

new/usr/src/tools/cw/Makefile

1

```
*****
2010 Fri Dec 28 22:22:46 2018
new/usr/src/tools/cw/Makefile
Cleanup temp files without rm
*****
1 #
2 # CDDL HEADER START
3 #
4 # The contents of this file are subject to the terms of the
5 # Common Development and Distribution License (the "License").
6 # You may not use this file except in compliance with the License.
7 #
8 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
9 # or http://www.opensolaris.org/os/licensing.
10 # See the License for the specific language governing permissions
11 # and limitations under the License.
12 #
13 # When distributing Covered Code, include this CDDL HEADER in each
14 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
15 # If applicable, add the following below this CDDL HEADER, with the
16 # fields enclosed by brackets "[]" replaced with your own identifying
17 # information: Portions Copyright [yyyy] [name of copyright owner]
18 #
19 # CDDL HEADER END
20 #
21 #
22 # Copyright 2007 Sun Microsystems, Inc. All rights reserved.
23 # Use is subject to license terms.
24 #
25 # Copyright 2018 Joyent, Inc.

27 PROG      = cw

29 MAN1ONBLDFILES= cw.1onbld

31 include ../Makefile.tools

33 # Bootstrap problem -- we have to build cw before we can use it
34 i386_CC=      $(SPRO_VROOT)/bin/cc
35 sparc_CC=     $(SPRO_VROOT)/bin/cc
36 $(__GNUC)i386_CC=  $(GNUC_ROOT)/bin/gcc
37 $(__GNUC)sparc_CC= $(GNUC_ROOT)/bin/gcc

39 CFLAGS += $(CCVERBOSE)

41 # Override CFLAGS. This is needed only for bootstrap of cw.
42 $(__GNUC)CFLAGS= -O -D_sun -Wall -Wno-unknown-pragmas -Werror \
43                -std=gnu99 -nodefaultlibs
44 $(__SUNC)CFLAGS= -xspace -Xa -xildoff -errtags=yes -errwarn=all \
45                -xc99=all -W0,-xglobalstatic -v

48 $(__GNUC)LDLIBS += -lc
49 $(__GNUC)LDFLAGS= $(MAPFILE.NES:%=-Wl,-M%)

51 $(ROOTONBLDMAN1ONBLDFILES) := FILEMODE= 644
52 CSTD= $(CSTD_GNU99)
53 #endif /* ! codereview */

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

58 .KEEP_STATE:

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

new/usr/src/tools/cw/Makefile

2

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

64 lint: lint_PROG

66 clean:

68 #
69 # Not run by default: bootstrap...
70 check:
71     $(ROOTONBLDBINMACH)/mandoc -Tlint -Wwarning $(MAN1ONBLDFILES)

73 include ../Makefile.targ
```

```

*****
6241 Fri Dec 28 22:22:47 2018
new/usr/src/tools/cw/cw.1onbld
Revert "Revert most of "9899 cw(1onbld) should shadow more compilation""
This reverts commit 67deef8cbc83060db238a0f4ee252d1ba74641ef.
*****
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
128 may use the flag
129 .Fl _g++= .
130 .El
131 .Sh SHADOW COMPILATION
132 If
133 .Fl -shadow
134 compilers are specified
135 .Nm cw
136 will invoke each shadow compiler, with the outputs modified (as well as any
137 translation for compiler style) as follows:
138 .Bl -enum
139 .It
140 If
141 .Nm cw
142 is invoked to link-edit without compilation (the input files are all objects),
143 the shadow compiler is not invoked.
144 If neither of
145 .Fl c ,
146 .Fl S
147 appears in the argument list (that is, linking is attempted or only the
148 pre-processor is invoked), the shadow compilers will not be invoked.
149 .It
150 If the option
151 .Fl o
152 was not provided,
153 .Fl o Ar tempfile
154 will be added to the end of the argument list used to invoke
155 the shadow compilers.
156 .El
157 When shadow compilation is in effect,
158 .Nm cw
159 writes to standard error each compiler's standard error output following its
160 argument list.
161 Messages from the compilers will not be interleaved.
162 If
163 .Nm cw
164 is used to invoke the preprocessor and no output location is specified,
165 .Nm cw
166 will write to standard output the primary compiler's standard output.
167 .Pp
168 Because the Sun compilers write intermediate objects to fixed
169 filenames in the current directory when instructed to compile and
170 link multiple source files via a single command line, it would be
171 unsafe to invoke more than one compiler in this fashion.
172 Therefore
173 .Nm cw
174 does not accept multiple source files unless the preprocessor is to be
175 invoked.
176 An attempt to invoke
177 .Nm cw
178 in this manner will result in an error.
179 .Sh ARGUMENT TRANSLATION
180 If the compiler to be invoked is a GNU-style C or C++ compiler, a set of
181 default flags is added to the beginning of the argument list, and the
182 remaining arguments are translated to their closest appropriate
183 semantic equivalents and passed in the same order as their
184 counterparts given to
185 .Nm cw .
186 See the comments at the head of
187 .Pa usr/src/tools/cw/cw.c

```

```

188 for a detailed list of translations.
189 .Sh ENVIRONMENT
190 .Bl -tag -width indent
191 .It CW_SHADOW_SERIAL
192 If this variable is set in the environment, invoke the primary compiler, wait
193 for it to complete, then invoke the shadow compilers.
194 Normally the primary and shadow compilers are invoked in parallel.
195 .It CW_NO_EXEC
196 f this variable is set in the environment, write the usual output to
197 standard error but do not actually invoke any compiler.
198 This is useful for debugging the translation engine.
199 .El
200 .Sh EXIT STATUS
201 The following exit status values are returned:
202 .Bl -tag -width indent
203 .It 0
204 The primary compiler, and shadow compilers if invoked, all completed
205 successfully.
206 .It >0
207 A usage error occurred, or one or more compilers returned a nonzero
208 exit status.
209 .El
210 .Sh SEE ALSO
211 .Xr cc 1 ,
212 .Xr CC 1 ,
213 .Xr gcc 1
214 .Sh BUGS
215 The translations provided for gcc are not always exact and in some cases
216 reflect local policy rather than actual equivalence.
217 .Pp
218 Additional compiler types should be supported.
219 .Pp
220 The translation engine is hacky.

