```
**********************************************************
    2010 Tue Jan 15 20:51:36 2019
new/usr/src/tools/cw/Makefile
9899 cw(1onbld) should shadow more compilation
**********************************************************
```

```
#
# 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 2007 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#
# Copyright 2018 Joyent, Inc.

PROG    = cw

MAN1ONBLDFILES= cw.1onbld

include ../Makefile.tools

# Bootstrap problem -- we have to build cw before we can use it
i386_CC=                $(SPRO_VROOT)/bin/cc
sparc_CC=               $(SPRO_VROOT)/bin/cc
$(__GNUC)i386_CC=       $(GNUC_ROOT)/bin/gcc
$(__GNUC)sparc_CC=      $(GNUC_ROOT)/bin/gcc

CFLAGS += $(CCVERBOSE)

# Override CFLAGS.  This is needed only for bootstrap of cw.
$(__GNUC)CFLAGS=        -O -D__sun -Wall -Wno-unknown-pragmas -Werror \
                        -std=gnu99 -nodefaultlibs
$(__SUNC)CFLAGS=        -xspace -Xa -xildoff -errtags=yes -errwarn=%all \
                        -xc99=%all -W0,-xglobalstatic -v


$(__GNUC)LDLIBS +=      -lc
$(__GNUC)LDFLAGS=       $(MAPFILE.NES:%=-Wl,-M%)

$(ROOTONBLDMAN1ONBLDFILES) := FILEMODE=      644
CSTD=   $(CSTD_GNU99)
#endif /* ! codereview */

# Assume we don't have the install.bin available yet
INS.file=       $(RM) $@; $(CP) $< $(@D); $(CHMOD) $(FILEMODE) $@

.KEEP_STATE:

all: $(PROG) $(MAN1ONBLDFILES)
```



```
install: all .WAIT $(ROOTONBLDMACHPROG) $(ROOTONBLDMAN1ONBLDFILES)

lint: lint_PROG

clean:

#
# Not run by default: bootstrap...
check:
        $(ROOTONBLDBINMACH)/mandoc -Tlint -Wwarning $(MAN1ONBLDFILES)

include ../Makefile.targ
```

```
**********************************************************
    6241 Tue Jan 15 20:51:36 2019
new/usr/src/tools/cw/cw.1onbld
9899 cw(1onbld) should shadow more compilation
**********************************************************
   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 .\" Copyright 2010 Sun Microsystems, Inc.  All rights reserved.
  22 .\" Use is subject to license terms.
  23 .\"
  24 .\" Copyright 2018 Joyent, Inc.
  25 .\"
  26 .Dd September 4, 2018
  27 .Dt CW 1ONBLD
  28 .Os
  29 .Sh NAME
  30 .Nm cw
  31 .Nd invoke one or more compilers with argument translation
  32 .Sh SYNOPSIS
  33 .Nm cw
  34 .Op Fl C
  35 .Op Fl -versions
  36 .Op Fl -noecho
  37 .Fl -primary Ar compiler
  38 .Op Fl -shadow Ar compiler ...
  39 .Fl -
  40 .Ar compiler args ...
  41 .Sh DESCRIPTION
  42 .Nm cw
  43 is a facility for invoking one or more compilers, providing translation from
  44 Sun style arguments as appropriate.
  45 This allows the use of arbitrary compilers without the need to alter large
  46 numbers of makefiles.
  47 A mode called shadow compilation invokes multiple compilers so that warnings
  48 and errors may be obtained from all of them.
  49 See
  50 .Sx SHADOW COMPILATION
  51 for details.
  52 This version of cw supports compilers with both Sun Studio 12 and GCC-style
  53 command lines.
  54 .Sh ARGUMENTS
  55 Both the
  56 .Fl -primary
  57 and
  58 .Fl -shadow
  59 parameters take a
  60 .Em compiler specification .
  61 This is a comma-separated list of the form
```



```
  62 .Ar name,executable,style
  63 Where
  64 .Ar name
  65 is a name for the compiler,
  66 .Ar executable
  67 is the full path to the compiler executable, and
  68 .Ar style
  69 is the style of command-line options the compiler expects, either
  70 .Em sun
  71 or
  72 .Em gnu .
  73 .Bl -tag -width indent
  74 .It Fl -primary Ar compiler
  75 Specify the compiler to be used primarily (that which is used for link-editing
  76 and pre-processing, and whos objects we deliver).
  77 .It Fl -shadow Ar compiler
  78 Specify a shadow compiler, which builds sources for the sake of checking code
  79 quality and compatibility, but has its output discarded.
  80 .It Fl -noecho
  81 Do not echo the actual command line of any compilers invoked.
  82 .It Fl -versions
  83 Request from each configured primary and shadow compiler its version
  84 information.
  85 .It Fl C
  86 The sources being compiled are C++.  This is necessary as it affects the
  87 translation of compiler arguments.
  88 .It Fl -
  89 Arguments intended for the compilers themselves must be separated from those
  90 of
  91 .Nm cw
  92 by a
  93 .Fl - .
  94 .It Fl _name=
  95 .It Fl _style=
  96 Parameters intended for the compiler be guarded with options of the form
  97 .Fl _name=
  98 and
  99 .Fl _style=
 100 Where
 101 .Em name
 102 and
 103 .Em style
 104 are those passed to
 105 .Fl -primary
 106 and
 107 .Fl -shadow
 108 this allows certain flags to be passed only to certain classes of compiler.
 109 .Pp
 110 For historical reasons, the
 111 .Fl _style=
 112 option is also translated such that a style of
 113 .Em sun
 114 may use the flag
 115 .Fl _cc=
 116 and a style of
 117 .Em gnu
 118 may use the flag
 119 .Fl _gcc= ,
 120 and when the
 121 .Fl C
 122 option is given and C++ is in use the style of
 123 .Em sun
 124 may use the flag
 125 .Fl _CC=
 126 and the style of
 127 .Em gnu
```

```
may use the flag
.Fl _g++= .
.El
.Sh SHADOW COMPILATION
If
.Fl -shadow
compilers are specified
.Nm cw
will invoke each shadow compiler, with the outputs modified (as well as any
translation for compiler style) as follows:
.Bl -enum
.It
If
.Nm cw
is invoked to link-edit without compilation (the input files are all objects),
the shadow compiler is not invoked.
If neither of
.Fl c ,
.Fl S
appears in the argument list (that is, linking is attempted or only the
pre-processor is invoked), the shadow compilers will not be invoked.
.It
If the
.Fl o Ar filename
option was provided, with or without a separating space, it will be replaced wit
.Fl o Ar tempfile
.It
If the option
.Fl o
was not provided,
.Fl o Ar tempfile
will be added to the end of the argument list used to invoke
the shadow compilers.
.El
When shadow compilation is in effect,
.Nm cw
writes to standard error each compiler’s standard error output following its
argument list.
Messages from the compilers will not be interleaved.
If
.Nm cw
is used to invoke the preprocessor and no output location is specified,
.Nm cw
will write to standard output the primary compiler’s standard output.
.Pp
Because the Sun compilers write intermediate objects to fixed
filenames in the current directory when instructed to compile and
link multiple source files via a single command line, it would be
unsafe to invoke more than one compiler in this fashion.
Therefore
.Nm cw
does not accept multiple source files unless the preprocessor is to be
invoked.
An attempt to invoke
.Nm cw
in this manner will result in an error.
.Sh ARGUMENT TRANSLATION
If the compiler to be invoked is a GNU-style C or C++ compiler, a set of
default flags is added to the beginning of the argument list, and the
remaining arguments are translated to their closest appropriate
semantic equivalents and passed in the same order as their
counterparts given to
.Nm cw .
See the comments at the head of
.Pa usr/src/tools/cw/cw.c
for a detailed list of translations.
```



```
.Sh ENVIRONMENT
.Bl -tag -width indent
.It CW_SHADOW_SERIAL
If this variable is set in the environment, invoke the primary compiler, wait
for it to complete, then invoke the shadow compilers.
Normally the primary and shadow compilers are invoked in parallel.
