****
668 Fri Jun 12 18:13:47 2015
new/usr/src/cmd/make/Makefile.com
fixup! make: build and install
****

```
#
# This file and its contents are supplied under the terms of the
# Common Development and Distribution License ("CDDL"), version 1.0.
# You may only use this file in accordance with the terms of version
# 1.0 of the CDDL.
#
# A full copy of the text of the CDDL should have accompanied this
# source.  A copy of the CDDL is also available via the Internet at
# http://www.illumos.org/license/CDDL.
#

# Copyright 2015, Richard Lowe.

MAKE_INCLUDE= $(SRC)/cmd/make/include
$(RELEASE_BUILD)MAKE_DEFS += -DNDEBUG
CFLAGS += $(CCVERBOSE)
CPPFLAGS += -I$(MAKE_INCLUDE) $(MAKE_DEFS)

# So that it's set even for the libraries we build
TEXT_DOMAIN = SUNW_OST_OSCMD

$(POFILE): $(POFILES)
        $(CAT) $(POFILES) > $@
```

```
**********************************************************
   86716 Fri Jun 12 18:13:49 2015
new/usr/src/cmd/make/bin/main.cc
fixup! make: be serial if 'make', parallel if 'dmake', and parallel if '-j' is s
fixup! make: translate using gettext, rather than the unmaintainable catgets
fixup! make: unifdef for two bugfixes conditioned for unknown reasons (defined)
**********************************************************
```

```
/*
 * 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.
 */

/*
 *      main.cc
 *
 *      make program main routine plus some helper routines
 */

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


#include <locale.h>             /* setlocale() */
#include <libgen.h>
#include <mk/defs.h>
#include <mksh/macro.h>         /* getvar() */
#include <mksh/misc.h>          /* getmem(), setup_char_semantics() */

#include <pwd.h>                /* getpwnam() */
#include <setjmp.h>
#include <signal.h>
#include <stdlib.h>
#include <sys/errno.h>          /* ENOENT */
#include <sys/stat.h>           /* fstat() */
#include <fcntl.h>              /* open() */

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

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

// From read2.cc
```

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

// From parallel.cc
#define MAXJOBS_ADJUST_RFE4694000

extern void job_adjust_fini();


/*
 * Defined macros
 */
#define LD_SUPPORT_ENV_VAR      "SGS_SUPPORT_32"
#define LD_SUPPORT_ENV_VAR_32   "SGS_SUPPORT_32"
#define LD_SUPPORT_ENV_VAR_64   "SGS_SUPPORT_64"
#define LD_SUPPORT_MAKE_LIB     "libmakestate.so.1"
#ifdef __i386
#define LD_SUPPORT_MAKE_ARCH    "i386"
#elif __sparc
#define LD_SUPPORT_MAKE_ARCH    "sparc"
#else
#error "Unsupported architecture"
#endif

/*
 * typedefs & structs
 */

/*
 * Static variables
 */
static  char            *argv_zero_string;
static  Boolean         build_failed_ever_seen;
static  Boolean         continue_after_error_ever_seen; /* '-k' */
static  Boolean         dmake_group_specified;          /* '-g' */
static  Boolean         dmake_max_jobs_specified;       /* '-j' */
static  Boolean         dmake_mode_specified;           /* '-m' */
static  Boolean         dmake_add_mode_specified;       /* '-x' */
static  Boolean         dmake_output_mode_specified;    /* '-x DMAKE_OUTPUT_MODE
static  Boolean         dmake_compat_mode_specified;    /* '-x SUN_MAKE_COMPAT_M
static  Boolean         dmake_odir_specified;           /* '-o' */
static  Boolean         dmake_rcfile_specified;         /* '-c' */
static  Boolean         env_wins;                       /* '-e' */
static  Boolean         ignore_default_mk;              /* '-r' */
static  Boolean         list_all_targets;               /* '-T' */
static  int             mf_argc;
static  char            **mf_argv;
static  Dependency_rec  not_auto_depen_struct;
static  Dependency      not_auto_depen = &not_auto_depen_struct;
static  Boolean         pmake_cap_r_specified;          /* '-R' */
static  Boolean         pmake_machinesfile_specified;   /* '-M' */
static  Boolean         stop_after_error_ever_seen;     /* '-S' */
static  Boolean         trace_status;                   /* '-p' */