```

```

*****
43270 Fri Dec 28 22:22:48 2018
new/usr/src/tools/cw/cw.c
Cleanup temp files without rm
Cleanup cw tempfiles properly
Revert "Revert most of "9899 cw(lonbld) should shadow more compilation""
This reverts commit 67deef8cbc83060db238a0f4ee252d1ba74641ef.
*****

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

23 /*
24 * Copyright 2018, Richard Lowe.
25 */
26 /*
27 * Copyright 2010 Sun Microsystems, Inc. All rights reserved.
28 * Use is subject to license terms.
29 */

31 /*
32 * Wrapper for the GNU C compiler to make it accept the Sun C compiler
33 * arguments where possible.
34 *
35 * Since the translation is inexact, this is something of a work-in-progress.
36 *
37 */

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

42 /*
43 * -# Verbose mode
44 * -### Show compiler commands built by driver, no compilation
45 * -A<name[(tokens)]> Preprocessor predicate assertion
46 * -B<[static|dynamic]> Specify dynamic or static binding
47 * -C Prevent preprocessor from removing comments
48 * -c Compile only - produce .o files, suppress linking
49 * -cg92 Alias for -xtarget=ss1000
50 * -D<name=[token]> Associate name with token as if by #define
51 * -d[y|n] dynamic [-dy] or static [-dn] option to linker
52 * -E Compile source through preprocessor only, output to stdout
53 * -erroff=<t> Suppress warnings specified by tags t(%none, %all, <tag list>)
54 * -errtags=<a> Display messages with tags a(no, yes)
55 * -errwarn=<t> Treats warnings specified by tags t(%none, %all, <tag list>)
56 * as errors
57 * -fast Optimize using a selection of options
58 * -fd Report old-style function definitions and declarations

```

```

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

```

```

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

147 /*
148 * Translation table:
149 */
150 /*
151 * -#             -v
152 * -###          error
153 * -A<name[(tokens)]> pass-thru
154 * -B<[static|dynamic]> pass-thru (syntax error for anything else)
155 * -C            pass-thru
156 * -c            pass-thru
157 * -cg92        -m32 -mcpu=v8 -mtune=supersparc (SPARC only)
158 * -D<name[=token]> pass-thru
159 * -dy or -dn   -Wl,-dy or -Wl,-dn
160 * -E            pass-thru
161 * -erroff=E_EMPTY_TRANSLATION_UNIT ignore
162 * -errtags=%all -Wall
163 * -errwarn=%all -Werror else -Wno-error
164 * -fast        error
165 * -fd          error
166 * -fnonstd     error
167 * -fns[=<yes|no>] error
168 * -fprecision=<p> error
169 * -fround=<r>  error
170 * -fsimple[=<n>] error
171 * -fsingle[=<n>] error
172 * -ftrap=<t>   error
173 * -fstore      error
174 * -G           pass-thru
175 * -g           pass-thru
176 * -H           pass-thru
177 * -h <name>    pass-thru
178 * -I<dir>     pass-thru
179 * -i           pass-thru
180 * -keeptmp     -save-temps
181 * -L<dir>     pass-thru
182 * -l<name>    pass-thru
183 * -mc          error
184 * -mr          error
185 * -mr,"string" error
186 * -mt         -D_REENTRANT
187 * -native     error
188 * -nofstore   error
189 * -nolib      -nodefaultlibs
190 * -norunpath  ignore

```

```

191 * -O           -O1 (Check the man page to be certain)
192 * -o <outputfile> pass-thru
193 * -P          -E -o filename.i (or error)
194 * -p          pass-thru
195 * -Q[y|n]     error
196 * -R<dir[:dir]> pass-thru
197 * -S          pass-thru
198 * -s          -Wl,-s
199 * -t          -Wl,-t
200 * -U<name>    pass-thru
201 * -V          --version
202 * -v          -Wall
203 * -Wa,<arg>   pass-thru
204 * -Wp,<arg>   pass-thru except -xc99=<a>
205 * -Wl,<arg>   pass-thru
206 * -W{m,0,2,h,i,u} error/ignore
207 * -xmodel=kernel -ffreestanding -mcmmodel=kernel -mno-red-zone
208 * -Wu,-save_args -msave_args
209 * -w          pass-thru
210 * -Xa         -std=iso9899:199409 or -ansi
211 * -Xt         error
212 * -Xs         -traditional -std=c89
213 * -xarch=<a>  table
214 * -xbuiltin[=<b>] -fbuiltin (-fno-builtin otherwise)
215 * -xCC        ignore
216 * -xchip=<c>  table
217 * -xcode=<c>  table
218 * -xdebugformat=<format> ignore (always use dwarf-2 for gcc)
219 * -xcrossfile[=<n>] ignore
220 * -xe         error
221 * -xF         error
222 * -xhelp=<f>  error
223 * -xildoff    ignore
224 * -xildon     ignore
225 * -xinline    ignore
226 * -xlibmieee  error
227 * -xlibmil    error
228 * -xlic_lib=sunperf error
229 * -xmaxopt=[...] error
230 * -xO<n>      -O<n>
231 * -xP         error
232 * -xprofile=<p> error
233 * -xregs=<r>  table
234 * -xs         error
235 * -xsb        error
236 * -xsbfast    error
237 * -xsfpconst error
238 * -xspace     ignore (-not -Os)
239 * -xstrconst  ignore
240 * -xtarget=<t> table
241 * -xtemp=<dir> error
242 * -xtime      error
243 * -xtransition -Wtransition
244 * -xunroll=n  error
245 * -W0,-xdbggen=no%usedonly -fno-eliminate-unused-debug-symbols
246 * -fno-eliminate-unused-debug-types
247 * -Y<c>,<dir> error
248 * -YA,<dir>   error
249 * -YI,<dir>   -nostdinc -I<dir>
250 * -YP,<dir>   error
251 * -YS,<dir>   error
252 */

254 #include <ctype.h>
255 #include <err.h>
256 #include <errno.h>