.It CW_NO_EXEC
f this variable is set in the environment, write the usual output to
standard error but do not actually invoke any compiler.
This is useful for debugging the translation engine.
.El
.Sh EXIT STATUS
The following exit status values are returned:
.Bl -tag -width indent
.It 0
The primary compiler, and shadow compilers if invoked, all completed
successfully.
.It >0
A usage error occurred, or one or more compilers returned a nonzero
exit status.
.El
.Sh SEE ALSO
.Xr cc 1 ,
.Xr CC 1 ,
.Xr gcc 1
.Sh BUGS
The translations provided for gcc are not always exact and in some cases
reflect local policy rather than actual equivalence.
.Pp
Additional compiler types should be supported.
.Pp
The translation engine is hacky.
```

```
**********************************************************
   43870 Tue Jan 15 20:51:37 2019
new/usr/src/tools/cw/cw.c
9899 cw(1onbld) should shadow more compilation
**********************************************************

/*
 * 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 2018, Richard Lowe.
 */
/*
 * Copyright 2010 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 *
 * Copyright 2019 Joyent, Inc.
 */

/*
 * Wrapper for the GNU C compiler to make it accept the Sun C compiler
 * arguments where possible.
 *
 * Since the translation is inexact, this is something of a work-in-progress.
 *
 */

/* If you modify this file, you must increment CW_VERSION */
#define CW_VERSION      "3.0"

/*
 * -#           Verbose mode
 * -###         Show compiler commands built by driver, no compilation
 * -A<name[(tokens)]>   Preprocessor predicate assertion
 * -B<[static|dynamic]> Specify dynamic or static binding
 * -C           Prevent preprocessor from removing comments
 * -c           Compile only - produce .o files, suppress linking
 * -cg92        Alias for -xtarget=ss1000
 * -D<name[=token]>     Associate name with token as if by #define
 * -d[y|n]      dynamic [-dy] or static [-dn] option to linker
 * -E           Compile source through preprocessor only, output to stdout
 * -erroff=<t>  Suppress warnings specified by tags t(%none, %all, <tag list>)
 * -errtags=<a> Display messages with tags a(no, yes)
 * -errwarn=<t> Treats warnings specified by tags t(%none, %all, <tag list>)
 *              as errors
 * -fast        Optimize using a selection of options
 * -fd          Report old-style function definitions and declarations
 * -fnonstd     Initialize floating-point hardware to non-standard preferences
```



```
 * -fns[=<yes|no>] Select non-standard floating point mode
 * -fprecision=<p> Set FP rounding precision mode p(single, double, extended)
 * -fround=<r>  Select the IEEE rounding mode in effect at startup
 * -fsimple[=<n>] Select floating-point optimization preferences <n>
 * -fsingle     Use single-precision arithmetic (-Xt and -Xs modes only)
 * -ftrap=<t>   Select floating-point trapping mode in effect at startup
 * -fstore      force floating pt. values to target precision on assignment
 * -G           Build a dynamic shared library
 * -g           Compile for debugging
 * -H           Print path name of each file included during compilation
 * -h <name>    Assign <name> to generated dynamic shared library
 * -I<dir>      Add <dir> to preprocessor #include file search path
 * -i           Passed to linker to ignore any LD_LIBRARY_PATH setting
 * -keeptmp     Keep temporary files created during compilation
 * -L<dir>      Pass to linker to add <dir> to the library search path
 * -l<name>     Link with library lib<name>.a or lib<name>.so
 * -mc          Remove duplicate strings from .comment section of output files
 * -mr          Remove all strings from .comment section of output files
 * -mr,"string" Remove all strings and append "string" to .comment section
 * -mt          Specify options needed when compiling multi-threaded code
 * -native      Find available processor, generate code accordingly
 * -nofstore    Do not force floating pt. values to target precision
 *              on assignment
 * -norunpath   Do not build in a runtime path for shared libraries
 * -O           Use default optimization level (-xO2 or -xO3. Check man page.)
 * -o <outputfile> Set name of output file to <outputfile>
 * -P           Compile source through preprocessor only, output to .i  file
 * -p           Compile for profiling with prof
 * -Q[y|n]      Emit/don't emit identification info to output file
 * -R<dir[:dir]> Build runtime search path list into executable
 * -S           Compile and only generate assembly code (.s)
 * -s           Strip symbol table from the executable file
 * -t           Turn off duplicate symbol warnings when linking
 * -U<name>     Delete initial definition of preprocessor symbol <name>
 * -V           Report version number of each compilation phase
 * -v           Do stricter semantic checking
 * -W<c>,<arg>  Pass <arg> to specified component <c> (a,l,m,p,0,2,h,i,u)
 * -w           Suppress compiler warning messages
 * -Xa          Compile assuming ANSI C conformance, allow K & R extensions
 *              (default mode)
 * -Xs          Compile assuming (pre-ANSI) K & R C style code
 * -Xt          Compile assuming K & R conformance, allow ANSI C
 * -xarch=<a>   Specify target architecture instruction set
 * -xbuiltin[=<b>] When profitable inline, or substitute intrinisic functions
 *              for system functions, b={%all,%none}
 * -xCC         Accept C++ style comments
 * -xchip=<c>   Specify the target processor for use by the optimizer
 * -xcode=<c>   Generate different code for forming addresses
 * -xcrossfile[=<n>] Enable optimization and inlining across source files,
 *              n={0|1}
 * -xe          Perform only syntax/semantic checking, no code generation
 * -xF          Compile for later mapfile reordering or unused section
 *              elimination
 * -xhelp=<f>   Display on-line help information f(flags, readme, errors)
 * -xildoff     Cancel -xildon
 * -xildon      Enable use of the incremental linker, ild
 * -xinline=[<a>,...,<a>]  Attempt inlining of specified user routines,
 *              <a>={%auto,func,no%func}
 * -xlibmieee   Force IEEE 754 return values for math routines in
 *              exceptional cases
 * -xlibmil     Inline selected libm math routines for optimization
 * -xlic_lib=sunperf    Link in the Sun supplied performance libraries
 * -xlicinfo    Show license server information
 * -xmaxopt=[off,1,2,3,4,5] maximum optimization level allowed on #pragma opt
 * -xO<n>       Generate optimized code (n={1|2|3|4|5})
 * -xP          Print prototypes for function definitions
```

```
 128  * -xprofile=<p> Collect data for a profile or use a profile to optimize
 129  *              <p>={{collect,use}[:<path>],tcov}
 130  * -xregs=<r>   Control register allocation
 131  * -xs          Allow debugging without object (.o) files
 132  * -xsb         Compile for use with the WorkShop source browser
 133  * -xsbfast     Generate only WorkShop source browser info, no compilation
 134  * -xsfpconst   Represent unsuffixed floating point constants as single
 135  *              precision
 136  * -xspace      Do not do optimizations that increase code size
 137  * -xstrconst   Place string literals into read-only data segment
 138  * -xtarget=<t> Specify target system for optimization
 139  * -xtemp=<dir> Set directory for temporary files to <dir>
 140  * -xtime       Report the execution time for each compilation phase
 141  * -xunroll=n   Enable unrolling loops n times where possible
 142  * -Y<c>,<dir>  Specify <dir> for location of component <c> (a,l,m,p,0,h,i,u)
 143  * -YA,<dir>    Change default directory searched for components
 144  * -YI,<dir>    Change default directory searched for include files
 145  * -YP,<dir>    Change default directory for finding libraries files
 146  * -YS,<dir>    Change default directory for startup object files
 147  */

 149 /*
 150  * Translation table:
 151  */
 152 /*
 153  * -#                            -v
 154  * -###                          error
 155  * -A<name[(tokens)]>            pass-thru
 156  * -B<[static|dynamic]>          pass-thru (syntax error for anything else)
 157  * -C                            pass-thru
 158  * -c                            pass-thru
 159  * -cg92                         -m32 -mcpu=v8 -mtune=supersparc (SPARC only)
 160  * -D<name[=token]>              pass-thru
 161  * -dy or -dn                    -Wl,-dy or -Wl,-dn
 162  * -E                            pass-thru
