void            setvar_append(register Name name, register Name value);
436 #ifdef DISTRIBUTED
437 extern  void            setvar_envvar(Avo_DoJobMsg *dmake_job_msg);
438 #else
439 extern  void            setvar_envvar(void);
440 #endif
441 extern  void            special_reader(Name target, register Name_vector depes,
442 extern  void            startup_rxm();
443 extern  Doname          target_can_be_built(register Name target);
444 extern  char            *time_to_string(const timestruc_t &time);
445 extern  void            update_target(Property line, Doname result);
446 extern  void            warning(char *, ...);
447 extern  void            write_state_file(int report_recursive, Boolean exiting);
448 extern  Name            vpath_translation(register Name cmd);

450 #define DEPINFO_FMT_VERSION "VERS2$"
451 #define VER_LEN strlen(DEPINFO_FMT_VERSION)

453 #ifdef NSE

455 /*
456  *  NSE version for depinfo format
457  */
458 extern Boolean          nse;
459 extern Name             nse_backquote_seen;
460 extern Boolean          nse_did_recursion;
461 extern Name             nse_shell_var_used;
462 extern Boolean          nse_watch_vars;
463 extern wchar_t          current_makefile[MAXPATHLEN];
464 extern Boolean          nse_depinfo_locked;
465 extern char             nse_depinfo_lockfile[MAXPATHLEN];
466 extern Name             derived_src;

468 extern void             depvar_dep_macro_used(Name);
469 extern void             depvar_rule_macro_used(Name);
470 extern Boolean          nse_backquotes(wchar_t *);
471 extern void             nse_check_cd(Property);
472 extern void             nse_check_derived_src(Name, wchar_t *, Cmd_line);
473 extern void             nse_check_file_backquotes(wchar_t *);
474 extern void             nse_check_no_deps_no_rule(Name, Property, Property);
475 extern void             nse_check_sccs(wchar_t *, wchar_t *);
476 extern void             nse_dep_cmdmacro(wchar_t *);
477 extern int              nse_exit_status(void);
478 extern void             nse_init_source_suffixes(void);
479 extern void             nse_no_makefile(Name);
480 extern void             nse_rule_cmdmacro(wchar_t *);
481 extern void             nse_wildcard(wchar_t *, wchar_t *);
482 #endif

484 #endif
```

```
**********************************************************
   23312 Wed May 20 11:27:39 2015
new/usr/src/cmd/make/include/mksh/defs.h
make: unifdef for other OSes (undefined)
**********************************************************
```

```
#ifndef _MKSH_DEFS_H
#define _MKSH_DEFS_H
/*
 * 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.
 */

/*
 * This is not "#ifdef TEAMWARE_MAKE_CMN" because we're currently
 * using the TW fake i18n headers and libraries to build both
 * SMake and PMake on SPARC/S1 and x86/S2.
 */

#include <avo/intl.h>
#include <limits.h>             /* MB_LEN_MAX */
#include <stdio.h>
#include <stdlib.h>             /* wchar_t */
#include <string.h>             /* strcmp() */
#include <nl_types.h>           /* catgets() */
#include <sys/param.h>          /* MAXPATHLEN */
#include <sys/types.h>          /* time_t, caddr_t */
#include <vroot/vroot.h>        /* pathpt */
#include <sys/time.h>           /* timestruc_t */
#include <errno.h>              /* errno */

#if defined (HP_UX) || defined (linux)
#define  MAXNAMELEN          256
#define  RW_NO_OVERLOAD_WCHAR 1  /* Rogue Wave, belongs in <rw/compiler.h> */
#else
#include <wctype.h>
#include <widec.h>
#endif

#if defined (linux)
/*
 * Definition of wchar functions.
 */
#       include <wctype.h>
#       include <wchar.h>
#       define wsdup(x) wcsdup(x)
#       define wschr(x,y) wcschr(x,y)
```



```
#       define wscat(x,y) wcscat(x,y)
#       define wsrchr(x,y) wcsrchr(x,y)
#       define wslen(x) wcslen(x)
#       define wscpy(x,y) wcscpy(x,y)
#       define wsncpy(x,y,z) wcsncpy(x,y,z)
#       define wscmp(x,y) wcscmp(x,y)
#       define wsncmp(x,y,z) wcsncmp(x,y,z)
#endif

/*
 * A type and some utilities for boolean values
 */

#define false   BOOLEAN_false
#define true    BOOLEAN_true

typedef enum {
        false =         0,
        true =          1,
        failed =        0,
        succeeded =     1
} Boolean;
#define BOOLEAN(expr)           ((expr) ? true : false)

/*
 * Some random constants (in an enum so dbx knows their values)
 */
enum {
        update_delay = 30,              /* time between rstat checks */
#ifdef sun386
        ar_member_name_len = 14,
#else
        ar_member_name_len = 1024,
#endif

        hashsize = 2048                 /* size of hash table */
};
```

unchanged_portion_omitted_

```
/*
 * Magic values for the timestamp stored with each name object
 */

#if defined (linux)
typedef struct timespec timestruc_t;
#endif

extern const timestruc_t file_no_time;
extern const timestruc_t file_doesnt_exist;
extern const timestruc_t file_is_dir;
extern const timestruc_t file_min_time;
extern const timestruc_t file_max_time;

/*
 * Each Name has a list of properties
 * The properties are used to store information that only
 * a subset of the Names need
 */
typedef enum {
        no_prop,
        conditional_prop,
        line_prop,
        macro_prop,
        makefile_prop,
        member_prop,
        recursive_prop,
```