```

```

257 #include <fcntl.h>
258 #include <getopt.h>
259 #include <stdio.h>
260 #include <stdlib.h>
261 #include <string.h>
262 #include <unistd.h>
263 #include <dirent.h>
264 #endif /* ! codereview */

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

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

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

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

294 typedef enum {
295     GNU,
296     SUN
297 } compiler_style_t;

299 typedef struct {
300     char *c_name;
301     char *c_path;
302     compiler_style_t c_style;
303 } cw_compiler_t;

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

unchanged_portion_omitted

558 /*
559 * The compiler wants the output file to end in appropriate extension.  If
560 * we're generating a name from whole cloth (path == NULL), we assume that
561 * extension to be .o, otherwise we match the extension of the caller.

```

```

562 */
563 static char *
564 discard_file_name(cw_ictx_t *ctx, const char *path)
565 {
566     char *ret, *ext;
567     char tmp1[] = "cwXXXXXX";

569     if (path == NULL) {
570         ext = ".o";
571     } else {
572         ext = strrchr(path, '.');
573     }

575     /*
576     * We need absolute control over where the temporary file goes, since
577     * we rely on it for cleanup so tempnam(3C) and tmpnam(3C) are
578     * inappropriate (they use TMPDIR, preferentially).
579     *
580     * mkstemp(3C) doesn't actually help us, since the temporary file
581     * isn't used by us, only its name.
582     */
583     if (mktemp(tmp1) == NULL)
584         nomem();

586     (void) asprintf(&ret, "%s/%s%s", ctx->i_tmpdir, tmp1,
587                  (ext != NULL) ? ext : "");

589     if (ret == NULL)
590         nomem();

592     return (ret);
593 }

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

607     if (ctx->i_flags & CW_F_PROG) {
608         newae(ctx->i_ae, "--version");
609         return;
610     }

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

620 #if defined(__sparc)
621     /*
622     * The SPARC ldd and std instructions require 8-byte alignment of
623     * their address operand.  gcc correctly uses them only when the
624     * ABI requires 8-byte alignment; unfortunately we have a number of
625     * pieces of buggy code that doesn't conform to the ABI.  This
626     * flag makes gcc work more like Studio with -xmemalign=4.
627     */

```

```

628     newae(ctx->i_ae, "-mno-integer-ldd-std");
629 #endif

631     /*
632     * This is needed because 'u' is defined
633     * under a conditional on 'sun'. Should
634     * probably just remove the conditional,
635     * or make it be dependent on '__sun'.
636     *
637     * -Dunix is also missing in enhanced ANSI mode
638     */
639     newae(ctx->i_ae, "-D__sun");

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

644     /*
645     * Walk the argument list, translating as we go ..
646     */
647     while (--ctx->i_oldargc > 0) {
648         char *arg = ++ctx->i_oldargv;
649         size_t arglen = strlen(arg);

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

661             if (!in_output && arglen > 2 &&
662                 arg[arglen - 2] == '.' &&
663                 (arg[arglen - 1] == 's' || arg[arglen - 1] == 's' ||
664                  arg[arglen - 1] == 'c' || arg[arglen - 1] == 'i'))
665                 c_files++;

667             /*
668             * Otherwise, filenames and partial arguments
669             * are passed through for gcc to chew on. However,
670             * output is always discarded for the secondary
671             * compiler.
672             */
673             if ((ctx->i_flags & CW_F_SHADOW) && in_output) {
674                 newae(ctx->i_ae, discard_file_name(ctx, arg));
675             } else {
676                 if ((ctx->i_flags & CW_F_SHADOW) && in_output)
677                     newae(ctx->i_ae, ctx->i_discard);
678                 else
679                     newae(ctx->i_ae, arg);
680             }
681         }

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

```

```

691     }
692     if (strcmp(arg, "-Qoption") == 0) {
693         /* discard -Qoption and its two arguments */
694         if (ctx->i_oldargc < 3)
695             error(arg);
696         ctx->i_oldargc -= 2;
697         ctx->i_oldargv += 2;
698         continue;
699     }
700     if (strcmp(arg, "-xwe") == 0) {
701         /* turn warnings into errors */
702         newae(ctx->i_ae, "-Werror");
703         continue;
704     }
705     if (strcmp(arg, "-norunpath") == 0) {
706         /* gcc has no corresponding option */
707         continue;
708     }
709     if (strcmp(arg, "-nolib") == 0) {
710         /* -nodefaultlibs is on by default */
711         nolibc = 1;
712         continue;
713     }
714     #if defined(__sparc)
715     if (strcmp(arg, "-cg92") == 0) {
716         mflag |= xlate_xtb(ctx->i_ae, "v8");
717         xlate(ctx->i_ae, "super", xchip_tbl);
718         continue;
719     }
720     #endif /* __sparc */
721     }

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

