```
**********************************************************
   23286 Wed May 20 11:51:04 2015
new/usr/src/cmd/make/bin/ar.cc
make: avo_alloca.h works around a very old Solaris bug, delete it
**********************************************************
```

```
/*
 * 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.
 */

/*
 *      ar.c
 *
 *      Deal with the lib.a(member.o) and lib.a((entry-point)) notations
 *
 * Look inside archives for notations a(b) and a((b))
 *      a(b)    is file member   b  in archive a
 *      a((b))  is entry point   b  in object archive a
 *
 * For 6.0, create a make which can understand all archive
 * formats.  This is kind of tricky, and <ar.h> isnt any help.
 */

/*
 * Included files
 */
#include <alloca.h>             /* alloca() */
#include <avo/avo_alloca.h>             /* alloca() */
#include <ar.h>
#include <errno.h>              /* errno */
#include <fcntl.h>              /* open() */
#include <mk/defs.h>
#include <mksh/misc.h>          /* retmem_mb() */

struct ranlib {
        union {
                off_t   ran_strx;       /* string table index of */
                char    *ran_name;      /* symbol defined by */
        }       ran_un;
        off_t   ran_off;                /* library member at this offset */
};
```

*_____unchanged_portion_omitted_*

```
**********************************************************
  104099 Wed May 20 11:51:05 2015
new/usr/src/cmd/make/bin/doname.cc
make: avo_alloca.h works around a very old Solaris bug, delete it
**********************************************************
```

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

/*
 *      doname.c
 *
 *      Figure out which targets are out of date and rebuild them
 */

/*
 * Included files
 */
#include <alloca.h>             /* alloca() */
#include <avo/avo_alloca.h>             /* alloca() */
#if defined(TEAMWARE_MAKE_CMN)
#include <avo/util.h>           /* avo_get_user(), avo_hostname() */
#endif

#if defined(DISTRIBUTED) || defined(MAKETOOL) /* tolik */
#       include <avo/strings.h> /* AVO_STRDUP() */
#       include <dm/Avo_MToolJobResultMsg.h>
#       include <dm/Avo_MToolJobStartMsg.h>
#       include <dm/Avo_MToolRsrcInfoMsg.h>
#       include <dm/Avo_macro_defs.h> /* AVO_BLOCK_INTERUPTS & AVO_UNBLOCK_INTER
#       include <dmthread/Avo_ServerState.h>
#       include <rw/pstream.h>
#       include <rw/xdrstrea.h>
#endif

#include <fcntl.h>
#include <mk/defs.h>
#include <mksh/i18n.h>          /* get_char_semantics_value() */
#include <mksh/macro.h>         /* getvar(), expand_value() */
#include <mksh/misc.h>          /* getmem() */
#include <poll.h>


#include <signal.h>
```



```
#       include <stropts.h>

#include <sys/errno.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <sys/utsname.h>        /* uname() */
#include <sys/wait.h>
#include <unistd.h>             /* close() */

/*
 * Defined macros
 */
#       define LOCALHOST "localhost"

#define MAXRULES 100

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

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

/*
 * typedefs & structs
 */

/*
 * Static variables
 */
static char     hostName[MAXNAMELEN] = "";
static char     userName[MAXNAMELEN] = "";

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

static int      second_pass = 0;

/*
 * File table of contents
 */
extern  Doname          doname_check(register Name target, register Boolean do_g
extern  Doname          doname(register Name target, register Boolean do_get, re
static  Boolean         check_dependencies(Doname *result, Property line, Boolea
void            dynamic_dependencies(Name target);
static  Doname          run_command(register Property line, Boolean print_machin
extern  Doname          execute_serial(Property line);
extern  Name            vpath_translation(register Name cmd);
extern  void            check_state(Name temp_file_name);
static  void            read_dependency_file(register Name filename);
static  void            check_read_state_file(void);
static  void            do_assign(register Name line, register Name target);
static  void            build_command_strings(Name target, register Property lin
static  Doname          touch_command(register Property line, register Name targ
extern  void            update_target(Property line, Doname result);
static  Doname          sccs_get(register Name target, register Property *comman
extern  void            read_directory_of_file(register Name file);
static  void            add_pattern_conditionals(register Name target);
```

```
127 extern  void            set_locals(register Name target, register Property old_l
128 extern  void            reset_locals(register Name target, register Property old
129 extern  Boolean         check_auto_dependencies(Name target, int auto_count, Nam
130 static  void            delete_query_chain(Chain ch);