```
391         sccs_prop,
392         suffix_prop,
393         target_prop,
394         time_prop,
395         vpath_alias_prop,
396         long_member_name_prop,
397         macro_append_prop,
398         env_mem_prop
399 } Property_id;
```

_unchanged_portion_omitted_

```
415 struct _Macro {
416         /*
417         * For "ABC = xyz" constructs
418         * Name "ABC" get one macro prop
419         */
420         struct _Name            *value;
421 #ifdef NSE
422         Boolean                 imported:1;
423 #endif
424         Boolean                 exported:1;
425         Boolean                 read_only:1;
426         /*
427         * This macro is defined conditionally
428         */
429         Boolean                 is_conditional:1;
430         /*
431         * The list for $? is stored as a structured list that
432         * is translated into a string iff it is referenced.
433         * This is why  some macro values need a daemon.
434         */
464 #if defined(HP_UX) || defined(linux)
465         Daemon                  daemon;
466 #else
435         Daemon                  daemon:2;
468 #endif
436 };
```

_unchanged_portion_omitted_

```
450 struct _Name {
451         struct _Property        *prop;          /* List of properties */
452         char                    *string_mb;     /* Multi-byte name string */
453         struct {
454                 unsigned int            length;
455         }                       hash;
456         struct {
457                 timestruc_t             time;           /* Modification */
458                 int                     stat_errno;     /* error from "stat" */
459                 off_t                   size;           /* Of file */
460                 mode_t                  mode;           /* Of file */
494 #if defined(HP_UX) || defined(linux)
495                 Boolean                 is_file;
496                 Boolean                 is_dir;
497                 Boolean                 is_sym_link;
498                 Boolean                 is_precious;
499                 enum sccs_stat          has_sccs;
500 #else
461                 Boolean                 is_file:1;
462                 Boolean                 is_dir:1;
463                 Boolean                 is_sym_link:1;
464                 Boolean                 is_precious:1;
465 #ifdef NSE
466                 Boolean                 is_derived_src:1;
467 #endif
468                 enum sccs_stat          has_sccs:2;
509 #endif
```



```
469         }                       stat;
470         /*
471          * Count instances of :: definitions for this target
472          */
473         short                   colon_splits;
474         /*
475          * We only clear the automatic depes once per target per report
476          */
477         short                   temp_file_number;
478         /*
479          * Count how many conditional macros this target has defined
480          */
481         short                   conditional_cnt;
482         /*
483          * A conditional macro was used when building this target
484          */
485         Boolean                 depends_on_conditional:1;
486         /*
487          * Pointer to list of conditional macros which were used to build
488          * this target
489          */
490         struct _Macro_list      *conditional_macro_list;
491         Boolean                 has_member_depe:1;
492         Boolean                 is_member:1;
493         /*
494          * This target is a directory that has been read
495          */
496         Boolean                 has_read_dir:1;
497         /*
498          * This name is a macro that is now being expanded
499          */
500         Boolean                 being_expanded:1;
501         /*
502          * This name is a magic name that the reader must know about
503          */
545 #if defined(HP_UX) || defined(linux)
546         Special                 special_reader;
547         Doname                  state;
548         Separator               colons;
549 #else
504         Special                 special_reader:5;
505         Doname                  state:3;
506         Separator               colons:3;
553 #endif
507         Boolean                 has_depe_list_expanded:1;
508         Boolean                 suffix_scan_done:1;
509         Boolean                 has_complained:1;       /* For sccs */
510         /*
511          * This target has been built during this make run
512          */
513         Boolean                 ran_command:1;
514         Boolean                 with_squiggle:1;        /* for .SUFFIXES */
515         Boolean                 without_squiggle:1;     /* for .SUFFIXES */
516         Boolean                 has_read_suffixes:1;    /* Suffix list cached*/
517         Boolean                 has_suffixes:1;
518         Boolean                 has_target_prop:1;
519         Boolean                 has_vpath_alias_prop:1;
520         Boolean                 dependency_printed:1;   /* For dump_make_state()
521         Boolean                 dollar:1;               /* In namestring */
522         Boolean                 meta:1;                 /* In namestring */
523         Boolean                 percent:1;              /* In namestring */
524         Boolean                 wildcard:1;             /* In namestring */
525         Boolean                 has_parent:1;
526         Boolean                 is_target:1;
527         Boolean                 has_built:1;
528         Boolean                 colon:1;                /* In namestring */
```

```
529         Boolean                 parenleft:1;            /* In namestring */
530         Boolean                 has_recursive_dependency:1;
531         Boolean                 has_regular_dependency:1;
532         Boolean                 is_double_colon:1;
533         Boolean                 is_double_colon_parent:1;
534         Boolean                 has_long_member_name:1;
535         /*
536          * allowed to run in parallel
537          */
538         Boolean                 parallel:1;
539         /*
540          * not allowed to run in parallel
541          */
542         Boolean                 no_parallel:1;
543         /*
544          * used in dependency_conflict
545          */
546         Boolean                 checking_subtree:1;
547         Boolean                 added_pattern_conditionals:1;
548         /*
549          * rechecking target for possible rebuild
550          */
551         Boolean                 rechecking_target:1;
552         /*
553          * build this target in silent mode
554          */
555         Boolean                 silent_mode:1;
556         /*
557          * build this target in ignore error mode
558          */
559         Boolean                 ignore_error_mode:1;
560         Boolean                 dont_activate_cond_values:1;
561         /*
562          * allowed to run serially on local host
563          */
564         Boolean                 localhost:1;
565 };
______unchanged_portion_omitted_