```

```

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

```

```

822             * gcc does not need any flag to do this.
823             */
824             break;
825         }
826         error(arg);
827         break;
828     case 'e':
829         if (strcmp(arg,
830             "-errorff=E_EMPTY_TRANSLATION_UNIT") == 0) {
831             /*
832              * Accept but ignore this -- gcc doesn't
833              * seem to complain about empty translation
834              * units
835              */
836             break;
837         }
838         /* XX64 -- ignore all -errorff= options, for now */
839         if (strncmp(arg, "-errorff=", 8) == 0)
840             break;
841         if (strcmp(arg, "-errtags=yes") == 0) {
842             warnings(ctx->i_ae);
843             break;
844         }
845         if (strcmp(arg, "-errwarn=%all") == 0) {
846             newae(ctx->i_ae, "-Werror");
847             break;
848         }
849         error(arg);
850         break;
851     case 'G':
852         newae(ctx->i_ae, "--shared");
853         nolIBC = 1;
854         break;
855     case 'k':
856         if (strcmp(arg, "-keeptmp") == 0) {
857             newae(ctx->i_ae, "-save-temps");
858             break;
859         }
860         error(arg);
861         break;
862     case 'm':
863         if (strcmp(arg, "-mt") == 0) {
864             newae(ctx->i_ae, "-D_REENTRANT");
865             break;
866         }
867         if (strcmp(arg, "-m64") == 0) {
868             newae(ctx->i_ae, "-m64");
869             #if defined(__x86)
870                 newae(ctx->i_ae, "-mtune=opteron");
871             #endif
872             mflag |= M64;
873             break;
874         }
875         if (strcmp(arg, "-m32") == 0) {
876             newae(ctx->i_ae, "-m32");
877             mflag |= M32;
878             break;
879         }
880         error(arg);
881         break;
882     case 'B':
883     case 'M':
884     case 'z':
885         {
886             char *opt;
887             size_t len;

```



```

888         char *s;
889
890         if (arglen == 1) {
891             opt = **++ctx->i_oldargv;
892             if (opt == NULL || *opt == '\\0')
893                 error(arg);
894             ctx->i_oldargc--;
895         } else {
896             opt = arg + 2;
897         }
898         len = strlen(opt) + 7;
899         if ((s = malloc(len)) == NULL)
900             nomem();
901         (void) snprintf(s, len, "-Wl,-%c%s", c, opt);
902         newae(ctx->i_ae, s);
903         free(s);
904     }
905     break;
906 case 'O':
907     if (arglen == 1) {
908         newae(ctx->i_ae, "-O");
909         break;
910     }
911     error(arg);
912     break;
913 case 'P':
914     /*
915     * We could do '-E -o filename.i', but that's hard,
916     * and we don't need it for the case that's triggering
917     * this addition. We'll require the user to specify
918     * -o in the Makefile. If they don't they'll find out
919     * in a hurry.
920     */
921     newae(ctx->i_ae, "-E");
922     op = CW_O_PREPROCESS;
923     nolibc = 1;
924     break;
925 case 's':
926     if (arglen == 1) {
927         newae(ctx->i_ae, "-Wl,-s");
928         break;
929     }
930     error(arg);
931     break;
932 case 't':
933     if (arglen == 1) {
934         newae(ctx->i_ae, "-Wl,-t");
935         break;
936     }
937     error(arg);
938     break;
939 case 'V':
940     if (arglen == 1) {
941         ctx->i_flags &= ~CW_F_ECHO;
942         newae(ctx->i_ae, "--version");
943         break;
944     }
945     error(arg);
946     break;
947 case 'v':
948     if (arglen == 1) {
949         warnings(ctx->i_ae);
950         break;
951     }
952     error(arg);
953     break;

```

```

954     case 'W':
955         if (strncmp(arg, "-Wp,-xc99", 9) == 0) {
956             /*
957             * gcc's preprocessor will accept c99
958             * regardless, so accept and ignore.
959             */
960             break;
961         }
962         if (strncmp(arg, "-Wa,", 4) == 0 ||
963             strncmp(arg, "-Wp,", 4) == 0 ||
964             strncmp(arg, "-Wl,", 4) == 0) {
965             newae(ctx->i_ae, arg);
966             break;
967         }
968         if (strcmp(arg, "-W0,-noglobal") == 0 ||
969             strcmp(arg, "-W0,-xglobalstatic") == 0) {
970             /*
971             * gcc doesn't prefix local symbols
972             * in debug mode, so this is not needed.
973             */
974             break;
975         }
976         if (strcmp(arg, "-W0,-It") == 0) {
977             /*
978             * Generate tests at the top of loops.
979             * There is no direct gcc equivalent, ignore.
980             */
981             break;
982         }
983         if (strcmp(arg, "-W0,-xdbggen=no%usedonly") == 0) {
984             newae(ctx->i_ae,
985                 "-fno-eliminate-unused-debug-symbols");
986             newae(ctx->i_ae,
987                 "-fno-eliminate-unused-debug-types");
988             break;
989         }
990         if (strcmp(arg, "-W2,-xwrap_int") == 0) {
991             /*
992             * Use the legacy behaviour (pre-SS11)
993             * for integer wrapping.
994             * gcc does not need this.
995             */
996             break;
997         }
998         if (strcmp(arg, "-Wd,-xsafe=unboundsym") == 0) {
999             /*
1000            * Prevents optimizing away checks for
1001            * unbound weak symbol addresses. gcc does
1002            * not do this, so it's not needed.
1003            */
1004             break;
1005         }
1006         if (strncmp(arg, "-Wc,-xcode=", 11) == 0) {
1007             xlate(ctx->i_ae, arg + 11, xcode_tbl);
1008             break;
1009         }
1010         if (strncmp(arg, "-Wc,-Qiselect", 13) == 0) {
1011             /*
1012             * Prevents insertion of register symbols.
1013             * gcc doesn't do this, so ignore it.
1014             */
1015             break;
1016         }
1017         if (strcmp(arg, "-Wc,-Qassembler-ounrefsym=0") == 0) {
1018             /*
1019             * Prevents optimizing away of static variables.