132 // From read2.cc
133 extern  Name            normalize_name(register wchar_t *name_string, register i


136 #if defined(DISTRIBUTED) || defined(MAKETOOL) /* tolik */
137         static void             append_job_result_msg(Avo_MToolJobResultMsg *job
138         static int              pollResults(char *outFn, char *errFn, char *host
139         static void             pollResultsAction(char *outFn, char *errFn);
140         static void             rxmGetNextResultsBlock(int fd);
141         static int              us_sleep(unsigned int nusecs);
142         extern "C" void         Avo_PollResultsAction_Sigusr1Handler(int foo);
143 #endif

145 /*
146  * DONE.
147  *
148  *      doname_check(target, do_get, implicit, automatic)
149  *
150  *      Will call doname() and then inspect the return value
151  *
152  *      Return value:
153  *                              Indication if the build failed or not
154  *
155  *      Parameters:
156  *              target          The target to build
157  *              do_get          Passed thru to doname()
158  *              implicit        Passed thru to doname()
159  *              automatic       Are we building a hidden dependency?
160  *
161  *      Global variables used:
162  *              build_failed_seen       Set if -k is on and error occurs
163  *              continue_after_error    Indicates that -k is on
164  *              report_dependencies     No error msg if -P is on
165  */
166 Doname
167 doname_check(register Name target, register Boolean do_get, register Boolean imp
168 {
169         int first_time = 1;
170         (void) fflush(stdout);
171 try_again:
172         switch (doname(target, do_get, implicit, automatic)) {
173         case build_ok:
174                 second_pass = 0;
175                 return build_ok;
176         case build_running:
177                 second_pass = 0;
178                 return build_running;
179         case build_failed:
180                 if (!continue_after_error) {
181                         fatal(catgets(catd, 1, 13, "Target '%s' not remade becau
182                               target->string_mb);
183                 }
184                 build_failed_seen = true;
185                 second_pass = 0;
186                 return build_failed;
187         case build_dont_know:
188                 /*
189                  * If we can't figure out how to build an automatic
190                  * (hidden) dependency, we just ignore it.
191                  * We later declare the target to be out of date just in
192                  * case something changed.
```