718 #define PROPERTY_HEAD_SIZE (sizeof (struct _Property)-sizeof (union Body))
719 struct _Property {
720         struct _Property        *next;
768 #if defined(HP_UX) || defined(linux)
769         Property_id             type;
770 #else
721         Property_id             type:4;
772 #endif
722         union Body              body;
723 };
______unchanged_portion_omitted_
```

```
**********************************************************
   20901 Wed May 20 11:27:39 2015
new/usr/src/cmd/make/lib/mksh/dosys.cc
make: unifdef for other OSes (undefined)
**********************************************************
```

```
/*
 * 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 2005 Sun Microsystems, Inc. All rights reserved.
 * Use is subject to license terms.
 */


/*
 *      dosys.cc
 *
 *      Execute one commandline
 */

/*
 * Included files
 */
#include <sys/wait.h>                   /* WIFEXITED(status) */
#include <avo/avo_alloca.h>             /* alloca() */

#if defined(TEAMWARE_MAKE_CMN) || defined(MAKETOOL) /* tolik */
#       include <avo/strings.h> /* AVO_STRDUP() */
#if defined(DISTRIBUTED)
#       include <dm/Avo_CmdOutput.h>
#       include <rw/xdrstrea.h>
#endif
#endif

#include <stdio.h>              /* errno */
#include <errno.h>              /* errno */
#include <fcntl.h>              /* open() */
#include <mksh/dosys.h>
#include <mksh/macro.h>         /* getvar() */
#include <mksh/misc.h>          /* getmem(), fatal_mksh(), errmsg() */
#include <mksdmsi18n/mksdmsi18n.h>      /* libmksdmsi18n_init() */
#include <sys/signal.h>         /* SIG_DFL */
#include <sys/stat.h>           /* open() */
#include <sys/wait.h>           /* wait() */
#include <ulimit.h>             /* ulimit() */
#include <unistd.h>             /* close(), dup2() */

#if defined (HP_UX) || defined (linux)
#       include <sys/param.h>
```



```
#       include <wctype.h>
#       include <wchar.h>
#endif

#if defined (linux)
#       define wslen(x) wcslen(x)
#       define wscpy(x,y) wcscpy(x,y)
#endif

/*
 * Defined macros
 */
#if defined(DISTRIBUTED) || defined(MAKETOOL) /* tolik */
#define SEND_MTOOL_MSG(cmds) \
        if (send_mtool_msgs) { \
                cmds \
        }
#else
#define SEND_MTOOL_MSG(cmds)
#endif

/*
 * typedefs & structs
 */

/*
 * Static variables
 */

/*
 * File table of contents
 */
static Boolean  exec_vp(register char *name, register char **argv, char **envp,

/*
 * Workaround for NFS bug. Sometimes, when running 'open' on a remote
 * dmake server, it fails with "Stale NFS file handle" error.
 * The second attempt seems to work.
 */
int
my_open(const char *path, int oflag, mode_t mode) {
        int res = open(path, oflag, mode);
#ifdef linux
// Workaround for NFS problem: even when all directories in 'path'
// exist, 'open' (file creation) fails with ENOENT.
        int nattempt = 0;
        while (res < 0 && (errno == ESTALE || errno == EAGAIN || errno == ENOENT
                nattempt++;
                if(nattempt > 30) {
                        break;
                }
                sleep(1);
#else
        if (res < 0 && (errno == ESTALE || errno == EAGAIN)) {
#endif
                /* Stale NFS file handle. Try again */
                res = open(path, oflag, mode);
        }
        return res;
}

/*
 *      void
 *      redirect_io(char *stdout_file, char *stderr_file)
 *
 *      Redirects stdout and stderr for a child mksh process.
```

```
 107  */
 108 void
 109 redirect_io(char *stdout_file, char *stderr_file)
 110 {
 111         long            descriptor_limit;
 112         int             i;

 135 #if defined (HP_UX) || defined (linux)
 136         /*
 137          *  HP-UX does not support the UL_GDESLIM command for ulimit().
 138          *  NOFILE == max num open files per process (from <sys/param.h>)
 139          */
 140         descriptor_limit = NOFILE;
 141 #else
 114         if ((descriptor_limit = ulimit(UL_GDESLIM)) < 0) {
 115                 fatal_mksh(catgets(libmksdmsi18n_catd, 1, 89, "ulimit() failed:
 116         }
 145 #endif
 117         for (i = 3; i < descriptor_limit; i++) {
 118                 (void) close(i);
 119         }
 120         if ((i = my_open(stdout_file,
 121                  O_WRONLY | O_CREAT | O_TRUNC | O_DSYNC,
 122                  S_IREAD | S_IWRITE)) < 0) {
 123                 fatal_mksh(catgets(libmksdmsi18n_catd, 1, 90, "Couldn't open sta
 124                       stdout_file,
 125                       errmsg(errno));
 126         } else {
 127                 if (dup2(i, 1) == -1) {
 128                         fatal_mksh(NOCATGETS("*** Error: dup2(3, 1) failed: %s")
 129                                 errmsg(errno));
 130                 }
 131                 close(i);
 132         }
 133         if (stderr_file == NULL) {
 134                 if (dup2(1, 2) == -1) {
 135                         fatal_mksh(NOCATGETS("*** Error: dup2(1, 2) failed: %s")
 136                                 errmsg(errno));
 137                 }
 138         } else if ((i = my_open(stderr_file,
 139                       O_WRONLY | O_CREAT | O_TRUNC | O_DSYNC,
 140                       S_IREAD | S_IWRITE)) < 0) {
 141                 fatal_mksh(catgets(libmksdmsi18n_catd, 1, 91, "Couldn't open sta
 142                       stderr_file,
 143                       errmsg(errno));
 144         } else {
 145                 if (dup2(i, 2) == -1) {
 146                         fatal_mksh(NOCATGETS("*** Error: dup2(3, 2) failed: %s")
 147                                 errmsg(errno));
 148                 }
 149                 close(i);
 150         }
 151 }
```

unchanged_portion_omitted_