```

```

1020         * gcc does not do this, so it's not needed.
1021         */
1022         break;
1023     }
1024 #if defined(__x86)
1025     if (strcmp(arg, "-Wu,-save_args") == 0) {
1026         newae(ctx->i_ae, "-msave_args");
1027         break;
1028     }
1029 #endif /* __x86 */
1030     error(arg);
1031     break;
1032 case 'X':
1033     if (strcmp(arg, "-Xa") == 0 ||
1034         strcmp(arg, "-Xt") == 0) {
1035         break;
1036     }
1037     if (strcmp(arg, "-Xs") == 0) {
1038         Xsmode(ctx->i_ae);
1039         break;
1040     }
1041     error(arg);
1042     break;
1043 case 'x':
1044     if (arglen == 1)
1045         error(arg);
1046     switch (arg[2]) {
1047     case 'a':
1048         if (strcmp(arg, "-xarch=", 7) == 0) {
1049             mflag |= xlate_xtb(ctx->i_ae, arg + 7);
1050             break;
1051         }
1052         error(arg);
1053         break;
1054     case 'b':
1055         if (strcmp(arg, "-xbuiltin=", 10) == 0) {
1056             if (strcmp(arg + 10, "%all"))
1057                 newae(ctx->i_ae, "-fbuiltin");
1058             break;
1059         }
1060         error(arg);
1061         break;
1062     case 'C':
1063         /* Accept C++ style comments -- ignore */
1064         if (strcmp(arg, "-xCC") == 0)
1065             break;
1066         error(arg);
1067         break;
1068     case 'c':
1069         if (strcmp(arg, "-xc99=%all", 10) == 0) {
1070             newae(ctx->i_ae, "-std=gnu99");
1071             break;
1072         }
1073         if (strcmp(arg, "-xc99=%none", 11) == 0) {
1074             newae(ctx->i_ae, "-std=gnu89");
1075             break;
1076         }
1077         if (strcmp(arg, "-xchip=", 7) == 0) {
1078             xlate(ctx->i_ae, arg + 7, xchip_tbl);
1079             break;
1080         }
1081         if (strcmp(arg, "-xcode=", 7) == 0) {
1082             xlate(ctx->i_ae, arg + 7, xcode_tbl);
1083             break;
1084         }
1085         if (strcmp(arg, "-xcrossfile", 11) == 0)

```

```

1086         break;
1087     error(arg);
1088     break;
1089 case 'd':
1090     if (strcmp(arg, "-xdebugformat=", 14) == 0)
1091         break;
1092     error(arg);
1093     break;
1094 case 'F':
1095     /*
1096     * Compile for mapfile reordering, or unused
1097     * section elimination, syntax can be -xF or
1098     * more complex, like -xF=%all -- ignore.
1099     */
1100     if (strcmp(arg, "-xF", 3) == 0)
1101         break;
1102     error(arg);
1103     break;
1104 case 'i':
1105     if (strcmp(arg, "-xinline", 8) == 0)
1106         /* No inlining; ignore */
1107         break;
1108     if (strcmp(arg, "-xildon") == 0 ||
1109         strcmp(arg, "-xildoff") == 0)
1110         /* No incremental linking; ignore */
1111         break;
1112     error(arg);
1113     break;
1114 #if defined(__x86)
1115 case 'm':
1116     if (strcmp(arg, "-xmodel=kernel") == 0) {
1117         newae(ctx->i_ae, "-ffreestanding");
1118         newae(ctx->i_ae, "-mno-red-zone");
1119         model = "-mcmmodel=kernel";
1120         nolibc = 1;
1121         break;
1122     }
1123     error(arg);
1124     break;
1125 #endif /* __x86 */
1126 case 'O':
1127     if (strcmp(arg, "-xO", 3) == 0) {
1128         size_t len = strlen(arg);
1129         char *s = NULL;
1130         int c = *(arg + 3);
1131         int level;
1132
1133         if (len != 4 || !isdigit(c))
1134             error(arg);
1135
1136         level = atoi(arg + 3);
1137         if (level > 5)
1138             error(arg);
1139         if (level >= 2) {
1140             /*
1141             * For gcc-3.4.x at -O2 we
1142             * need to disable optimizations
1143             * that break ON.
1144             */
1145             optim_disable(ctx->i_ae, level);
1146             /*
1147             * limit -xO3 to -O2 as well.
1148             */
1149             level = 2;
1150         }
1151         if (asprintf(&s, "-O%d", level) == -1)

```

```

1152             nomem();
1153             newae(ctx->i_ae, s);
1154             free(s);
1155             break;
1156         }
1157         error(arg);
1158         break;
1159     case 'r':
1160         if (strncmp(arg, "-xregs=", 7) == 0) {
1161             xlate(ctx->i_ae, arg + 7, xregs_tbl);
1162             break;
1163         }
1164         error(arg);
1165         break;
1166     case 's':
1167         if (strcmp(arg, "-xs") == 0 ||
1168             strcmp(arg, "-xspace") == 0 ||
1169             strcmp(arg, "-xstrconst") == 0)
1170             break;
1171         error(arg);
1172         break;
1173     case 't':
1174         if (strncmp(arg, "-xtarget=", 9) == 0) {
1175             xlate(ctx->i_ae, arg + 9, xtarget_tbl);
1176             break;
1177         }
1178         error(arg);
1179         break;
1180     case 'e':
1181     case 'h':
1182     case 'l':
1183     default:
1184         error(arg);
1185         break;
1186     }
1187     break;
1188 case 'Y':
1189     if (arglen == 1) {
1190         if ((arg = **++ctx->i_oldargv) == NULL ||
1191             *arg == '\0')
1192             error("-Y");
1193         ctx->i_oldargc--;
1194         arglen = strlen(arg + 1);
1195     } else {
1196         arg += 2;
1197     }
1198     /* Just ignore -YS,... for now */
1199     if (strncmp(arg, "S", 2) == 0)
1200         break;
1201     if (strncmp(arg, "I", 2) == 0) {
1202         char *s = strdup(arg);
1203         s[0] = '-';
1204         s[1] = 'B';
1205         newae(ctx->i_ae, s);
1206         free(s);
1207         break;
1208     }
1209     if (strncmp(arg, "I", 2) == 0) {
1210         char *s = strdup(arg);
1211         s[0] = '-';
1212         s[1] = 'I';
1213         newae(ctx->i_ae, "-nostdinc");
1214         newae(ctx->i_ae, s);
1215         free(s);
1216         break;
1217     }