#ifdef DMAKE_STATISTICS
static  Boolean         getname_stat = false;
#endif

        static  time_t          start_time;
        static  int             g_argc;
        static  char            **g_argv;

/*
 * File table of contents
 */
        extern "C" void         cleanup_after_exit(void);
extern "C" {
        extern  void            dmake_exit_callback(void);
        extern  void            dmake_message_callback(char *);
}

extern  Name            normalize_name(register wchar_t *name_string, register i

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

static  void            append_makeflags_string(Name, String);
static  void            doalarm(int);
static  void            enter_argv_values(int , char **, ASCII_Dyn_Array *);
static  void            make_targets(int, char **, Boolean);
static  int             parse_command_option(char);
static  void            read_command_options(int, char **);
static  void            read_environment(Boolean);
static  void            read_files_and_state(int, char **);
static  Boolean         read_makefile(Name, Boolean, Boolean, Boolean);
static  void            report_recursion(Name);
static  void            set_sgs_support(void);
static  void            setup_for_projectdir(void);
static  void            setup_makeflags_argv(void);
static  void            report_dir_enter_leave(Boolean entering);

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

static const char       verstring[] = "illumos make";

jmp_buf jmpbuffer;

/*
 *      main(argc, argv)
 *
 *      Parameters:
 *              argc                    You know what this is
 *              argv                    You know what this is
 *
 *      Static variables used:
 *              list_all_targets        make -T seen
 *              trace_status            make -p seen
 *
 *      Global variables used:
 *              debug_level             Should we trace make actions?
 *              keep_state              Set if .KEEP_STATE seen
 *              makeflags               The Name "MAKEFLAGS", used to get macro
 *              remote_command_name     Name of remote invocation cmd ("on")
 *              running_list            List of parallel running processes
 *              stdout_stderr_same      true if stdout and stderr are the same
 *              auto_dependencies       The Name "SUNPRO_DEPENDENCIES"
 *              temp_file_directory     Set to the dir where we create tmp file
 *              trace_reader            Set to reflect tracing status
 *              working_on_targets      Set when building user targets
 */
int
main(int argc, char *argv[])
{
        /*
         * cp is a -> to the value of the MAKEFLAGS env var,
         * which has to be regular chars.
         */
        register char           *cp;
        char                    make_state_dir[MAXPATHLEN];
        Boolean                 parallel_flag = false;
        char                    *prognameptr;
        char                    *slash_ptr;
        mode_t                  um;
```

```
        int                     i;
        struct itimerval        value;
        char                    def_dmakerc_path[MAXPATHLEN];
        Name                    dmake_name, dmake_name2;
        Name                    dmake_value, dmake_value2;
        Property                prop, prop2;
        struct stat             statbuf;
        int                     statval;

        struct stat             out_stat, err_stat;
        hostid = gethostid();
        bsd_signals();

        (void) setlocale(LC_ALL, "");


#ifdef DMAKE_STATISTICS
        if (getenv("DMAKE_STATISTICS")) {
                getname_stat = true;
        }
#endif

#ifndef TEXT_DOMAIN
#define TEXT_DOMAIN     "SYS_TEST"
#endif
        textdomain(TEXT_DOMAIN);

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

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

        if (*argv[0] == (int) slash_char) {
                /* argv[0] starts with a slash */
                argv_zero_string = strdup(argv[0]);
        } else if (strchr(argv[0], (int) slash_char) == NULL) {
                /* argv[0] contains no slashes */
                argv_zero_string = strdup(argv[0]);
        } else {
                /*
                 * argv[0] contains at least one slash,
                 * but doesn’t start with a slash
                 */
                char    *tmp_current_path;
                char    *tmp_string;