```
 268 /*
 269  *      doshell(command, ignore_error)
 270  *
 271  *      Used to run command lines that include shell meta-characters.
 272  *      The make macro SHELL is supposed to contain a path to the shell.
 273  *
 274  *      Return value:
 275  *                              The pid of the process we started
 276  *
 277  *      Parameters:
 278  *              command         The command to run
```



```
 279  *              ignore_error    Should we abort on error?
 280  *
 281  *      Global variables used:
 282  *              filter_stderr   If -X is on we redirect stderr
 283  *              shell_name      The Name "SHELL", used to get the path to shell
 284  */
 285 int
 286 doshell(wchar_t *command, register Boolean ignore_error, Boolean redirect_out_er
 287 {
 288         char                    *argv[6];
 289         int                     argv_index = 0;
 290         int                     cmd_argv_index;
 291         int                     length;
 292         char                    nice_prio_buf[MAXPATHLEN];
 293         register Name           shell = getvar(shell_name);
 294         register char           *shellname;
 295         char                    *tmp_mbs_buffer;


 298         if (IS_EQUAL(shell->string_mb, "")) {
 299                 shell = shell_name;
 300         }
 301         if ((shellname = strrchr(shell->string_mb, (int) slash_char)) == NULL) {
 302                 shellname = shell->string_mb;
 303         } else {
 304                 shellname++;
 305         }

 307         /*
 308          * Only prepend the /usr/bin/nice command to the original command
 309          * if the nice priority, nice_prio, is NOT zero (0).
 310          * Nice priorities can be a positive or a negative number.
 311          */
 312         if (nice_prio != 0) {
 313                 argv[argv_index++] = (char *)NOCATGETS("nice");
 314                 (void) sprintf(nice_prio_buf, NOCATGETS("-%d"), nice_prio);
 315                 argv[argv_index++] = strdup(nice_prio_buf);
 316         }
 317         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
 318         argv[argv_index++] = (char*)(ignore_error ? NOCATGETS("-c") : NOCATGETS(
 355 #endif
 319         if ((length = wslen(command)) >= MAXPATHLEN) {
 320                 tmp_mbs_buffer = getmem((length * MB_LEN_MAX) + 1);
 321                 (void) wcstombs(tmp_mbs_buffer, command, (length * MB_LEN_MAX) +
 322                 cmd_argv_index = argv_index;
 323                 argv[argv_index++] = strdup(tmp_mbs_buffer);
 324                 retmem_mb(tmp_mbs_buffer);
 325         } else {
 326                 WCSTOMBS(mbs_buffer, command);
 327                 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
```

```
328                 argv[argv_index++] = strdup(mbs_buffer);
329         }
330         argv[argv_index] = NULL;
331         (void) fflush(stdout);
332         if ((childPid = fork()) == 0) {
333                 enable_interrupt((void (*) (int)) SIG_DFL);
334                 if (redirect_out_err) {
335                         redirect_io(stdout_file, stderr_file);
336                 }
337 #if 0
338                 if (filter_stderr) {
339                         redirect_stderr();
340                 }
341 #endif
342                 if (nice_prio != 0) {
343                         (void) execve(NOCATGETS("/usr/bin/nice"), argv, environ)
344                         fatal_mksh(catgets(libmksdmsi18n_catd, 1, 92, "Could not
345                               errmsg(errno));
346                 } else {
347                         (void) execve(shell->string_mb, argv, environ);
348                         fatal_mksh(catgets(libmksdmsi18n_catd, 1, 93, "Could not
349                               shell->string_mb,
350                               errmsg(errno));
351                 }
352         }
353         if (childPid  == -1) {
354                 fatal_mksh(catgets(libmksdmsi18n_catd, 1, 94, "fork failed: %s")
355                       errmsg(errno));
356         }
357         retmem_mb(argv[cmd_argv_index]);
358         return childPid;
359 }
```

unchanged_portion_omitted_

```
**********************************************************
    2181 Wed May 20 11:27:40 2015
new/usr/src/cmd/make/lib/mksh/i18n.cc
make: unifdef for other OSes (undefined)
**********************************************************
```

```
/*
 * 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 2003 Sun Microsystems, Inc. All rights reserved.
 * Use is subject to license terms.
 */


/*
 *      i18n.cc
 *
 *      Deal with internationalization conversions
 */

/*
 * Included files
 */
#include <mksh/i18n.h>
#include <mksh/misc.h>          /* setup_char_semantics() */
#if defined (linux)
#       include <wctype.h>
#       include <wchar.h>
#       define wschr(x,y) wcschr(x,y)
#endif