```

```

1218         error(arg);
1219         break;
1220     case 'Q':
1221         /*
1222          * We could map -Qy into -Wl,-Qy etc.
1223          */
1224     default:
1225         error(arg);
1226         break;
1227     }
1228 }
1229
1230 free(nameflag);
1231
1232 /*
1233  * When compiling multiple source files in a single invocation some
1234  * compilers output objects into the current directory with
1235  * predictable and conventional names.
1236  *
1237  * We prevent any attempt to compile multiple files at once so that
1238  * any such objects created by a shadow can't escape into a later
1239  * link-edit.
1240  */
1241 if (c_files > 1 && op != CW_O_PREPROCESS) {
1242     if (c_files > 1 && (ctx->i_flags & CW_F_SHADOW) &&
1243         op != CW_O_PREPROCESS) {
1244         errx(2, "multiple source files are "
1245             "allowed only with -E or -P");
1246     }
1247 }
1248
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 }
1258
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, "-mmo-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         #endif

```

```

1282         exit(2);
1283 #endif
1284         break;
1285     case SS11:
1286         /* FALLTHROUGH */
1287     case (SS11|M32):
1288     case (SS11|M64):
1289         break;
1290     case (SS12|M32):
1291 #if defined(__sparc)
1292         /*
1293          * Need to add in further 32 bit options because with SS12
1294          * the xarch=sparcvis option can be applied to 32 or 64
1295          * bit, and so the translation table (xtbl) cannot handle
1296          * that.
1297          */
1298         newae(ctx->i_ae, "-mv8plus");
1299 #endif
1300         break;
1301     case (SS12|M64):
1302         break;
1303     default:
1304         (void) fprintf(stderr,
1305             "Incompatible -xarch= and/or -m32/-m64 options used.\n");
1306         exit(2);
1307     }

1309     if (ctx->i_flags & CW_F_SHADOW) {
1310         if (op == CW_O_PREPROCESS)
1311             exit(0);
1312         else if (op == CW_O_LINK && c_files == 0)
1313             if ((op == CW_O_LINK || op == CW_O_PREPROCESS) &&
1314                 (ctx->i_flags & CW_F_SHADOW))
1315                 exit(0);
1316     }
1317 #endif /* ! codereview */

1317     if (model != NULL)
1318         newae(ctx->i_ae, model);
1319     if (!nolibc)
1320         newae(ctx->i_ae, "-lc");
1321     if (!seen_o && (ctx->i_flags & CW_F_SHADOW)) {
1322         newae(ctx->i_ae, "-o");
1323         newae(ctx->i_ae, discard_file_name(ctx, NULL));
1324     }
1325 }

1327 static void
1328 do_cc(cw_ictx_t *ctx)
1329 {
1330     int in_output = 0, seen_o = 0, c_files = 0;
1331     int in_output = 0, seen_o = 0;
1332     cw_op_t op = CW_O_LINK;
1333     char *nameflag;

1334     if (ctx->i_flags & CW_F_PROG) {
1335         newae(ctx->i_ae, "-V");
1336         return;
1337     }

1339     if (asprintf(&nameflag, "-_%s=", ctx->i_compiler->c_name) == -1)
1340         nomen();

1342     while (--ctx->i_oldargc > 0) {
1343         char *arg = ++ctx->i_oldargv;

```

```

1344         size_t arglen = strlen(arg);
1345 #endif /* ! codereview */

1347         if (strncmp(arg, "-_CC=", 5) == 0) {
1348             newae(ctx->i_ae, strchr(arg, '=') + 1);
1349             continue;
1350         }

1352         if (*arg != '-') {
1353             if (!in_output && arglen > 2 &&
1354                 arg[arglen - 2] == '.' &&
1355                 (arg[arglen - 1] == 's' || arg[arglen - 1] == 's' ||
1356                 arg[arglen - 1] == 'c' || arg[arglen - 1] == 'i'))
1357                 c_files++;
1358         }

1359 #endif /* ! codereview */
1360     if (in_output == 0 || !(ctx->i_flags & CW_F_SHADOW)) {
1361         newae(ctx->i_ae, arg);
1362     } else {
1363         in_output = 0;
1364         newae(ctx->i_ae, discard_file_name(ctx, arg));
1365         newae(ctx->i_ae, ctx->i_discard);
1366     }
1367     continue;
1368 }
1369 switch (*(arg + 1)) {
1370 case '_':
1371     if ((strncmp(arg, nameflag, strlen(nameflag)) == 0) ||
1372         (strncmp(arg, "-_cc=", 5) == 0) ||
1373         (strncmp(arg, "-_sun=", 6) == 0)) {
1374         newae(ctx->i_ae, strchr(arg, '=') + 1);
1375     }
1376     break;

1377 case 'V':
1378     ctx->i_flags &= ~CW_F_ECHO;
1379     newae(ctx->i_ae, arg);
1380     break;
1381 case 'o':
1382     seen_o = 1;
1383     if (strlen(arg) == 2) {
1384         in_output = 1;
1385         newae(ctx->i_ae, arg);
1386     } else if (ctx->i_flags & CW_F_SHADOW) {
1387         newae(ctx->i_ae, "-o");
1388         newae(ctx->i_ae, discard_file_name(ctx, arg));
1389     } else {
1390         newae(ctx->i_ae, arg);
1391     }
1392     break;
1393 case 'c':
1394 case 'S':
1395     if (strlen(arg) == 2)
1396         op = CW_O_COMPILE;
1397     newae(ctx->i_ae, arg);
1398     break;
1399 case 'E':
1400 case 'P':
1401     if (strlen(arg) == 2)
1402         op = CW_O_PREPROCESS;
1403     /*FALLTHROUGH*/
1404     default:
1405         newae(ctx->i_ae, arg);
1406     }
1407 }