 163  * -erroff=E_EMPTY_TRANSLATION_UNIT ignore
 164  * -errtags=%all                 -Wall
 165  * -errwarn=%all                 -Werror else -Wno-error
 166  * -fast                         error
 167  * -fd                           error
 168  * -fnonstd                      error
 169  * -fns[=<yes|no>]               error
 170  * -fprecision=<p>               error
 171  * -fround=<r>                   error
 172  * -fsimple[=<n>]                error
 173  * -fsingle[=<n>]                error
 174  * -ftrap=<t>                    error
 175  * -fstore                       error
 176  * -G                            pass-thru
 177  * -g                            pass-thru
 178  * -H                            pass-thru
 179  * -h <name>                     pass-thru
 180  * -I<dir>                       pass-thru
 181  * -i                            pass-thru
 182  * -keeptmp                      -save-temps
 183  * -L<dir>                       pass-thru
 184  * -l<name>                      pass-thru
 185  * -mc                           error
 186  * -mr                           error
 187  * -mr,"string"                  error
 188  * -mt                           -D_REENTRANT
 189  * -native                       error
 190  * -nofstore                     error
 191  * -nolib                        -nodefaultlibs
 192  * -norunpath                    ignore
 193  * -O                            -O1 (Check the man page to be certain)
```



```
 194  * -o <outputfile>              pass-thru
 195  * -P                           -E -o filename.i (or error)
 196  * -p                           pass-thru
 197  * -Q[y|n]                      error
 198  * -R<dir[:dir]>                pass-thru
 199  * -S                           pass-thru
 200  * -s                           -Wl,-s
 201  * -t                           -Wl,-t
 202  * -U<name>                     pass-thru
 203  * -V                           --version
 204  * -v                           -Wall
 205  * -Wa,<arg>                    pass-thru
 206  * -Wp,<arg>                    pass-thru except -xc99=<a>
 207  * -Wl,<arg>                    pass-thru
 208  * -W{m,0,2,h,i,u>              error/ignore
 209  * -xmodel=kernel               -ffreestanding -mcmodel=kernel -mno-red-zone
 210  * -Wu,-save_args               -msave-args
 211  * -w                           pass-thru
 212  * -Xa                          -std=iso9899:199409 or -ansi
 213  * -Xt                          error
 214  * -Xs                          -traditional -std=c89
 215  * -xarch=<a>                   table
 216  * -xbuiltin[=<b>]              -fbuiltin (-fno-builtin otherwise)
 217  * -xCC                         ignore
 218  * -xchip=<c>                   table
 219  * -xcode=<c>                   table
 220  * -xdebugformat=<format>       ignore (always use dwarf-2 for gcc)
 221  * -xcrossfile[=<n>]            ignore
 222  * -xe                          error
 223  * -xF                          error
 224  * -xhelp=<f>                   error
 225  * -xildoff                     ignore
 226  * -xildon                      ignore
 227  * -xinline                     ignore
 228  * -xlibmieee                   error
 229  * -xlibmil                     error
 230  * -xlic_lib=sunperf            error
 231  * -xmaxopt=[...]               error
 232  * -xO<n>                       -O<n>
 233  * -xP                          error
 234  * -xprofile=<p>                error
 235  * -xregs=<r>                   table
 236  * -xs                          error
 237  * -xsb                         error
 238  * -xsbfast                     error
 239  * -xsfpconst                   error
 240  * -xspace                      ignore (-not -Os)
 241  * -xstrconst                   ignore
 242  * -xtarget=<t>                 table
 243  * -xtemp=<dir>                 error
 244  * -xtime                       error
 245  * -xtransition                 -Wtransition
 246  * -xunroll=n                   error
 247  * -W0,-xdbggen=no%usedonly     -fno-eliminate-unused-debug-symbols
 248  *                              -fno-eliminate-unused-debug-types
 249  * -Y<c>,<dir>                  error
 250  * -YA,<dir>                    error
 251  * -YI,<dir>                    -nostdinc -I<dir>
 252  * -YP,<dir>                    error
 253  * -YS,<dir>                    error
 254  */

 256 #include <ctype.h>
 257 #include <err.h>
 258 #include <errno.h>
 259 #include <fcntl.h>
```

```
260 #include <getopt.h>
261 #include <stdio.h>
262 #include <stdlib.h>
263 #include <string.h>
264 #include <unistd.h>
265 #include <dirent.h>
266 #endif /* ! codereview */

268 #include <sys/param.h>
269 #include <sys/stat.h>
270 #include <sys/types.h>
271 #include <sys/utsname.h>
272 #include <sys/wait.h>

274 #define CW_F_CXX        0x01
275 #define CW_F_SHADOW     0x02
276 #define CW_F_EXEC       0x04
277 #define CW_F_ECHO       0x08
278 #define CW_F_XLATE      0x10
279 #define CW_F_PROG       0x20

281 typedef enum cw_op {
282         CW_O_NONE = 0,
283         CW_O_PREPROCESS,
284         CW_O_COMPILE,
285         CW_O_LINK
286 } cw_op_t;

288 struct aelist {
289         struct ae {
290                 struct ae *ae_next;
291                 char *ae_arg;
292         } *ael_head, *ael_tail;
293         int ael_argc;
294 };

296 typedef enum {
297         GNU,
298         SUN,
299         SMATCH
300 } compiler_style_t;

302 typedef struct {
303         char *c_name;
304         char *c_path;
305         compiler_style_t c_style;
306 } cw_compiler_t;

308 typedef struct cw_ictx {
309         struct cw_ictx  *i_next;
310         cw_compiler_t   *i_compiler;
311         struct aelist   *i_ae;
312         uint32_t        i_flags;
313         int             i_oldargc;
314         char            **i_oldargv;
315         pid_t           i_pid;
316         char            *i_tmpdir;
265         char            i_discard[MAXPATHLEN];
317         char            *i_stderr;
318 } cw_ictx_t;
```

*unchanged_portion_omitted_*

```
561 /*
562  * The compiler wants the output file to end in appropriate extension.  If
563  * we're generating a name from whole cloth (path == NULL), we assume that
564  * extension to be .o, otherwise we match the extension of the caller.
```



```
565  */
566 static char *
567 discard_file_name(cw_ictx_t *ctx, const char *path)
568 {
569         char *ret, *ext;
570         char tmpl[] = "cwXXXXXX";

572         if (path == NULL) {
573                 ext = ".o";
574         } else {
575                 ext = strrchr(path, '.');
576         }

578         /*
579          * We need absolute control over where the temporary file goes, since
580          * we rely on it for cleanup so tempnam(3C) and tmpnam(3C) are
581          * inappropriate (they use TMPDIR, preferentially).
582          *
583          * mkstemp(3C) doesn't actually help us, since the temporary file
584          * isn't used by us, only its name.
585          */
586         if (mktemp(tmpl) == NULL)
587                 nomem();

589         (void) asprintf(&ret, "%s/%s%s", ctx->i_tmpdir, tmpl,
590             (ext != NULL) ? ext : "");

592         if (ret == NULL)
593                 nomem();

595         return (ret);
596 }

598 #endif /* ! codereview */
599 static void
600 do_gcc(cw_ictx_t *ctx)
601 {
602         int c;
603         int nolibc = 0;
604         int in_output = 0, seen_o = 0, c_files = 0;
605         cw_op_t op = CW_O_LINK;
606         char *model = NULL;
607         char *nameflag;
608         int mflag = 0;

610         if (ctx->i_flags & CW_F_PROG) {
611                 newae(ctx->i_ae, "--version");
612                 return;
613         }

615         newae(ctx->i_ae, "-fident");
616         newae(ctx->i_ae, "-finline");
617         newae(ctx->i_ae, "-fno-inline-functions");
618         newae(ctx->i_ae, "-fno-builtin");
619         newae(ctx->i_ae, "-fno-asm");
620         newae(ctx->i_ae, "-fdiagnostics-show-option");
621         newae(ctx->i_ae, "-nodefaultlibs");

623 #if defined(__sparc)
624         /*
625          * The SPARC ldd and std instructions require 8-byte alignment of
626          * their address operand.  gcc correctly uses them only when the
627          * ABI requires 8-byte alignment; unfortunately we have a number of
628          * pieces of buggy code that doesn't conform to the ABI.  This
629          * flag makes gcc work more like Studio with -xmemalign=4.