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

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


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

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

        /*
         * Find the dmake_compat_mode: posix, sun, svr4, or gnu_style, .
         */
        char * dmake_compat_mode_var = getenv("SUN_MAKE_COMPAT_MODE");
        if (dmake_compat_mode_var != NULL) {
                if (0 == strcasecmp(dmake_compat_mode_var, "GNU")) {
                        gnu_style = true;
                }
                //svr4 = false;
                //posix = false;
        }

        /*
         * Temporary directory set up.
         */
        char * tmpdir_var = getenv("TMPDIR");
        if (tmpdir_var != NULL && *tmpdir_var == '/' && strlen(tmpdir_var) < MAX
                strcpy(mbs_buffer, tmpdir_var);
                for (tmpdir_var = mbs_buffer+strlen(mbs_buffer);
                        *(--tmpdir_var) == '/' && tmpdir_var > mbs_buffer;
                        *tmpdir_var = '\0');
                if (strlen(mbs_buffer) + 32 < MAXPATHLEN) { /* 32 = strlen("/dma
                        sprintf(mbs_buffer2, "%s/dmake.tst.%d.XXXXXX",
                                mbs_buffer, getpid());
                        int fd = mkstemp(mbs_buffer2);
                        if (fd >= 0) {
                                close(fd);
```

```
                                unlink(mbs_buffer2);
                                tmpdir = strdup(mbs_buffer);
                        }
                }
        }

        /* find out if stdout and stderr point to the same place */
        if (fstat(1, &out_stat) < 0) {
                fatal(gettext("fstat of standard out failed: %s"), errmsg(errno)
        }
        if (fstat(2, &err_stat) < 0) {
                fatal(gettext("fstat of standard error failed: %s"), errmsg(errn
        }
        if ((out_stat.st_dev == err_stat.st_dev) &&
            (out_stat.st_ino == err_stat.st_ino)) {
                stdout_stderr_same = true;
        } else {
                stdout_stderr_same = false;
        }
        /* Make the vroot package scan the path using shell semantics */
        set_path_style(0);

        setup_char_semantics();

        setup_for_projectdir();

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

        load_cached_names();

/*
 *      Set command line flags
 */
        setup_makeflags_argv();
        read_command_options(mf_argc, mf_argv);
        read_command_options(argc, argv);
        if (debug_level > 0) {
                cp = getenv(makeflags->string_mb);
                (void) printf(gettext("MAKEFLAGS value: %s\n"), cp == NULL ? ""
        }

        setup_interrupt(handle_interrupt);

        read_files_and_state(argc, argv);

        /*
         * Find the dmake_output_mode: TXT1, TXT2 or HTML1.
         */
        MBSTOWCS(wcs_buffer, "DMAKE_OUTPUT_MODE");
        dmake_name2 = GETNAME(wcs_buffer, FIND_LENGTH);
        prop2 = get_prop(dmake_name2->prop, macro_prop);
        if (prop2 == NULL) {
                /* DMAKE_OUTPUT_MODE not defined, default to TXT1 mode */
                output_mode = txt1_mode;
        } else {
                dmake_value2 = prop2->body.macro.value;
                if ((dmake_value2 == NULL) ||
                    (IS_EQUAL(dmake_value2->string_mb, "TXT1"))) {
                        output_mode = txt1_mode;
                } else if (IS_EQUAL(dmake_value2->string_mb, "TXT2")) {
                        output_mode = txt2_mode;
                } else if (IS_EQUAL(dmake_value2->string_mb, "HTML1")) {
                        output_mode = html1_mode;
                } else {
                        warning(gettext("Unsupported value ‘%s’ for DMAKE_OUTPUT
                              dmake_value2->string_mb);
                }
        }
        /*
         * Find the dmake_mode: parallel, or serial.
         */
    if ((!pmake_cap_r_specified) &&
        (!pmake_machinesfile_specified)) {
        char *s, *b;

        if ((s = strdup(argv[0])) == NULL)
                fatal(gettext("Out of memory"));

        b = basename(s);
        char *s = strdup(argv[0]);