```
193                  * Also, don't complain if just reporting the dependencies
194                  * and not building anything.
195                  */
196                 if (automatic || (report_dependencies_level > 0)) {
197                         second_pass = 0;
198                         return build_dont_know;
199                 }
200                 if(first_time) {
201                         first_time = 0;
202                         second_pass = 1;
203                         goto try_again;
204                 }
205                 second_pass = 0;
206                 if (continue_after_error && !svr4) {
207                         warning(catgets(catd, 1, 14, "Don't know how to make tar
208                                 target->string_mb);
209                         build_failed_seen = true;
210                         return build_failed;
211                 }
212                 fatal(catgets(catd, 1, 15, "Don't know how to make target '%s'")
213                 break;
214         }
215 #ifdef lint
216          return build_failed;
217 #endif
218 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   56887 Wed May 20 11:51:06 2015
new/usr/src/cmd/make/bin/read.cc
make: avo_alloca.h works around a very old Solaris bug, delete it
**********************************************************
```

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

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

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


/*
 * typedefs & structs
 */

/*
 * Static variables
 */

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

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



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

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


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



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

        if (max_include_depth++ >= 40) {
                fatal(catgets(catd, 1, 66, "Too many nested include statements")
        }
        if (makefile->body.makefile.contents != NULL) {
                retmem(makefile->body.makefile.contents);
        }
        source->inp_buf =
          source->inp_buf_ptr =
```

```
127             source->inp_buf_end = NULL;
128         source->error_converting = false;
129         makefile->body.makefile.contents = NULL;
130         makefile->body.makefile.size = 0;
131         if ((makefile_name->hash.length != 1) ||
132             (wcb[0] != (int) hyphen_char)) {
133                 if ((makefile->body.makefile.contents == NULL) &&
134                     (doname_it)) {
135                         if (makefile_path == NULL) {
136                                 add_dir_to_path(".",
137                                                 &makefile_path,
138                                                 -1);
139                                 add_dir_to_path(NOCATGETS("/usr/share/lib/make")
140                                                 &makefile_path,
141                                                 -1);
142                                 add_dir_to_path(NOCATGETS("/etc/default"),
143                                                 &makefile_path,
144                                                 -1);
145                         }
146                         save_makefile_type = makefile_type;
147                         makefile_type = reading_nothing;
148                         if (doname(makefile_name, true, false) == build_dont_kno
149                                 /* Try normalized filename */
150                                 string_start=get_wstring(makefile_name->string_m
151                                 for (string_end=string_start+1; *string_end != L
152                                 normalized_makefile_name=normalize_name(string_s
153                                 if ((strcmp(makefile_name->string_mb, normalized
154                                         (doname(normalized_makefile_name, true,
155                                         n = access_vroot(makefile_name->string_m
156                                                  4,
157                                                  chase_path ?
158                                                  makefile_path : NULL,
159                                                  VROOT_DEFAULT);
160                                         if (n == 0) {
161                                                 get_vroot_path((char **) NULL,
162                                                        &path,
163                                                        (char **) NULL);
164                                                 if ((path[0] == (int) period_cha
165                                                     (path[1] == (int) slash_char
166                                                          path += 2;
167                                                 }
168                                                 MBSTOWCS(wcs_buffer, path);
169                                                 makefile_name = GETNAME(wcs_buff
170                                                                 FIND_LENGTH);
171                                         }
172                                 }
173                                 retmem(string_start);
174                                 /*
175                                  * Commented out: retmem_mb(normalized_makefile_
176                                  * We have to return this memory, but it seems t
177                                  * in dmake or in Sun C++ 5.7 compiler (it works
178                                  * is compiled using Sun C++ 5.6).
179                                  */
180                                 // retmem_mb(normalized_makefile_name->string_mb
181                         }
182                         makefile_type = save_makefile_type;
183                 }
184                 source->string.free_after_use = false;
185                 source->previous = NULL;
186                 source->already_expanded = false;
187                 /* Lock the file for read, but not when -n. */
188                 if (lock_makefile &&
189                     !do_not_exec_rule) {

191                          make_state_lockfile = getmem(strlen(make_state->string_
192                          (void) sprintf(make_state_lockfile,
```



```
193                                                 NOCATGETS("%s.lock"),
194                                                 make_state->string_mb);
195                         (void) file_lock(make_state->string_mb,
196                                          make_state_lockfile,
197                                          (int *) &make_state_locked,
198                                          0);
199                         if(!make_state_locked) {
200                                 printf(NOCATGETS("-- NO LOCKING for read\n"));
201                                 retmem_mb(make_state_lockfile);
202                                 make_state_lockfile = 0;
203                                 return failed;
204                         }
205                 }
206                 if (makefile->body.makefile.contents == NULL) {
207                         save_makefile_type = makefile_type;
208                         makefile_type = reading_nothing;
209                         if ((doname_it) &&
210                             (doname(makefile_name, true, false) == build_failed)
211                                 if (complain) {
212                                         (void) fprintf(stderr,
213 #ifdef DISTRIBUTED
214                                                 catgets(catd, 1, 67, "dma
215 #else
216                                                 catgets(catd, 1, 237, "ma
217 #endif
218                                                 makefile_name->string_mb)
219                                 }
220                                 max_include_depth--;
221                                 makefile_type = save_makefile_type;
222                                 return failed;
223                         }
224                         makefile_type = save_makefile_type;
225                         //
226                         // Before calling exists() make sure that we have the ri
227                         //
228                         makefile_name->stat.time = file_no_time;

230                         if (exists(makefile_name) == file_doesnt_exist) {
231                                 if (complain ||
232                                     (makefile_name->stat.stat_errno != ENOENT))
233                                         if (must_exist) {
234                                                 fatal(catgets(catd, 1, 68, "Can'
235                                                       makefile_name->string_mb,
236                                                       errmsg(makefile_name->
237                                                              stat.stat_errno));
238                                         } else {
239                                                 warning(catgets(catd, 1, 69, "Ca
240                                                         makefile_name->string_mb
241                                                         errmsg(makefile_name->
242                                                                stat.stat_errno))
243                                         }
244                                 }
245                                 max_include_depth--;
246                                 if(make_state_locked && (make_state_lockfile !=
247                                         (void) unlink(make_state_lockfile);
248                                         retmem_mb(make_state_lockfile);
249                                         make_state_lockfile = NULL;
250                                         make_state_locked = false;
251                                 }
252                                 retmem(wcb);
253                                 retmem_mb((char *)source);
254                                 return failed;
255                         }
256                         /*
257                          * These values are the size and bytes of
258                          * the MULTI-BYTE makefile.
```

```
259                          */
260                         orig_makefile->body.makefile.size =
261                           makefile->body.makefile.size =
262                             source->bytes_left_in_file =
263                               makefile_name->stat.size;
264                         if (report_file) {
265                                 for (dpp = &makefiles_used;
266                                      *dpp != NULL;
267                                      dpp = &(*dpp)->next);
268                                 dp = ALLOC(Dependency);
269                                 dp->next = NULL;
270                                 dp->name = makefile_name;
271                                 dp->automatic = false;
272                                 dp->stale = false;
273                                 dp->built = false;
274                                 *dpp = dp;
275                         }
276                         source->fd = open_vroot(makefile_name->string_mb,
277                                                 O_RDONLY,
278                                                 0,
279                                                 NULL,
280                                                 VROOT_DEFAULT);
281                         if (source->fd < 0) {
282                                 if (complain || (errno != ENOENT)) {
283                                         if (must_exist) {
284                                                 fatal(catgets(catd, 1, 70, "Can'
285                                                       makefile_name->string_mb,
286                                                       errmsg(errno));
287                                         } else {
288                                                 warning(catgets(catd, 1, 71, "Ca
289                                                         makefile_name->string_mb
290                                                         errmsg(errno));
291                                         }
292                                 }
293                                 max_include_depth--;
294                                 return failed;
295                         }
296                         (void) fcntl(source->fd, F_SETFD, 1);
297                         orig_makefile->body.makefile.contents =
298                           makefile->body.makefile.contents =
299                             source->string.text.p =
300                               source->string.buffer.start =
301                                 ALLOC_WC((int) (makefile_name->stat.size + 2));
302                         if (makefile_type == reading_cpp_file) {
303                                 forget_after_parse = true;
304                         }
305                         source->string.text.end = source->string.text.p;
306                         source->string.buffer.end =
307                           source->string.text.p + makefile_name->stat.size;
308                 } else {
309                         /* Do we ever reach here? */
310                         source->fd = -1;
311                         source->string.text.p =
312                           source->string.buffer.start =
313                             makefile->body.makefile.contents;
314                         source->string.text.end =
315                           source->string.buffer.end =
316                             source->string.text.p + makefile->body.makefile.size
317                         source->bytes_left_in_file =
318                           makefile->body.makefile.size;
319                 }
320                 file_being_read = wcb;
321         } else {
322                 char            *stdin_text_p;
323                 char            *stdin_text_end;
324                 char            *stdin_buffer_start;
```