630          */
```

```
631         newae(ctx->i_ae, "-mno-integer-ldd-std");
632 #endif

634         /*
635          * This is needed because 'u' is defined
636          * under a conditional on 'sun'.  Should
637          * probably just remove the conditional,
638          * or make it be dependent on '__sun'.
639          *
640          * -Dunix is also missing in enhanced ANSI mode
641          */
642         newae(ctx->i_ae, "-D__sun");

644         if (asprintf(&nameflag, "-_%s=", ctx->i_compiler->c_name) == -1)
645                 nomem();

647         /*
648          * Walk the argument list, translating as we go ..
649          */
650         while (--ctx->i_oldargc > 0) {
651                 char *arg = *++ctx->i_oldargv;
652                 size_t arglen = strlen(arg);

654                 if (*arg == '-') {
655                         arglen--;
656                 } else {
657                         /*
658                          * Discard inline files that gcc doesn't grok
659                          */
660                         if (!in_output && arglen > 3 &&
661                             strcmp(arg + arglen - 3, ".il") == 0)
662                                 continue;

664                         if (!in_output && arglen > 2 &&
665                             arg[arglen - 2] == '.' &&
666                             (arg[arglen - 1] == 'S' || arg[arglen - 1] == 's' ||
667                             arg[arglen - 1] == 'c' || arg[arglen - 1] == 'i'))
668                                 c_files++;

670                         /*
671                          * Otherwise, filenames and partial arguments
672                          * are passed through for gcc to chew on.  However,
673                          * output is always discarded for the secondary
674                          * compiler.
675                          */
676                         if ((ctx->i_flags & CW_F_SHADOW) && in_output) {
677                                 newae(ctx->i_ae, discard_file_name(ctx, arg));
678                         } else {
510                         if ((ctx->i_flags & CW_F_SHADOW) && in_output)
511                                 newae(ctx->i_ae, ctx->i_discard);
512                         else
679                                 newae(ctx->i_ae, arg);
680                         }
681 #endif /* ! codereview */
682                         in_output = 0;
683                         continue;
684                 }

686                 if (ctx->i_flags & CW_F_CXX) {
687                         if (strncmp(arg, "-_g++=", 6) == 0) {
688                                 newae(ctx->i_ae, strchr(arg, '=') + 1);
689                                 continue;
690                         }
691                         if (strncmp(arg, "-compat=", 8) == 0) {
692                                 /* discard -compat=4 and -compat=5 */
693                                 continue;
```



```
694                         }
695                         if (strcmp(arg, "-Qoption") == 0) {
696                                 /* discard -Qoption and its two arguments */
697                                 if (ctx->i_oldargc < 3)
698                                         error(arg);
699                                 ctx->i_oldargc -= 2;
700                                 ctx->i_oldargv += 2;
701                                 continue;
702                         }
703                         if (strcmp(arg, "-xwe") == 0) {
704                                 /* turn warnings into errors */
705                                 newae(ctx->i_ae, "-Werror");
706                                 continue;
707                         }
708                         if (strcmp(arg, "-norunpath") == 0) {
709                                 /* gcc has no corresponding option */
710                                 continue;
711                         }
712                         if (strcmp(arg, "-nolib") == 0) {
713                                 /* -nodefaultlibs is on by default */
714                                 nolibc = 1;
715                                 continue;
716                         }
717 #if defined(__sparc)
718                         if (strcmp(arg, "-cg92") == 0) {
719                                 mflag |= xlate_xtb(ctx->i_ae, "v8");
720                                 xlate(ctx->i_ae, "super", xchip_tbl);
721                                 continue;
722                         }
723 #endif  /* __sparc */
724                 }

726                 switch ((c = arg[1])) {
727                 case '_':
728                         if ((strncmp(arg, nameflag, strlen(nameflag)) == 0) ||
729                             (strncmp(arg, "-_gcc=", 6) == 0) ||
730                             (strncmp(arg, "-_gnu=", 6) == 0)) {
731                                 newae(ctx->i_ae, strchr(arg, '=') + 1);
732                         }
733                         break;
734                 case '#':
735                         if (arglen == 1) {
736                                 newae(ctx->i_ae, "-v");
737                                 break;
738                         }
739                         error(arg);
740                         break;
741                 case 'f':
742                         if ((strcmp(arg, "-fpic") == 0) ||
743                             (strcmp(arg, "-fPIC") == 0)) {
744                                 newae(ctx->i_ae, arg);
745                                 break;
746                         }
747                         error(arg);
748                         break;
749                 case 'g':
750                         newae(ctx->i_ae, "-gdwarf-2");
751                         break;
752                 case 'E':
753                         if (arglen == 1) {
754                                 newae(ctx->i_ae, "-xc");
755                                 newae(ctx->i_ae, arg);
756                                 op = CW_O_PREPROCESS;
757                                 nolibc = 1;
758                                 break;
759                         }
```

```
760                         error(arg);
761                         break;
762                 case 'c':
763                 case 'S':
764                         if (arglen == 1) {
765                                 op = CW_O_COMPILE;
766                                 nolibc = 1;
767                         }
768                         /* FALLTHROUGH */
769                 case 'C':
770                 case 'H':
771                 case 'p':
772                         if (arglen == 1) {
773                                 newae(ctx->i_ae, arg);
774                                 break;
775                         }
776                         error(arg);
777                         break;
778                 case 'A':
779                 case 'h':
780                 case 'I':
781                 case 'i':
782                 case 'L':
783                 case 'l':
784                 case 'R':
785                 case 'U':
786                 case 'u':
787                 case 'w':
788                         newae(ctx->i_ae, arg);
789                         break;
790                 case 'o':
791                         seen_o = 1;
792                         if (arglen == 1) {
793                                 in_output = 1;
794                                 newae(ctx->i_ae, arg);
795                         } else if (ctx->i_flags & CW_F_SHADOW) {
796                                 newae(ctx->i_ae, "-o");
797                                 newae(ctx->i_ae, discard_file_name(ctx, arg));
514                                 newae(ctx->i_ae, ctx->i_discard);
798                         } else {
799                                 newae(ctx->i_ae, arg);
800                         }
801                         break;
802                 case 'D':
803                         newae(ctx->i_ae, arg);
804                         /*
805                          * XXX  Clearly a hack ... do we need _KADB too?
806                          */
807                         if (strcmp(arg, "-D_KERNEL") == 0 ||
808                             strcmp(arg, "-D_BOOT") == 0)
809                                 newae(ctx->i_ae, "-ffreestanding");
810                         break;
811                 case 'd':
812                         if (arglen == 2) {
813                                 if (strcmp(arg, "-dy") == 0) {
814                                         newae(ctx->i_ae, "-Wl,-dy");
815                                         break;
816                                 }
817                                 if (strcmp(arg, "-dn") == 0) {
818                                         newae(ctx->i_ae, "-Wl,-dn");
819                                         break;
820                                 }
821                         }
822                         if (strcmp(arg, "-dalign") == 0) {
823                                 /*
824                                  * -dalign forces alignment in some cases;
```



```
825                                  * gcc does not need any flag to do this.