        MBSTOWCS(wcs_buffer, "DMAKE_MODE");
        dmake_name2 = GETNAME(wcs_buffer, FIND_LENGTH);
        prop2 = get_prop(dmake_name2->prop, macro_prop);
        // If we’re invoked as ’make’ run serially, regardless of DMAKE_MODE
        // If we’re invoked as ’make’ but passed -j, run parallel
        // If we’re invoked as ’dmake’, without DMAKE_MODE, default parallel
        // If we’re invoked as ’dmake’ and DMAKE_MODE is set, honour it.
        if ((strcmp(b, "make") == 0) &&
        if ((strcmp(basename(s), "make") == 0) &&
            !dmake_max_jobs_specified) {
                dmake_mode_type = serial_mode;
                no_parallel = true;
        } else if (prop2 == NULL) {
                /* DMAKE_MODE not defined, default based on our name */
                if (strcmp(b, "dmake") == 0) {
                char *s = strdup(argv[0]);

                if (strcmp(basename(s), "dmake") == 0) {
                        dmake_mode_type = parallel_mode;
                        no_parallel = false;
                }
        } else {
                dmake_value2 = prop2->body.macro.value;
                if (IS_EQUAL(dmake_value2->string_mb, "parallel")) {
                        dmake_mode_type = parallel_mode;
                        no_parallel = false;
                } else if (IS_EQUAL(dmake_value2->string_mb, "serial")) {
                        dmake_mode_type = serial_mode;
                        no_parallel = true;
                } else {
                        fatal(gettext("Unknown dmake mode argument ‘%s’ after -m
                }
        }
        free(s);
    }

        parallel_flag = true;
        putenv(strdup("DMAKE_CHILD=TRUE"));

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

```
        }

        /*
         * 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("__DMAKE_SEPARATE_STDERR");
        if (dmake_sep_var == NULL || (0 != strcasecmp(dmake_sep_var, "NO"))) {
                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;
                        }
                }
        }


/*
 *      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.
 */
        if (keep_state) {
                maybe_append_prop(sunpro_dependencies, macro_prop)->
                  body.macro.exported = true;
        } 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);
                }
        }


        report_dir_enter_leave(true);

        make_targets(argc, argv, parallel_flag);

        report_dir_enter_leave(false);

        if (build_failed_ever_seen) {
                if (posix) {
                        exit_status = 1;
                }
                exit(1);
        }
        exit_status = 0;
        exit(0);
        /* NOTREACHED */
}
```

______***unchanged_portion_omitted_***

```
**********************************************************
    1608 Fri Jun 12 18:13:50 2015
new/usr/src/cmd/make/include/mksh/dosys.h
fixup! make: remove maketool support
**********************************************************
#ifndef _MKSH_DOSYS_H
#define _MKSH_DOSYS_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.
 */

#include <mksh/defs.h>
#include <vroot/vroot.h>

extern Boolean  await(register Boolean ignore_error, register Boolean silent_err
extern int      doexec(register wchar_t *command, register Boolean ignore_error,
extern int      doshell(wchar_t *command, register Boolean ignore_error, char *s
extern Doname   dosys_mksh(register Name command, register Boolean ignore_error,
extern void     redirect_io(char *stdout_file, char *stderr_file);
extern void     sh_command2string(register String command, register String desti

#endif
```