/*
 *      get_char_semantics_value(ch)
 *
 *      Return value:
 *              The character semantics of ch.
 *
 *      Parameters:
 *              ch              character we want semantics for.
 *
 */
char
get_char_semantics_value(wchar_t ch)
{
        static Boolean  char_semantics_setup;

        if (!char_semantics_setup) {
                setup_char_semantics();
                char_semantics_setup = true;
```



```
        }
        return char_semantics[get_char_semantics_entry(ch)];
}
```

_____*unchanged_portion_omitted_*

```
**********************************************************
   38887 Wed May 20 11:27:40 2015
new/usr/src/cmd/make/lib/mksh/macro.cc
make: unifdef for other OSes (undefined)
**********************************************************
______unchanged_portion_omitted_

 900 /*
 901  *      init_arch_macros(void)
 902  *
 903  *      Set the magic macros TARGET_ARCH, HOST_ARCH,
 904  *
 905  *      Parameters:
 906  *
 907  *      Global variables used:
 908  *                              host_arch   Property for magic macro HOST_ARCH
 909  *                              target_arch Property for magic macro TARGET_ARCH
 910  *
 911  *      Return value:
 912  *                              The function does not return a value, but can
 913  *                              call fatal() in case of error.
 914  */
 915 static void
 916 init_arch_macros(void)
 917 {
 918         String_rec      result_string;
 919         wchar_t         wc_buf[STRING_BUFFER_LENGTH];
 920         char            mb_buf[STRING_BUFFER_LENGTH];
 921         FILE            *pipe;
 922         Name            value;
 923         int             set_host, set_target;
 924 #ifdef NSE
 925         Property        macro;
 926 #endif
 927 #if defined(linux)
 928         const char      *mach_command = NOCATGETS("/bin/uname -p");
 929 #else
 927         const char      *mach_command = NOCATGETS("/bin/mach");
 931 #endif

 929         set_host = (get_prop(host_arch->prop, macro_prop) == NULL);
 930         set_target = (get_prop(target_arch->prop, macro_prop) == NULL);

 932         if (set_host || set_target) {
 933                 INIT_STRING_FROM_STACK(result_string, wc_buf);
 934                 append_char((int) hyphen_char, &result_string);

 936                 if ((pipe = popen(mach_command, "r")) == NULL) {
 937                         fatal_mksh(catgets(libmksdmsi18n_catd, 1, 185, "Execute
 938                 }
 939                 while (fgets(mb_buf, sizeof(mb_buf), pipe) != NULL) {
 940                         MBSTOWCS(wcs_buffer, mb_buf);
 941                         append_string(wcs_buffer, &result_string, wslen(wcs_buff
 942                 }
 943                 if (pclose(pipe) != 0) {
 944                         fatal_mksh(catgets(libmksdmsi18n_catd, 1, 186, "Execute
 945                 }

 947                 value = GETNAME(result_string.buffer.start, wslen(result_string.

 949 #ifdef NSE
 950                 macro = setvar_daemon(host_arch, value, false, no_daemon, true,
 951                 macro->body.macro.imported= true;
 952                 macro = setvar_daemon(target_arch, value, false, no_daemon, true
 953                 macro->body.macro.imported= true;
 954 #else
```



```
 955                 if (set_host) {
 956                         (void) setvar_daemon(host_arch, value, false, no_daemon,
 957                 }
 958                 if (set_target) {
 959                         (void) setvar_daemon(target_arch, value, false, no_daemo
 960                 }
 961 #endif
 962         }
 963 }
______unchanged_portion_omitted_

1094 /*
1095  * We use a permanent buffer to reset SUNPRO_DEPENDENCIES value.
1096  */
1097 char    *sunpro_dependencies_buf = NULL;
1098 char    *sunpro_dependencies_oldbuf = NULL;
1099 int     sunpro_dependencies_buf_size = 0;

1101 /*
1102  *      setvar_daemon(name, value, append, daemon, strip_trailing_spaces)
1103  *
1104  *      Set a macro value, possibly supplying a daemon to be used
1105  *      when referencing the value.
1106  *
1107  *      Return value:
1108  *                              The property block with the new value
1109  *
1110  *      Parameters:
1111  *              name            Name of the macro to set
1112  *              value           The value to set
1113  *              append          Should we reset or append to the current value?
1114  *              daemon          Special treatment when reading the value
1115  *              strip_trailing_spaces from the end of value->string
1116  *              debug_level     Indicates how much tracing we should do
1117  *
1118  *      Global variables used:
1119  *              makefile_type   Used to check if we should enforce read only
1120  *              path_name       The Name "PATH", compared against
1121  *              virtual_root    The Name "VIRTUAL_ROOT", compared against
1122  *              vpath_defined   Set if the macro VPATH is set
1123  *              vpath_name      The Name "VPATH", compared against
1124  *              envvar          A list of environment vars with $ in value
1125  */
1126 Property
1127 setvar_daemon(register Name name, register Name value, Boolean append, Daemon da
1128 {
1129         register Property       macro = maybe_append_prop(name, macro_prop);
1130         register Property       macro_apx = get_prop(name->prop, macro_append_pr
1131         int                     length = 0;
1132         String_rec              destination;
1133         wchar_t                 buffer[STRING_BUFFER_LENGTH];
1134         register Chain          chain;
1135         Name                    val;
1136         wchar_t                 *val_string = (wchar_t*)NULL;
1137         Wstring                 wcb;

1139 #ifdef NSE
1140         macro->body.macro.imported = false;
1141 #endif

1143         if ((makefile_type != reading_nothing) &&
1144             macro->body.macro.read_only) {
1145                 return macro;
1146         }
1147         /* Strip spaces from the end of the value */
1148         if (daemon == no_daemon) {
```