826                                  */
827                                 break;
828                         }
829                         error(arg);
830                         break;
831                 case 'e':
832                         if (strcmp(arg,
833                             "-erroff=E_EMPTY_TRANSLATION_UNIT") == 0) {
834                                 /*
835                                  * Accept but ignore this -- gcc doesn't
836                                  * seem to complain about empty translation
837                                  * units
838                                  */
839                                 break;
840                         }
841                         /* XX64 -- ignore all -erroff= options, for now */
842                         if (strncmp(arg, "-erroff=", 8) == 0)
843                                 break;
844                         if (strcmp(arg, "-errtags=yes") == 0) {
845                                 warnings(ctx->i_ae);
846                                 break;
847                         }
848                         if (strcmp(arg, "-errwarn=%all") == 0) {
849                                 newae(ctx->i_ae, "-Werror");
850                                 break;
851                         }
852                         error(arg);
853                         break;
854                 case 'G':
855                         newae(ctx->i_ae, "-shared");
856                         nolibc = 1;
857                         break;
858                 case 'k':
859                         if (strcmp(arg, "-keeptmp") == 0) {
860                                 newae(ctx->i_ae, "-save-temps");
861                                 break;
862                         }
863                         error(arg);
864                         break;
865                 case 'm':
866                         if (strcmp(arg, "-mt") == 0) {
867                                 newae(ctx->i_ae, "-D_REENTRANT");
868                                 break;
869                         }
870                         if (strcmp(arg, "-m64") == 0) {
871                                 newae(ctx->i_ae, "-m64");
872 #if defined(__x86)
873                                 newae(ctx->i_ae, "-mtune=opteron");
874 #endif
875                                 mflag |= M64;
876                                 break;
877                         }
878                         if (strcmp(arg, "-m32") == 0) {
879                                 newae(ctx->i_ae, "-m32");
880                                 mflag |= M32;
881                                 break;
882                         }
883                         error(arg);
884                         break;
885                 case 'B':       /* linker options */
886                 case 'M':
887                 case 'z':
888                         {
889                                 char *opt;
890                                 size_t len;
```

```
891                                 char *s;
892
893                                 if (arglen == 1) {
894                                         opt = *++ctx->i_oldargv;
895                                         if (opt == NULL || *opt == '\0')
896                                                 error(arg);
897                                         ctx->i_oldargc--;
898                                 } else {
899                                         opt = arg + 2;
900                                 }
901                                 len = strlen(opt) + 7;
902                                 if ((s = malloc(len)) == NULL)
903                                         nomem();
904                                 (void) snprintf(s, len, "-Wl,-%c%s", c, opt);
905                                 newae(ctx->i_ae, s);
906                                 free(s);
907                         }
908                         break;
909                 case 'O':
910                         if (arglen == 1) {
911                                 newae(ctx->i_ae, "-O");
912                                 break;
913                         }
914                         error(arg);
915                         break;
916                 case 'P':
917                         /*
918                          * We could do '-E -o filename.i', but that's hard,
919                          * and we don't need it for the case that's triggering
920                          * this addition.  We'll require the user to specify
921                          * -o in the Makefile.  If they don't they'll find out
922                          * in a hurry.
923                          */
924                         newae(ctx->i_ae, "-E");
925                         op = CW_O_PREPROCESS;
926                         nolibc = 1;
927                         break;
928                 case 's':
929                         if (arglen == 1) {
930                                 newae(ctx->i_ae, "-Wl,-s");
931                                 break;
932                         }
933                         error(arg);
934                         break;
935                 case 't':
936                         if (arglen == 1) {
937                                 newae(ctx->i_ae, "-Wl,-t");
938                                 break;
939                         }
940                         error(arg);
941                         break;
942                 case 'V':
943                         if (arglen == 1) {
944                                 ctx->i_flags &= ~CW_F_ECHO;
945                                 newae(ctx->i_ae, "--version");
946                                 break;
947                         }
948                         error(arg);
949                         break;
950                 case 'v':
951                         if (arglen == 1) {
952                                 warnings(ctx->i_ae);
953                                 break;
954                         }
955                         error(arg);
956                         break;
```



```
957                 case 'W':
958                         if (strncmp(arg, "-Wp,-xc99", 9) == 0) {
959                                 /*
960                                  * gcc's preprocessor will accept c99
961                                  * regardless, so accept and ignore.
962                                  */
963                                 break;
964                         }
965                         if (strncmp(arg, "-Wa,", 4) == 0 ||
966                             strncmp(arg, "-Wp,", 4) == 0 ||
967                             strncmp(arg, "-Wl,", 4) == 0) {
968                                 newae(ctx->i_ae, arg);
969                                 break;
970                         }
971                         if (strcmp(arg, "-W0,-noglobal") == 0 ||
972                             strcmp(arg, "-W0,-xglobalstatic") == 0) {
973                                 /*
974                                  * gcc doesn't prefix local symbols
975                                  * in debug mode, so this is not needed.
976                                  */
977                                 break;
978                         }
979                         if (strcmp(arg, "-W0,-Lt") == 0) {
980                                 /*
981                                  * Generate tests at the top of loops.
982                                  * There is no direct gcc equivalent, ignore.
983                                  */
984                                 break;
985                         }
986                         if (strcmp(arg, "-W0,-xdbggen=no%usedonly") == 0) {
987                                 newae(ctx->i_ae,
988                                     "-fno-eliminate-unused-debug-symbols");
989                                 newae(ctx->i_ae,
990                                     "-fno-eliminate-unused-debug-types");
991                                 break;
992                         }
993                         if (strcmp(arg, "-W2,-xwrap_int") == 0) {
994                                 /*
995                                  * Use the legacy behaviour (pre-SS11)
996                                  * for integer wrapping.
997                                  * gcc does not need this.
998                                  */
999                                 break;
1000                        }
1001                        if (strcmp(arg, "-Wd,-xsafe=unboundsym") == 0) {
1002                                /*
1003                                 * Prevents optimizing away checks for
1004                                 * unbound weak symbol addresses.  gcc does
1005                                 * not do this, so it's not needed.
1006                                 */
1007                                break;
1008                        }
1009                        if (strncmp(arg, "-Wc,-xcode=", 11) == 0) {
1010                                xlate(ctx->i_ae, arg + 11, xcode_tbl);
1011                                break;
1012                        }
1013                        if (strncmp(arg, "-Wc,-Qiselect", 13) == 0) {
1014                                /*
1015                                 * Prevents insertion of register symbols.
1016                                 * gcc doesn't do this, so ignore it.
1017                                 */
1018                                break;
1019                        }
1020                        if (strcmp(arg, "-Wc,-Qassembler-ounrefsym=0") == 0) {
1021                                /*
1022                                 * Prevents optimizing away of static variables.
```

```
1023                                 * gcc does not do this, so it's not needed.
1024                                 */
1025                                break;
1026                        }
1027 #if defined(__x86)
1028                        if (strcmp(arg, "-Wu,-save_args") == 0) {
1029                                newae(ctx->i_ae, "-msave-args");
1030                                break;
1031                        }
1032 #endif  /* __x86 */
1033                        error(arg);
1034                        break;
1035                case 'X':
1036                        if (strcmp(arg, "-Xa") == 0 ||
1037                            strcmp(arg, "-Xt") == 0) {
1038                                break;
1039                        }
1040                        if (strcmp(arg, "-Xs") == 0) {
1041                                Xsmode(ctx->i_ae);
1042                                break;
1043                        }
1044                        error(arg);
1045                        break;
1046                case 'x':
1047                        if (arglen == 1)
1048                                error(arg);
1049                        switch (arg[2]) {
1050                        case 'a':
1051                                if (strncmp(arg, "-xarch=", 7) == 0) {
1052                                        mflag |= xlate_xtb(ctx->i_ae, arg + 7);
1053                                        break;
1054                                }
1055                                error(arg);
1056                                break;
1057                        case 'b':
1058                                if (strncmp(arg, "-xbuiltin=", 10) == 0) {
1059                                        if (strcmp(arg + 10, "%all"))
1060                                                newae(ctx->i_ae, "-fbuiltin");
1061                                        break;
1062                                }
1063                                error(arg);
1064                                break;
1065                        case 'C':
1066                                /* Accept C++ style comments -- ignore */
1067                                if (strcmp(arg, "-xCC") == 0)
1068                                        break;
1069                                error(arg);
1070                                break;
1071                        case 'c':
1072                                if (strncmp(arg, "-xc99=%all", 10) == 0) {
1073                                        newae(ctx->i_ae, "-std=gnu99");
1074                                        break;
1075                                }
1076                                if (strncmp(arg, "-xc99=%none", 11) == 0) {
1077                                        newae(ctx->i_ae, "-std=gnu89");
1078                                        break;
1079                                }
1080                                if (strncmp(arg, "-xchip=", 7) == 0) {
1081                                        xlate(ctx->i_ae, arg + 7, xchip_tbl);
1082                                        break;
1083                                }
1084                                if (strncmp(arg, "-xcode=", 7) == 0) {
1085                                        xlate(ctx->i_ae, arg + 7, xcode_tbl);
1086                                        break;
1087                                }
1088                                if (strncmp(arg, "-xcrossfile", 11) == 0)
```



```
1089                                        break;
1090                                error(arg);
1091                                break;
1092                        case 'd':
1093                                if (strncmp(arg, "-xdebugformat=", 14) == 0)
1094                                        break;
1095                                error(arg);
1096                                break;
1097                        case 'F':
1098                                /*
1099                                 * Compile for mapfile reordering, or unused
1100                                 * section elimination, syntax can be -xF or
1101                                 * more complex, like -xF=%all -- ignore.