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

```
391                 }
392                 (void) retmem_mb(stdin_text_p);
393         }
394         line_number = 1;
395         if (trace_reader) {
396                 (void) printf(catgets(catd, 1, 73, ">>>>>>>>>>>>>>>> Reading mak
397                             makefile_name->string_mb);
398         }
399         parse_makefile(makefile_name, source);
400         if (trace_reader) {
401                 (void) printf(catgets(catd, 1, 74, ">>>>>>>>>>>>>>>> End of make
402                             makefile_name->string_mb);
403         }
404         if(file_being_read) {
405                 retmem(file_being_read);
406         }
407         file_being_read = previous_file_being_read;
408         line_number = previous_line_number;
409         makefile_type = reading_nothing;
410         max_include_depth--;
411         if (make_state_locked) {
412                 /* Unlock .make.state. */
413                 unlink(make_state_lockfile);
414                 make_state_locked = false;
415                 retmem_mb(make_state_lockfile);
416         }
417         if (forget_after_parse) {
418                 retmem(makefile->body.makefile.contents);
419                 makefile->body.makefile.contents = NULL;
420         }
421         retmem_mb((char *)source);
422         return succeeded;
423 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   20893 Wed May 20 11:51:06 2015
new/usr/src/cmd/make/lib/mksh/dosys.cc
make: avo_alloca.h works around a very old Solaris bug, delete it
**********************************************************
```

```
/*
 * 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 <alloca.h>             /* alloca() */
#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() */
```



```
/*
 * 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);
        if (res < 0 && (errno == ESTALE || errno == EAGAIN)) {
                /* Stale NFS file handle. Try again */
                res = open(path, oflag, mode);
        }
        return res;
}
```

*unchanged_portion_omitted_*