```
1149                 if(value != NULL) {
1150                         wcb.init(value);
1151                         length = wcb.length();
1152                         val_string = wcb.get_string();
1153                 }
1154                 if ((length > 0) && iswspace(val_string[length-1])) {
1155                         INIT_STRING_FROM_STACK(destination, buffer);
1156                         buffer[0] = 0;
1157                         append_string(val_string, &destination, length);
1158                         if (strip_trailing_spaces) {
1159                                 while ((length > 0) &&
1160                                        iswspace(destination.buffer.start[length-
1161                                         destination.buffer.start[--length] = 0;
1162                                 }
1163                         }
1164                         value = GETNAME(destination.buffer.start, FIND_LENGTH);
1165                 }
1166         }
1167
1168         if(macro_apx != NULL) {
1169                 val = macro_apx->body.macro_appendix.value;
1170         } else {
1171                 val = macro->body.macro.value;
1172         }

1174         if (append) {
1175                 /*
1176                  * If we are appending, we just tack the new value after
1177                  * the old one with a space in between.
1178                  */
1179                 INIT_STRING_FROM_STACK(destination, buffer);
1180                 buffer[0] = 0;
1181                 if ((macro != NULL) && (val != NULL)) {
1182                         APPEND_NAME(val,
1183                                     &destination,
1184                                     (int) val->hash.length);
1185                         if (value != NULL) {
1186                                 wcb.init(value);
1187                                 if(wcb.length() > 0) {
1188                                         MBTOWC(wcs_buffer, " ");
1189                                         append_char(wcs_buffer[0], &destination)
1190                                 }
1191                         }
1192                 }
1193                 if (value != NULL) {
1194                         APPEND_NAME(value,
1195                                     &destination,
1196                                     (int) value->hash.length);
1197                 }
1198                 value = GETNAME(destination.buffer.start, FIND_LENGTH);
1199                 wcb.init(value);
1200                 if (destination.free_after_use) {
1201                         retmem(destination.buffer.start);
1202                 }
1203         }

1205         /* Debugging trace */
1206         if (debug_level > 1) {
1207                 if (value != NULL) {
1208                         switch (daemon) {
1209                         case chain_daemon:
1210                                 (void) printf("%s =", name->string_mb);
1211                                 for (chain = (Chain) value;
1212                                      chain != NULL;
1213                                      chain = chain->next) {
1214                                         (void) printf(" %s", chain->name->string
```



```
1215                                 }
1216                                 (void) printf("\n");
1217                                 break;
1218                         case no_daemon:
1219                                 (void) printf("%s= %s\n",
1220                                               name->string_mb,
1221                                               value->string_mb);
1222                                 break;
1223                         }
1224                 } else {
1225                         (void) printf("%s =\n", name->string_mb);
1226                 }
1227         }
1228         /* Set the new values in the macro property block */
1229 /**/
1230         if(macro_apx != NULL) {
1231                 macro_apx->body.macro_appendix.value = value;
1232                 INIT_STRING_FROM_STACK(destination, buffer);
1233                 buffer[0] = 0;
1234                 if (value != NULL) {
1235                         APPEND_NAME(value,
1236                                     &destination,
1237                                     (int) value->hash.length);
1238                         if (macro_apx->body.macro_appendix.value_to_append != NU
1239                                 MBTOWC(wcs_buffer, " ");
1240                                 append_char(wcs_buffer[0], &destination);
1241                         }
1242                 }
1243                 if (macro_apx->body.macro_appendix.value_to_append != NULL) {
1244                         APPEND_NAME(macro_apx->body.macro_appendix.value_to_appe
1245                                     &destination,
1246                                     (int) macro_apx->body.macro_appendix.value
1247                 }
1248                 value = GETNAME(destination.buffer.start, FIND_LENGTH);
1249                 if (destination.free_after_use) {
1250                         retmem(destination.buffer.start);
1251                 }
1252         }
1253 /**/
1254         macro->body.macro.value = value;
1255         macro->body.macro.daemon = daemon;
1256         /*
1257          * If the user changes the VIRTUAL_ROOT, we need to flush
1258          * the vroot package cache.
1259          */
1260         if (name == path_name) {
1261                 flush_path_cache();
1262         }
1263         if (name == virtual_root) {
1264                 flush_vroot_cache();
1265         }
1266         /* If this sets the VPATH we remember that */
1267         if ((name == vpath_name) &&
1268             (value != NULL) &&
1269             (value->hash.length > 0)) {
1270                 vpath_defined = true;
1271         }
1272         /*
1273          * For environment variables we also set the
1274          * environment value each time.
1275          */
1276         if (macro->body.macro.exported) {
1277                 static char     *env;

1279 #ifdef DISTRIBUTED
1280                 if (!reading_environment && (value != NULL)) {
```

```
#else
                if (!reading_environment && (value != NULL) && value->dollar) {
#endif
                        Envvar  p;