```

```

1409     free(nameflag);

1411     /* See the comment on this same code in do_gcc() */
1412     if (c_files > 1 && op != CW_O_PREPROCESS) {
1413         errx(2, "multiple source files are "
1414             "allowed only with -E or -P");
1415     }

1417     if (ctx->i_flags & CW_F_SHADOW) {
1418         if (op == CW_O_PREPROCESS)
1090     if ((op == CW_O_LINK || op == CW_O_PREPROCESS) &&
1091         (ctx->i_flags & CW_F_SHADOW))
1419             exit(0);
1420     else if (op == CW_O_LINK && c_files == 0)
1421         exit(0);
1422     }
1423 #endif /* ! codereview */

1425     if (!seen_o && (ctx->i_flags & CW_F_SHADOW) {
1426         newae(ctx->i_ae, "-o");
1427         newae(ctx->i_ae, discard_file_name(ctx, NULL));
1093         newae(ctx->i_ae, ctx->i_discard);
1428     }
1429 }

    unchanged_portion_omitted_

1506 static int
1507 reap(cw_ictx_t *ctx)
1508 {
1509     int status, ret = 0;
1510     char buf[1024];
1511     struct stat s;

1513     /*
1514     * Only wait for one specific child.
1515     */
1516     if (ctx->i_pid <= 0)
1517         return (-1);

1519     do {
1520         if (waitpid(ctx->i_pid, &status, 0) < 0) {
1521             warn("cannot reap child");
1522             return (-1);
1523         }
1524         if (status != 0) {
1525             if (WIFSIGNALED(status)) {
1526                 ret = -WTERMSIG(status);
1527                 break;
1528             } else if (WIFEXITED(status)) {
1529                 ret = WEXITSTATUS(status);
1530                 break;
1531             }
1532         }
1533     } while (!WIFEXITED(status) && !WIFSIGNALED(status));

1201     (void) unlink(ctx->i_discard);

1535     if (stat(ctx->i_stderr, &s) < 0) {
1536         warn("stat failed on child cleanup");
1537         return (-1);
1538     }
1539     if (s.st_size != 0) {
1540         FILE *f;

1542         if ((f = fopen(ctx->i_stderr, "r")) != NULL) {

```

```

1543         while (fgets(buf, sizeof(buf), f))
1544             (void) fprintf(stderr, "%s", buf);
1545         (void) fflush(stderr);
1546         (void) fclose(f);
1547     }
1548 }
1549 (void) unlink(ctx->i_stderr);
1550 free(ctx->i_stderr);

1552     /*
1553     * cc returns an error code when given -V; we want that to succeed.
1554     */
1555     if (ctx->i_flags & CW_F_PROG)
1556         return (0);

1558     return (ret);
1559 }

1561 static int
1562 exec_ctx(cw_ictx_t *ctx, int block)
1563 {
1564     if ((ctx->i_stderr = tmpnam(ctx->i_tmpdir, "cw")) == NULL) {
1232         char *file;

1234         /*
1235         * To avoid offending cc's sensibilities, the name of its output
1236         * file must end in '.o'.
1237         */
1238         if ((file = tmpnam(NULL, ".cw")) == NULL) {
1239             nomem();
1240             return (-1);
1241         }
1242         (void) strcpy(ctx->i_discard, file, MAXPATHLEN);
1243         (void) strcat(ctx->i_discard, ".o", MAXPATHLEN);
1244         free(file);

1246         if ((ctx->i_stderr = tmpnam(NULL, ".cw")) == NULL) {
1565             nomem();
1566             return (-1);
1567         }

1569         if ((ctx->i_pid = fork()) == 0) {
1570             int fd;

1572             (void) fclose(stderr);
1573             if ((fd = open(ctx->i_stderr, O_WRONLY | O_CREAT | O_EXCL,
1574                 0666)) < 0) {
1575                 err(1, "open failed for standard error");
1576             }
1577             if (dup2(fd, 2) < 0) {
1578                 err(1, "dup2 failed for standard error");
1579             }
1580             if (fd != 2)
1581                 (void) close(fd);
1582             if (freopen("/dev/fd/2", "w", stderr) == NULL) {
1583                 err(1, "freopen failed for /dev/fd/2");
1584             }

1586             prepctx(ctx);
1587             exit(invoke(ctx));
1588         }

1590         if (ctx->i_pid < 0) {
1591             err(1, "fork failed");
1592         }

```

```

1594     if (block)
1595         return (reap(ctx));

1597     return (0);
1598 }
_____unchanged_portion_omitted_____

1632 static void
1633 cleanup(cw_ictx_t *ctx)
1634 {
1635     DIR *dirp;
1636     struct dirent *dp;
1637     char buf[MAXPATHLEN];

1639     if ((dirp = opendir(ctx->i_tmpdir)) == NULL) {
1640         if (errno != ENOENT) {
1641             err(1, "couldn't open temp directory: %s",
1642                 ctx->i_tmpdir);
1643         } else {
1644             return;
1645         }
1646     }

1648     errno = 0;
1649     while ((dp = readdir(dirp)) != NULL) {
1650         (void) snprintf(buf, MAXPATHLEN, "%s/%s", ctx->i_tmpdir,
1651             dp->d_name);

1653         if (strncmp(dp->d_name, ".", strlen(dp->d_name)) == 0 ||
1654             strncmp(dp->d_name, "..", strlen(dp->d_name)) == 0)
1655             continue;

1657         if (unlink(buf) == -1)
1658             err(1, "failed to unlink temp file: %s", dp->d_name);
1659         errno = 0;
1660     }

1662     if (errno != 0) {
1663         err(1, "failed to read temporary directory: %s",
1664             ctx->i_tmpdir);
1665     }

1667     (void) closedir(dirp);
1668     if (rmdir(ctx->i_tmpdir) != 0) {
1669         err(1, "failed to unlink temporary directory: %s",
1670             ctx->i_tmpdir);
1671     }
1672 }

1674 #endif /* ! codereview */
1675 int
1676 main(int argc, char **argv)
1677 {
1678     int ch;
1679     cw_compiler_t primary = { NULL, NULL, 0 };
1680     cw_compiler_t shadows[10];
1681     int nshadows = 0;
1682     int ret = 0;
1683     boolean_t do_serial = B_FALSE;
1684     boolean_t do_exec = B_FALSE;
1685     boolean_t vflg = B_FALSE;
1686     boolean_t cflg = B_FALSE;
1687     boolean_t cflg = B_FALSE;
1688     boolean_t nflg = B_FALSE;
1689     char *tmpdir;
1690 #endif /* ! codereview */