****
705 Fri Jun 12 18:13:51 2015
new/usr/src/man/man1s/Makefile
fixup! make: add the manual page
****

```
   1 #
   2 # This file and its contents are supplied under the terms of the
   3 # Common Development and Distribution License ("CDDL"), version 1.0.
   4 # You may only use this file in accordance with the terms of version
   5 # 1.0 of the CDDL.
   6 #
   7 # A full copy of the text of the CDDL should have accompanied this
   8 # source.  A copy of the CDDL is also available via the Internet
   9 # at http://www.illumos.org/license/CDDL.
  10 #

  12 #
  13 # Copyright 2011, Richard Lowe
  14 # Copyright 2013 Nexenta Systems, Inc.  All rights reserved.
  15 #

  17 include         $(SRC)/Makefile.master

  19 MANSECT=        1s

  21 MANFILES=       make.1s
  22 MANLINKS=       dmake.1s

  24 dmake.1s        := LINKSRC = make.1s
  25 #endif /* ! codereview */

  27 .KEEP_STATE:

  29 include         $(SRC)/man/Makefile.man

  31 install:        $(ROOTMANFILES) $(ROOTMANLINKS)
  22 install:        $(ROOTMANFILES)
```

****
1413 Fri Jun 12 18:13:51 2015
new/usr/src/pkg/manifests/developer-build-make.mf
fixup! make: add the manual page
****

```
#
# This file and its contents are supplied under the terms of the
# Common Development and Distribution License ("CDDL"), version 1.0.
# You may only use this file in accordance with the terms of version
# 1.0 of the CDDL.
#
# A full copy of the text of the CDDL should have accompanied this
# source.  A copy of the CDDL is also available via the Internet at
# http://www.illumos.org/license/CDDL.
#

# Copyright 2015, Richard Lowe.

set name=pkg.fmri value=pkg:/developer/build/make@$(PKGVERS)
set name=pkg.description value="Parallel make(1) build tool"
set name=pkg.summary value="Parallel make(1) build tool"
set name=info.classification \
    value="org.opensolaris.category.2008:Development/Source Code Management"
set name=variant.arch value=$(ARCH)
dir path=usr/share/lib/make
dir path=usr/share/man/man1
dir path=usr/share/man/man1s
file path=usr/bin/make mode=0555
file path=usr/lib/$(ARCH64)/libmakestate.so.1
file path=usr/lib/libmakestate.so.1
file path=usr/share/lib/make/make.rules
file path=usr/share/lib/make/svr4.make.rules
file path=usr/share/man/man1/sysV-make.1
file path=usr/share/man/man1s/make.1s
link path=usr/bin/dmake target=make
link path=usr/ccs/bin/make target=../../bin/make
link path=usr/ccs/lib/svr4.make target=../../bin/make
link path=usr/lib/svr4.make target=../bin/make
link path=usr/share/man/man1s/dmake.1s target=make.1s
#endif /* ! codereview */
link path=usr/xpg4/bin/make target=../../bin/make
```

****
633 Fri Jun 12 18:13:52 2015
new/usr/src/tools/make/Makefile.com
fixup! tools: build a tools copy of make (which the build will automatically use
****

```
#
# This file and its contents are supplied under the terms of the
# Common Development and Distribution License ("CDDL"), version 1.0.
# You may only use this file in accordance with the terms of version
# 1.0 of the CDDL.
#
# A full copy of the text of the CDDL should have accompanied this
# source.  A copy of the CDDL is also available via the Internet at
# http://www.illumos.org/license/CDDL.
#

# Copyright 2015, Richard Lowe.

MAKE_INCLUDE= $(SRC)/cmd/make/include
$(RELEASE_BUILD)MAKE_DEFS += -DNDEBUG

CC = $(NATIVECC)
CCC = $(NATIVECCC)
CFLAGS = $(NATIVE_CFLAGS)
CFLAGS += $(CCVERBOSE)
CPPFLAGS = $(CPPFLAGS.native)
CPPFLAGS += -I$(MAKE_INCLUDE) $(MAKE_DEFS)
```