                        for (p = envvar; p != NULL; p = p->next) {
                                if (p->name == name) {
                                        p->value = value;
                                        p->already_put = false;
                                        goto found_it;
                                }
                        }
                        p = ALLOC(Envvar);
                        p->name = name;
                        p->value = value;
                        p->next = envvar;
                        p->env_string = NULL;
                        p->already_put = false;
                        envvar = p;
found_it:;
#ifdef DISTRIBUTED
                }
                if (reading_environment || (value == NULL) || !value->dollar) {
#else
                } else {
#endif
                        length = 2 + strlen(name->string_mb);
                        if (value != NULL) {
                                length += strlen(value->string_mb);
                        }
                        Property env_prop = maybe_append_prop(name, env_mem_prop
                        /*
                         * We use a permanent buffer to reset SUNPRO_DEPENDENCIE
                         */
                        if (!strncmp(name->string_mb, NOCATGETS("SUNPRO_DEPENDEN
                                if (length >= sunpro_dependencies_buf_size) {
                                        sunpro_dependencies_buf_size=length*2;
                                        if (sunpro_dependencies_buf_size < 4096)
                                                sunpro_dependencies_buf_size = 4
                                        if (sunpro_dependencies_buf)
                                                sunpro_dependencies_oldbuf = sun
                                        sunpro_dependencies_buf=getmem(sunpro_de
                                }
                                env = sunpro_dependencies_buf;
                        } else {
                                env = getmem(length);
                        }
                        env_alloc_num++;
                        env_alloc_bytes += length;
                        (void) sprintf(env,
                                       "%s=%s",
                                       name->string_mb,
                                       value == NULL ?
                                         "" : value->string_mb);
                        (void) putenv(env);
                        env_prop->body.env_mem.value = env;
                        if (sunpro_dependencies_oldbuf) {
                                /* Return old buffer */
                                retmem_mb(sunpro_dependencies_oldbuf);
                                sunpro_dependencies_oldbuf = NULL;
                        }
                }
        }
        if (name == target_arch) {
                Name            ha = getvar(host_arch);
                Name            ta = getvar(target_arch);
```



```
                Name            vr = getvar(virtual_root);
                int             length;
                wchar_t         *new_value;
                wchar_t         *old_vr;
                Boolean         new_value_allocated = false;

                Wstring         ha_str(ha);
                Wstring         ta_str(ta);
                Wstring         vr_str(vr);

                wchar_t * wcb_ha = ha_str.get_string();
                wchar_t * wcb_ta = ta_str.get_string();
                wchar_t * wcb_vr = vr_str.get_string();

                length = 32 +
                  wslen(wcb_ha) +
                    wslen(wcb_ta) +
                      wslen(wcb_vr);
                old_vr = wcb_vr;
                MBSTOWCS(wcs_buffer, NOCATGETS("/usr/arch/"));
                if (IS_WEQUALN(old_vr,
                               wcs_buffer,
                               wslen(wcs_buffer))) {
                        old_vr = (wchar_t *) wschr(old_vr, (int) colon_char) + 1
                }
                if ( (ha == ta) || (wslen(wcb_ta) == 0) ) {
                        new_value = old_vr;
                } else {
                        new_value = ALLOC_WC(length);
                        new_value_allocated = true;
                        WCSTOMBS(mbs_buffer, old_vr);
#if !defined(linux)
                        (void) wsprintf(new_value,
                                        NOCATGETS("/usr/arch/%s/%s:%s"),
                                        ha->string_mb + 1,
                                        ta->string_mb + 1,
                                        mbs_buffer);
#else
                        char * mbs_new_value = (char *)getmem(length);
                        (void) sprintf(mbs_new_value,
                                        NOCATGETS("/usr/arch/%s/%s:%s"),
                                        ha->string_mb + 1,
                                        ta->string_mb + 1,
                                        mbs_buffer);
                        MBSTOWCS(new_value, mbs_new_value);
                        retmem_mb(mbs_new_value);
#endif
                }
                if (new_value[0] != 0) {
                        (void) setvar_daemon(virtual_root,
                                             GETNAME(new_value, FIND_LENGTH),
                                             false,
                                             no_daemon,
                                             true,
                                             debug_level);
                }
                if (new_value_allocated) {
                        retmem(new_value);
                }
        }
        return macro;
}
```

_____unchanged_portion_omitted_

```
**********************************************************
   24492 Wed May 20 11:27:41 2015
new/usr/src/cmd/make/lib/mksh/misc.cc
make: unifdef for other OSes (undefined)
**********************************************************
```

```
/*
 * 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() */

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

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

/*
 * Defined macros
 */
```



```
/*
 * typedefs & structs
 */

/*
 * Static variables
 */
extern "C" {
        void            (*sigivalue)(int) = SIG_DFL;
        void            (*sigqvalue)(int) = SIG_DFL;
        void            (*sigtvalue)(int) = SIG_DFL;
        void            (*sighvalue)(int) = SIG_DFL;
}
______unchanged_portion_omitted_

/*
 *      errmsg(errnum)
 *
 *      Return the error message for a system call error
 *
 *      Return value:
 *                              An error message string
 *
 *      Parameters:
 *              errnum          The number of the error we want to describe
 *
 *      Global variables used:
 *              sys_errlist     A vector of error messages
 *              sys_nerr        The size of sys_errlist
 */
char *
errmsg(int errnum)
{
#ifdef linux
        return strerror(errnum);
#else // linux

        extern int              sys_nerr;
#ifdef SUN4_x
        extern char             *sys_errlist[];
#endif
        char                    *errbuf;

        if ((errnum < 0) || (errnum > sys_nerr)) {
                errbuf = getmem(6+1+11+1);
                (void) sprintf(errbuf, catgets(libmksdmsi18n_catd, 1, 127, "Erro
                return errbuf;
        } else {
#ifdef SUN4_x
                return(sys_errlist[errnum]);
#endif
                return strerror(errnum);