1102                                 */
1103                                if (strncmp(arg, "-xF", 3) == 0)
1104                                        break;
1105                                error(arg);
1106                                break;
1107                        case 'i':
1108                                if (strncmp(arg, "-xinline", 8) == 0)
1109                                        /* No inlining; ignore */
1110                                        break;
1111                                if (strcmp(arg, "-xildon") == 0 ||
1112                                    strcmp(arg, "-xildoff") == 0)
1113                                        /* No incremental linking; ignore */
1114                                        break;
1115                                error(arg);
1116                                break;
1117 #if defined(__x86)
1118                        case 'm':
1119                                if (strcmp(arg, "-xmodel=kernel") == 0) {
1120                                        newae(ctx->i_ae, "-ffreestanding");
1121                                        newae(ctx->i_ae, "-mno-red-zone");
1122                                        model = "-mcmodel=kernel";
1123                                        nolibc = 1;
1124                                        break;
1125                                }
1126                                error(arg);
1127                                break;
1128 #endif  /* __x86 */
1129                        case 'O':
1130                                if (strncmp(arg, "-xO", 3) == 0) {
1131                                        size_t len = strlen(arg);
1132                                        char *s = NULL;
1133                                        int c = *(arg + 3);
1134                                        int level;
1135
1136                                        if (len != 4 || !isdigit(c))
1137                                                error(arg);
1138
1139                                        level = atoi(arg + 3);
1140                                        if (level > 5)
1141                                                error(arg);
1142                                        if (level >= 2) {
1143                                                /*
1144                                                 * For gcc-3.4.x at -O2 we
1145                                                 * need to disable optimizations
1146                                                 * that break ON.
1147                                                 */
1148                                                optim_disable(ctx->i_ae, level);
1149                                                /*
1150                                                 * limit -xO3 to -O2 as well.
1151                                                 */
1152                                                level = 2;
1153                                        }
1154                                        if (asprintf(&s, "-O%d", level) == -1)
```

```
1155                                            nomem();
1156                                    newae(ctx->i_ae, s);
1157                                    free(s);
1158                                    break;
1159                            }
1160                            error(arg);
1161                            break;
1162                    case 'r':
1163                            if (strncmp(arg, "-xregs=", 7) == 0) {
1164                                    xlate(ctx->i_ae, arg + 7, xregs_tbl);
1165                                    break;
1166                            }
1167                            error(arg);
1168                            break;
1169                    case 's':
1170                            if (strcmp(arg, "-xs") == 0 ||
1171                                strcmp(arg, "-xspace") == 0 ||
1172                                strcmp(arg, "-xstrconst") == 0)
1173                                    break;
1174                            error(arg);
1175                            break;
1176                    case 't':
1177                            if (strncmp(arg, "-xtarget=", 9) == 0) {
1178                                    xlate(ctx->i_ae, arg + 9, xtarget_tbl);
1179                                    break;
1180                            }
1181                            error(arg);
1182                            break;
1183                    case 'e':
1184                    case 'h':
1185                    case 'l':
1186                    default:
1187                            error(arg);
1188                            break;
1189                    }
1190                    break;
1191            case 'Y':
1192                    if (arglen == 1) {
1193                            if ((arg = *++ctx->i_oldargv) == NULL ||
1194                                *arg == '\0')
1195                                    error("-Y");
1196                            ctx->i_oldargc--;
1197                            arglen = strlen(arg + 1);
1198                    } else {
1199                            arg += 2;
1200                    }
1201                    /* Just ignore -YS,... for now */
1202                    if (strncmp(arg, "S,", 2) == 0)
1203                            break;
1204                    if (strncmp(arg, "l,", 2) == 0) {
1205                            char *s = strdup(arg);
1206                            s[0] = '-';
1207                            s[1] = 'B';
1208                            newae(ctx->i_ae, s);
1209                            free(s);
1210                            break;
1211                    }
1212                    if (strncmp(arg, "I,", 2) == 0) {
1213                            char *s = strdup(arg);
1214                            s[0] = '-';
1215                            s[1] = 'I';
1216                            newae(ctx->i_ae, "-nostdinc");
1217                            newae(ctx->i_ae, s);
1218                            free(s);
1219                            break;
1220                    }
```



```
1221                            error(arg);
1222                            break;
1223                    case 'Q':
1224                            /*
1225                             * We could map -Qy into -Wl,-Qy etc.
1226                             */
1227                    default:
1228                            error(arg);
1229                            break;
1230                    }
1231            }

1233            free(nameflag);

1235            /*
1236             * When compiling multiple source files in a single invocation some
1237             * compilers output objects into the current directory with
1238             * predictable and conventional names.
1239             *
1240             * We prevent any attempt to compile multiple files at once so that
1241             * any such objects created by a shadow can't escape into a later
1242             * link-edit.
1243             */
1244            if (c_files > 1 && op != CW_O_PREPROCESS) {
 952            if (c_files > 1 && (ctx->i_flags & CW_F_SHADOW) &&
 953                op != CW_O_PREPROCESS) {
1245                    errx(2, "multiple source files are "
1246                        "allowed only with -E or -P");
1247            }

1249            /*
1250             * Make sure that we do not have any unintended interactions between
1251             * the xarch options passed in and the version of the Studio compiler
1252             * used.
1253             */
1254            if ((mflag & (SS11|SS12)) == (SS11|SS12)) {
1255                    errx(2,
1256                        "Conflicting \"-xarch=\" flags (both Studio 11 and 12)\n");
1257            }

1259            switch (mflag) {
1260            case 0:
1261                    /* FALLTHROUGH */
1262            case M32:
1263 #if defined(__sparc)
1264                    /*
1265                     * Only -m32 is defined and so put in the missing xarch
1266                     * translation.
1267                     */
1268                    newae(ctx->i_ae, "-mcpu=v8");
1269                    newae(ctx->i_ae, "-mno-v8plus");
1270 #endif
1271                    break;
1272            case M64:
1273 #if defined(__sparc)
1274                    /*
1275                     * Only -m64 is defined and so put in the missing xarch
1276                     * translation.
1277                     */
1278                    newae(ctx->i_ae, "-mcpu=v9");
1279 #endif
1280                    break;
1281            case SS12:
1282 #if defined(__sparc)
1283                    /* no -m32/-m64 flag used - this is an error for sparc builds */
1284                    (void) fprintf(stderr, "No -m32/-m64 flag defined\n");
```

```
1285                 exit(2);
1286 #endif
1287                 break;
1288         case SS11:
1289                 /* FALLTHROUGH */
1290         case (SS11|M32):
1291         case (SS11|M64):
1292                 break;
1293         case (SS12|M32):
1294 #if defined(__sparc)
1295                 /*
1296                  * Need to add in further 32 bit options because with SS12
1297                  * the xarch=sparcvis option can be applied to 32 or 64
1298                  * bit, and so the translatation table (xtbl) cannot handle
1299                  * that.