```

```

1692     cw_ictx_t *main_ctx;

1694     static struct option longopts[] = {
1695         { "compiler", no_argument, NULL, 'c' },
1696         { "noecho", no_argument, NULL, 'n' },
1697         { "primary", required_argument, NULL, 'p' },
1698         { "shadow", required_argument, NULL, 's' },
1699         { "versions", no_argument, NULL, 'v' },
1700         { NULL, 0, NULL, 0 },
1701     };

1704     if ((main_ctx = newictx()) == NULL)
1705         nomem();

1707     while ((ch = getopt_long(argc, argv, "C", longopts, NULL)) != -1) {
1708         switch (ch) {
1709             case 'c':
1710                 cflg = B_TRUE;
1711                 break;
1712             case 'C':
1713                 Cflg = B_TRUE;
1714                 break;
1715             case 'n':
1716                 nflg = B_TRUE;
1717                 break;
1718             case 'p':
1719                 if (primary.c_path != NULL) {
1720                     warnx("Only one primary compiler may "
1721                         "be specified");
1722                     usage();
1723                 }

1725                 parse_compiler(optarg, &primary);
1726                 break;
1727             case 's':
1728                 if (nshadows >= 10)
1729                     errx(1, "May only use 10 shadows at "
1730                         "the moment");
1731                 parse_compiler(optarg, &shadows[nshadows]);
1732                 nshadows++;
1733                 break;
1734             case 'v':
1735                 vflg = B_TRUE;
1736                 break;
1737             default:
1738                 (void) fprintf(stderr, "Did you forget '--'? \n");
1739                 usage();
1740         }
1741     }

1743     if (primary.c_path == NULL) {
1744         warnx("A primary compiler must be specified");
1745         usage();
1746     }

1748     do_serial = (getenv("CW_SHADOW_SERIAL") == NULL) ? B_FALSE : B_TRUE;
1749     do_exec = (getenv("CW_NO_EXEC") == NULL) ? B_TRUE : B_FALSE;

1751     /* Leave room for argv[0] */
1752     argc -= (optind - 1);
1753     argv += (optind - 1);

1755     main_ctx->i_oldargc = argc;
1756     main_ctx->i_oldargv = argv;

```

```

1757     main_ctx->i_flags = CW_F_XLATE;
1758     if (nflg == 0)
1759         main_ctx->i_flags |= CW_F_ECHO;
1760     if (do_exec)
1761         main_ctx->i_flags |= CW_F_EXEC;
1762     if (Cflg)
1763         main_ctx->i_flags |= CW_F_CXX;
1764     main_ctx->i_compiler = &primary;

1766     if (cflg) {
1767         (void) fputs(primary.c_path, stdout);
1768     }

1770     if (vflg) {
1771         (void) printf("cw version %s\n", CW_VERSION);
1772         (void) fflush(stdout);
1773         main_ctx->i_flags &= ~CW_F_ECHO;
1774         main_ctx->i_flags |= CW_F_PROG | CW_F_EXEC;
1775         do_serial = 1;
1776     }

1778     tmpdir = getenv("TMPDIR");
1779     if (tmpdir == NULL)
1780         tmpdir = "/tmp";

1782     if (asprintf(&main_ctx->i_tmpdir, "%s/cw.XXXXXX", tmpdir) == -1)
1783         nomem();

1785     if ((main_ctx->i_tmpdir = mkdtemp(main_ctx->i_tmpdir)) == NULL)
1786         errx(1, "failed to create temporary directory");

1788 #endif /* ! codereview */
1789     ret |= exec_ctx(main_ctx, do_serial);

1791     for (int i = 0; i < nshadows; i++) {
1792         int r;
1793         cw_ictx_t *shadow_ctx;

1795         if ((shadow_ctx = newictx()) == NULL)
1796             nomem();

1798         (void) memcpy(shadow_ctx, main_ctx, sizeof (cw_ictx_t));
1314         memcpy(shadow_ctx, main_ctx, sizeof (cw_ictx_t));

1800         shadow_ctx->i_flags |= CW_F_SHADOW;
1801         shadow_ctx->i_compiler = &shadows[i];

1803         r = exec_ctx(shadow_ctx, do_serial);
1804         if (r == 0) {
1805             shadow_ctx->i_next = main_ctx->i_next;
1806             main_ctx->i_next = shadow_ctx;
1807         }
1808         ret |= r;
1809     }

1811     if (!do_serial) {
1812         cw_ictx_t *next = main_ctx;
1813         while (next != NULL) {
1814             cw_ictx_t *toreap = next;
1815             next = next->i_next;
1816             ret |= reap(toreap);
1817         }
1818     }

1820     cleanup(main_ctx);
1821 #endif /* ! codereview */

```

```

1822         return (ret);
1823     }

```