        }
#endif // linux
}
______unchanged_portion_omitted_
```

```
**********************************************************
    7523 Wed May 20 11:27:41 2015
new/usr/src/cmd/make/lib/mksh/mksh.cc
make: unifdef for other OSes (undefined)
**********************************************************
```

```
/*
 * 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.
 */


/*
 *      mksh.cc
 *
 *      Execute the command(s) of one Make or DMake rule
 */

/*
 * Included files
 */
#if defined(TEAMWARE_MAKE_CMN) || defined(MAKETOOL) /* tolik */
#       include <avo/util.h>
#endif

#include <mksh/dosys.h>         /* redirect_io() */
#include <mksh/misc.h>          /* retmem() */
#include <mksh/mksh.h>
#include <mksdmsi18n/mksdmsi18n.h>
#include <errno.h>
#include <signal.h>

#ifdef HP_UX
        extern void (*sigset(int, void (*)(__harg)))(__harg);
#endif

/*
 * Workaround for NFS bug. Sometimes, when running 'chdir' on a remote
 * dmake server, it fails with "Stale NFS file handle" error.
 * The second attempt seems to work.
 */
int
my_chdir(char * dir) {
        int res = chdir(dir);
        if (res != 0 && (errno == ESTALE || errno == EAGAIN)) {
                /* Stale NFS file handle. Try again */
                res = chdir(dir);
```



```
        }
        return res;
}
```

_____*unchanged_portion_omitted_*

```
**********************************************************
    5259 Wed May 20 11:27:42 2015
new/usr/src/cmd/make/lib/vroot/lock.cc
make: unifdef for other OSes (undefined)
**********************************************************
```

```
/*
 * 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 <avo/intl.h>   /* for NOCATGETS */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/errno.h>
#include <sys/param.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>
#include <vroot/vroot.h>
#include <mksdmsi18n/mksdmsi18n.h>
#include <signal.h>
#include <errno.h>                      /* errno */

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

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

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

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

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



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

int
file_lock(char *name, char *lockname, int *file_locked, int timeout)
{
        int             counter = 0;
        static char     msg[MAXPATHLEN+1];
        int             printed_warning = 0;
        int             r;
        struct stat     statb;
        sigset_t newset;
        sigset_t oldset;

        *file_locked = 0;
        if (timeout <= 0) {
                timeout = 120;
        }
        for (;;) {
                BLOCK_INTERUPTS;
                r = symlink(name, lockname);
                if (r == 0) {
                        *file_locked = 1;
                        UNBLOCK_INTERUPTS;
                        return 0; /* success */
                }
                UNBLOCK_INTERUPTS;

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

                counter = 0;
                for (;;) {
                        sleep(1);
                        r = lstat(lockname, &statb);
                        if (r == -1) {
                                /*
                                 * The lock must have just gone away - try
                                 * again.
```

```
126                                      */
127                                     break;
128                             }

130                             if ((counter > 5) && (!printed_warning)) {
131                                     /* Print waiting message after 5 secs */
132                                     (void) getcwd(msg, MAXPATHLEN);
133                                     fprintf(stderr,
134                                             catgets(libmksdmsi18n_catd, 1, 162, "fil
135                                             name);
136                                     fprintf(stderr,
137                                             catgets(libmksdmsi18n_catd, 1, 163, "fil
138                                             lockname);
139                                     fprintf(stderr,
140                                             catgets(libmksdmsi18n_catd, 1, 144, "Cur
141                                             msg);

143                                     printed_warning = 1;
144                             }

146                             if (++counter > timeout ) {
147                                     /*
148                                      * Waited enough - return an error..
149                                      */
150                                     return EEXIST;
151                             }
152                     }
153             }
154             /* NOTREACHED */
155     }

157     /*
158      * Format a message telling why the lock could not be created.
159      */
160     static  void
161     file_lock_error(char *msg, char *file, char *str, int arg1, int arg2)
162     {
163             int             len;

165             sprintf(msg, catgets(libmksdmsi18n_catd, 1, 145, "Could not lock file ‘%
166             len = strlen(msg);
167             sprintf(&msg[len], str, arg1, arg2);
168             strcat(msg, catgets(libmksdmsi18n_catd, 1, 146, " failed - "));
171     #if !defined(linux)
169             if (errno < sys_nerr) {
173     #ifdef SUN4_x
174                     strcat(msg, sys_errlist[errno]);
175     #endif
170                     strcat(msg, strerror(errno));
171             } else {
172                     len = strlen(msg);
173                     sprintf(&msg[len], NOCATGETS("errno %d"), errno);
174             }
181     #else
182             strcat(msg, strerror(errno));
183     #endif
175     }
```

_____***unchanged_portion_omitted_***

****
1492 Wed May 20 11:27:42 2015
new/usr/src/cmd/make/lib/vroot/mount.cc
make: unifdef for other OSes (undefined)
****

```
   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 1995 Sun Microsystems, Inc. All rights reserved.
  23  * Use is subject to license terms.
  24  */


  27 #include <sys/types.h>
  28 #include <sys/mount.h>

  30 #ifndef HP_UX
  30 extern int mount(const char *spec, const char *dir, int mflag, ...);
  32 #endif

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

  35 static int      mount_thunk(char *path)
  36 {
  37         vroot_result= mount(path, vroot_args.mount.name, vroot_args.mount.mode);
  38         return(vroot_result == 0);
  39 }
```

_____***unchanged_portion_omitted_***