1300                  */
1301                 newae(ctx->i_ae, "-mv8plus");
1302 #endif
1303                 break;
1304         case (SS12|M64):
1305                 break;
1306         default:
1307                 (void) fprintf(stderr,
1308                     "Incompatible -xarch= and/or -m32/-m64 options used.\n");
1309                 exit(2);
1310         }

1312         if (ctx->i_flags & CW_F_SHADOW) {
1313                 if (op == CW_O_PREPROCESS)
1314                         exit(0);
1315                 else if (op == CW_O_LINK && c_files == 0)
1021         if ((op == CW_O_LINK || op == CW_O_PREPROCESS) &&
1022             (ctx->i_flags & CW_F_SHADOW))
1316                         exit(0);
1317         }
1318 #endif /* ! codereview */

1320         if (model != NULL)
1321                 newae(ctx->i_ae, model);
1322         if (!nolibc)
1323                 newae(ctx->i_ae, "-lc");
1324         if (!seen_o && (ctx->i_flags & CW_F_SHADOW)) {
1325                 newae(ctx->i_ae, "-o");
1326                 newae(ctx->i_ae, discard_file_name(ctx, NULL));
1024                 newae(ctx->i_ae, ctx->i_discard);
1327         }
1328 }
```

_____*unchanged_portion_omitted_*

```
1356 static void
1357 do_cc(cw_ictx_t *ctx)
1358 {
1359         int in_output = 0, seen_o = 0, c_files = 0;
1057         int in_output = 0, seen_o = 0;
1360         cw_op_t op = CW_O_LINK;
1361         char *nameflag;

1363         if (ctx->i_flags & CW_F_PROG) {
1364                 newae(ctx->i_ae, "-V");
1365                 return;
1366         }

1368         if (asprintf(&nameflag, "-_%s=", ctx->i_compiler->c_name) == -1)
1369                 nomem();

1371         while (--ctx->i_oldargc > 0) {
```



```
1372                 char *arg = *++ctx->i_oldargv;
1373                 size_t arglen = strlen(arg);
1374 #endif /* ! codereview */

1376                 if (strncmp(arg, "-_CC=", 5) == 0) {
1377                         newae(ctx->i_ae, strchr(arg, '=') + 1);
1378                         continue;
1379                 }

1381                 if (*arg != '-') {
1382                         if (!in_output && arglen > 2 &&
1383                             arg[arglen - 2] == '.' &&
1384                             (arg[arglen - 1] == 'S' || arg[arglen - 1] == 's' ||
1385                             arg[arglen - 1] == 'c' || arg[arglen - 1] == 'i'))
1386                                 c_files++;

1388 #endif /* ! codereview */
1389                         if (in_output == 0 || !(ctx->i_flags & CW_F_SHADOW)) {
1390                                 newae(ctx->i_ae, arg);
1391                         } else {
1392                                 in_output = 0;
1393                                 newae(ctx->i_ae, discard_file_name(ctx, arg));
1071                                 newae(ctx->i_ae, ctx->i_discard);
1394                         }
1395                         continue;
1396                 }
1397                 switch (*(arg + 1)) {
1398                 case '_':
1399                         if ((strncmp(arg, nameflag, strlen(nameflag)) == 0) ||
1400                             (strncmp(arg, "-_cc=", 5) == 0) ||
1401                             (strncmp(arg, "-_sun=", 6) == 0)) {
1402                                 newae(ctx->i_ae, strchr(arg, '=') + 1);
1403                         }
1404                         break;

1406                 case 'V':
1407                         ctx->i_flags &= ~CW_F_ECHO;
1408                         newae(ctx->i_ae, arg);
1409                         break;
1410                 case 'o':
1411                         seen_o = 1;
1412                         if (strlen(arg) == 2) {
1413                                 in_output = 1;
1414                                 newae(ctx->i_ae, arg);
1415                         } else if (ctx->i_flags & CW_F_SHADOW) {
1416                                 newae(ctx->i_ae, "-o");
1417                                 newae(ctx->i_ae, discard_file_name(ctx, arg));
1095                                 newae(ctx->i_ae, ctx->i_discard);
1418                         } else {
1419                                 newae(ctx->i_ae, arg);
1420                         }
1421                         break;
1422                 case 'c':
1423                 case 'S':
1424                         if (strlen(arg) == 2)
1425                                 op = CW_O_COMPILE;
1426                         newae(ctx->i_ae, arg);
1427                         break;
1428                 case 'E':
1429                 case 'P':
1430                         if (strlen(arg) == 2)
1431                                 op = CW_O_PREPROCESS;
1432                 /*FALLTHROUGH*/
1433                 default:
1434                         newae(ctx->i_ae, arg);
1435                 }
```

```
1436         }

1438         free(nameflag);

1440         /* See the comment on this same code in do_gcc() */
1441         if (c_files > 1 && op != CW_O_PREPROCESS) {
1442                 errx(2, "multiple source files are "
1443                     "allowed only with -E or -P");
1444         }

1446         if (ctx->i_flags & CW_F_SHADOW) {
1447                 if (op == CW_O_PREPROCESS)
1118         if ((op == CW_O_LINK || op == CW_O_PREPROCESS) &&
1119             (ctx->i_flags & CW_F_SHADOW))
1448                         exit(0);
1449                 else if (op == CW_O_LINK && c_files == 0)
1450                         exit(0);
1451         }
1452 #endif /* ! codereview */

1454         if (!seen_o && (ctx->i_flags & CW_F_SHADOW)) {
1455                 newae(ctx->i_ae, "-o");
1456                 newae(ctx->i_ae, discard_file_name(ctx, NULL));
1121                 newae(ctx->i_ae, ctx->i_discard);
1457         }
1458 }
______unchanged_portion_omitted_

1538 static int
1539 reap(cw_ictx_t *ctx)
1540 {
1541         int status, ret = 0;
1542         char buf[1024];
1543         struct stat s;

1545         /*
1546          * Only wait for one specific child.
1547          */
1548         if (ctx->i_pid <= 0)
1549                 return (-1);

1551         do {
1552                 if (waitpid(ctx->i_pid, &status, 0) < 0) {
1553                         warn("cannot reap child");
1554                         return (-1);
1555                 }
1556                 if (status != 0) {
1557                         if (WIFSIGNALED(status)) {
1558                                 ret = -WTERMSIG(status);
1559                                 break;
1560                         } else if (WIFEXITED(status)) {
1561                                 ret = WEXITSTATUS(status);
1562                                 break;
1563                         }
1564                 }
1565         } while (!WIFEXITED(status) && !WIFSIGNALED(status));

1232         (void) unlink(ctx->i_discard);

1567         if (stat(ctx->i_stderr, &s) < 0) {
1568                 warn("stat failed on child cleanup");
1569                 return (-1);
1570         }
1571         if (s.st_size != 0) {
1572                 FILE *f;
```



```
1574                 if ((f = fopen(ctx->i_stderr, "r")) != NULL) {
1575                         while (fgets(buf, sizeof (buf), f))
1576                                 (void) fprintf(stderr, "%s", buf);
1577                         (void) fflush(stderr);
1578                         (void) fclose(f);
1579                 }
1580         }
1581         (void) unlink(ctx->i_stderr);
1582         free(ctx->i_stderr);

1584         /*
1585          * cc returns an error code when given -V; we want that to succeed.
1586          */
1587         if (ctx->i_flags & CW_F_PROG)
1588                 return (0);

1590         return (ret);
1591 }

1593 static int
1594 exec_ctx(cw_ictx_t *ctx, int block)
1595 {
1596         if ((ctx->i_stderr = tempnam(ctx->i_tmpdir, "cw")) == NULL) {
1263         char *file;

1265         /*
1266          * To avoid offending cc's sensibilities, the name of its output
1267          * file must end in '.o'.
1268          */
1269         if ((file = tempnam(NULL, ".cw")) == NULL) {
1270                 nomem();
1271                 return (-1);
1272         }
1273         (void) strlcpy(ctx->i_discard, file, MAXPATHLEN);
1274         (void) strlcat(ctx->i_discard, ".o", MAXPATHLEN);
1275         free(file);

1277         if ((ctx->i_stderr = tempnam(NULL, ".cw")) == NULL) {
1597                 nomem();
1598                 return (-1);
1599         }

1601         if ((ctx->i_pid = fork()) == 0) {
1602                 int fd;

1604                 (void) fclose(stderr);
1605                 if ((fd = open(ctx->i_stderr, O_WRONLY | O_CREAT | O_EXCL,
1606                     0666)) < 0) {
1607                         err(1, "open failed for standard error");
1608                 }
1609                 if (dup2(fd, 2) < 0) {
1610                         err(1, "dup2 failed for standard error");
1611                 }
1612                 if (fd != 2)
1613                         (void) close(fd);
1614                 if (freopen("/dev/fd/2", "w", stderr) == NULL) {
1615                         err(1, "freopen failed for /dev/fd/2");
1616                 }

1618                 prepctx(ctx);
1619                 exit(invoke(ctx));
1620         }

1622         if (ctx->i_pid < 0) {
1623                 err(1, "fork failed");
1624         }
```

```
1626         if (block)
1627                 return (reap(ctx));

1629         return (0);
1630 }
```

*unchanged_portion_omitted_*

```
1667 static void
1668 cleanup(cw_ictx_t *ctx)
1669 {
1670         DIR *dirp;
1671         struct dirent *dp;
1672         char buf[MAXPATHLEN];

1674         if ((dirp = opendir(ctx->i_tmpdir)) == NULL) {
1675                 if (errno != ENOENT) {
1676                         err(1, "couldn't open temp directory: %s",
1677                             ctx->i_tmpdir);
1678                 } else {
1679                         return;
1680                 }
1681         }

1683         errno = 0;
1684         while ((dp = readdir(dirp)) != NULL) {
1685                 (void) snprintf(buf, MAXPATHLEN, "%s/%s", ctx->i_tmpdir,
1686                     dp->d_name);

1688                 if (strncmp(dp->d_name, ".", strlen(dp->d_name)) == 0 ||
1689                     strncmp(dp->d_name, "..", strlen(dp->d_name)) == 0)
1690                         continue;

1692                 if (unlink(buf) == -1)
1693                         err(1, "failed to unlink temp file: %s", dp->d_name);
1694                 errno = 0;
1695         }

1697         if (errno != 0) {
1698                 err(1, "failed to read temporary directory: %s",
1699                     ctx->i_tmpdir);
1700         }

1702         (void) closedir(dirp);
1703         if (rmdir(ctx->i_tmpdir) != 0) {
1704                 err(1, "failed to unlink temporary directory: %s",
1705                     ctx->i_tmpdir);
1706         }
1707 }

1709 #endif /* ! codereview */
1710 int
1711 main(int argc, char **argv)
1712 {
1713         int ch;
1714         cw_compiler_t primary = { NULL, NULL, 0 };
1715         cw_compiler_t shadows[10];
1716         int nshadows = 0;
1717         int ret = 0;
1718         boolean_t do_serial = B_FALSE;
1719         boolean_t do_exec = B_FALSE;
1720         boolean_t vflg = B_FALSE;
1721         boolean_t Cflg = B_FALSE;
1722         boolean_t cflg = B_FALSE;
1723         boolean_t nflg = B_FALSE;
1724         char *tmpdir;
```



```
1725 #endif /* ! codereview */

1727         cw_ictx_t *main_ctx;

1729         static struct option longopts[] = {
1730                 { "compiler", no_argument, NULL, 'c' },
1731                 { "noecho", no_argument, NULL, 'n' },
1732                 { "primary", required_argument, NULL, 'p' },
1733                 { "shadow", required_argument, NULL, 's' },
1734                 { "versions", no_argument, NULL, 'v' },
1735                 { NULL, 0, NULL, 0 },
1736         };


1739         if ((main_ctx = newictx()) == NULL)
1740                 nomem();

1742         while ((ch = getopt_long(argc, argv, "C", longopts, NULL)) != -1) {
1743                 switch (ch) {
1744                 case 'c':
1745                         cflg = B_TRUE;
1746                         break;
1747                 case 'C':
1748                         Cflg = B_TRUE;
1749                         break;
1750                 case 'n':
1751                         nflg = B_TRUE;
1752                         break;
1753                 case 'p':
1754                         if (primary.c_path != NULL) {
1755                                 warnx("Only one primary compiler may "
1756                                     "be specified");
1757                                 usage();
1758                         }

1760                         parse_compiler(optarg, &primary);
1761                         break;
1762                 case 's':
1763                         if (nshadows >= 10)
1764                                 errx(1, "May only use 10 shadows at "
1765                                     "the moment");
1766                         parse_compiler(optarg, &shadows[nshadows]);
1767                         nshadows++;
1768                         break;
1769                 case 'v':
1770                         vflg = B_TRUE;
1771                         break;
1772                 default:
1773                         (void) fprintf(stderr, "Did you forget '--'?\n");
1774                         usage();
1775                 }
1776         }

1778         if (primary.c_path == NULL) {
1779                 warnx("A primary compiler must be specified");
1780                 usage();
1781         }

1783         do_serial = (getenv("CW_SHADOW_SERIAL") == NULL) ? B_FALSE : B_TRUE;
1784         do_exec = (getenv("CW_NO_EXEC") == NULL) ? B_TRUE : B_FALSE;

1786         /* Leave room for argv[0] */
1787         argc -= (optind - 1);
1788         argv += (optind - 1);

1790         main_ctx->i_oldargc = argc;
```

```
1791         main_ctx->i_oldargv = argv;
1792         main_ctx->i_flags = CW_F_XLATE;
1793         if (nflg == 0)
1794                 main_ctx->i_flags |= CW_F_ECHO;
1795         if (do_exec)
1796                 main_ctx->i_flags |= CW_F_EXEC;
1797         if (Cflg)
1798                 main_ctx->i_flags |= CW_F_CXX;
1799         main_ctx->i_compiler = &primary;

1801         if (cflg) {
1802                 (void) fputs(primary.c_path, stdout);
1803         }

1805         if (vflg) {
1806                 (void) printf("cw version %s\n", CW_VERSION);
1807                 (void) fflush(stdout);
1808                 main_ctx->i_flags &= ~CW_F_ECHO;
1809                 main_ctx->i_flags |= CW_F_PROG | CW_F_EXEC;
1810                 do_serial = 1;
1811         }

1813         tmpdir = getenv("TMPDIR");
1814         if (tmpdir == NULL)
1815                 tmpdir = "/tmp";

1817         if (asprintf(&main_ctx->i_tmpdir, "%s/cw.XXXXXX", tmpdir) == -1)
1818                 nomem();

1820         if ((main_ctx->i_tmpdir = mkdtemp(main_ctx->i_tmpdir)) == NULL)
1821                 errx(1, "failed to create temporary directory");

1823 #endif /* ! codereview */
1824         ret |= exec_ctx(main_ctx, do_serial);

1826         for (int i = 0; i < nshadows; i++) {
1827                 int r;
1828                 cw_ictx_t *shadow_ctx;

1830                 if ((shadow_ctx = newictx()) == NULL)
1831                         nomem();

1833                 (void) memcpy(shadow_ctx, main_ctx, sizeof (cw_ictx_t));
1348                 memcpy(shadow_ctx, main_ctx, sizeof (cw_ictx_t));

1835                 shadow_ctx->i_flags |= CW_F_SHADOW;
1836                 shadow_ctx->i_compiler = &shadows[i];

1838                 r = exec_ctx(shadow_ctx, do_serial);
1839                 if (r == 0) {
1840                         shadow_ctx->i_next = main_ctx->i_next;
1841                         main_ctx->i_next = shadow_ctx;
1842                 }
1843                 ret |= r;
1844         }

1846         if (!do_serial) {
1847                 cw_ictx_t *next = main_ctx;
1848                 while (next != NULL) {
1849                         cw_ictx_t *toreap = next;
1850                         next = next->i_next;
1851                         ret |= reap(toreap);
1852                 }
1853         }

1855         cleanup(main_ctx);
```



```
1856 #endif /* ! codereview */
1857         return (ret);
1858 }
```