```
**********************************************************
    4425 Sat Feb 17 20:00:50 2018
new/usr/src/common/mapfiles/gen/Makefile
Review from Yuri
**********************************************************
```

```
#
# 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 2015 Igor Kozhukhov <ikozhukhov@gmail.com>
#

include         $(SRC)/Makefile.master


$(__GNUC)FILES= $(MACH)_gcc_map.noexeglobs
$(__SUNC)FILES= $(MACH)_cc_map.noexeglobs

$(__GNUC)$(BUILD64)FILES += $(MACH64)_gcc_map.noexeglobs
$(__GNU)$(BUILD64)FILES += $(MACH64)_gcc_map.noexeglobs
$(__SUNC)$(BUILD64)FILES += $(MACH64)_cc_map.noexeglobs

SYMS1=          syms.1
SYMS2=          syms.2
MAIN1=          main.1
MAIN2=          main.2

TEMPLATE1=      map.noexeglobs.1.template
TEMPLATE2=      map.noexeglobs.2.template

all install:    $(FILES)

lint:

clean:
                $(RM) $(SYMS1) $(SYMS2) $(MAIN1) $(MAIN2)

clobber:        clean
                $(RM) $(FILES)

# A number of dynamic executables have their own definitions of interfaces that
# exist in system libraries.  To prevent name-space pollution it is desirable
# to demote the interfaces within these executable to locals.  However, various
# symbols defined by the compiler drivers crt/values files need to remain
# global in any dynamic object that includes these files.  These symbols
# interpose on symbols in libc, or provide call backs for other system
```



```
# libraries.  The various compiler drivers (cc and gcc), and the crt/values
# files that these drivers are configured to include, differ between the
# various compilations environments (platform, 32/64-bit).  Therefore, the
# only means of creating a mapfile to demote symbols is to dynamically generate
# the mapfile for a specific compilation environment.
#
# Here, template mapfiles are used to generate a number of compilation specific
# mapfiles.  These mapfiles are referenced by components of the build through
# the MAPFILE.NGB macro defined in Makefile.master.  These dynamically created
# mapfiles are not delivered into the $ROOT area, and therefore are not
# delivered as packaged components of the OSNet.

$(MACH)_cc_map.noexeglobs :=    LINK =  $(LINK.c)
$(MACH64)_cc_map.noexeglobs :=  LINK =  $(LINK64.c)
$(MACH)_gcc_map.noexeglobs :=   LINK =  $(LINK.c)
$(MACH64)_gcc_map.noexeglobs := LINK =  $(LINK64.c)

# This generic target creates two dynamic executables from an empty "main"
# program.  These objects are not executed, but are analyzed to determine the
# global symbols each provides.
#
# The first executable demotes a family of known interfaces to local and allows
# all other symbol definitions to remain global.  This executables provides the
# base for discovering all symbol definitions provided by the various
# compilation environments.  The second executable demotes all symbols to
# locals.  Within both executables, some symbols remain globals (_end, _etext,
# etc.) as the link-editor has special knowledge of these symbols and their
# expected visibility requirements.  By inspecting the deferences between the
# global symbols within the two executables, a mapfile can be generated to
# ensure the symbols defined by the compilation environments files remain
# global.

%map.noexeglobs:main.c $(TEMPLATE1) $(TEMPLATE2)
        $(LINK) -o $(MAIN1) -M$(TEMPLATE1) main.c
        $(ELFDUMP) -s -N.dynsym $(MAIN1) | $(EGREP) "WEAK|GLOB" | \
            $(GREP) -v UNDEF | $(AWK) '{print $$9 }' | $(SORT) > $(SYMS1)
        $(LINK) -o $(MAIN2) -M$(TEMPLATE2) main.c
        $(ELFDUMP) -s -N.dynsym $(MAIN2) | $(EGREP) "WEAK|GLOB" | \
            $(GREP) -v UNDEF | $(AWK) '{print $$9 }' | $(SORT) > $(SYMS2)
        $(ECHO) "# GENERATED FILE - DO NOT EDIT"        >  $@
        $(GREP) MAP-HEAD $(TEMPLATE2) | \
            $(SED) -e "s/       *# MAP-HEAD//"          >> $@
        $(DIFF) $(SYMS1) $(SYMS2) | $(GREP) "^<" | \
            $(SED) -e "s/^< \(.*\)/       \1;/"         >> $@
        $(GREP) MAP-TAIL $(TEMPLATE2) | \
            $(SED) -e "s/       *# MAP-TAIL//"          >> $@
        $(RM) $(SYMS1) $(SYMS2) $(MAIN1) $(MAIN2)
```

***********************************************************
***1117 Sat Feb 17 20:00:51 2018***
***new/usr/src/lib/smbsrv/Makefile.targ***
***Review from Robert, and nits***
***********************************************************

```
#
# 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.
#
#ident  "%Z%%M% %I%     %E% SMI"

#
# Common targets for smbsrv Makefiles
#

%_ndr.c: $(NDLDIR)/%.ndl
        $(NDRGEN) -Y "$(CC)" $<

pics/%.o:       $(SRC)/common/smbsrv/%.c
        $(COMPILE.c) -o $@ $<
        $(POST_PROCESS_O)

.KEEP_STATE:

all: $(LIBS)

lint: lintcheck
```

```
**********************************************************
    6388 Sat Feb 17 20:00:52 2018
new/usr/src/tools/cw/cw.1onbld
Review from Robert, and nits
**********************************************************
```

```
.\"
.\" 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 2010 Sun Microsystems, Inc.  All rights reserved.
.\" Use is subject to license terms.
.\"
.Dd February 10, 2018
.Dt CW 1ONBLD
.Os
.Sh NAME
.Nm cw
.Nd invoke one or more compilers with argument translation
.Sh SYNOPSIS
.Nm cw
.Op Fl C
.Op Fl -versions
.Op Fl -noecho
.Fl -primary Ar compiler
.Op Fl -shadow Ar compiler ...
.Fl -
.Ar compiler args ...
.Sh DESCRIPTION
.Nm cw
is a facility for invoking one or more compilers, providing translation from
Sun style arguments as appropriate.
This allows the use of arbitrary compilers without the need to alter large
numbers of makefiles.
A mode called shadow compilation invokes multiple compilers so that warnings
and errors may be obtained from both.
See SHADOW COMPILATION for details.
This version of cw supports compilers with both Sun Studio 12 and GCC-style
command lines.
.Sh ARGUMENTS
Both the
.Fl -primary
and
.Fl -secondary
parameters take a
.Em compiler specification .
This is a comma-separated list of the form
.Ar name,executable,style
Where
.Ar name
is a name for the compiler,
```



```
.Ar executable
is the full path to the compiler executable, and
.Ar style
is the style of command-line options the compiler expects, either
.Em sun
or
.Em gnu .
.Bl -tag -width indent
.It Fl -primary Ar compiler
Specify the compiler to be used primarily (that which is used for link-editing
and pre-processing, and whos objects we deliver).
.It Fl -shadow Ar compiler
Specify a shadow compiler, which builds sources for the sake of checking code
quality and compatibility, but has its output discarded.
.It Fl -noecho
Do not echo the actual command line of any compilers invoked.
.It Fl -versions
Request from each configured primary and shadow compiler its version
information.
.It Fl C
The sources being compiled are C++.  This is necessary as it affects the
translation of compiler arguments.
.It Fl -
Arguments intended for the compilers themselves must be separated from those
of
.Nm cw
by a
.Fl - .
.It Fl _name=
.It Fl _style=
Parameters intended for the compiler be guarded with options of the form
.Fl _name=
and
.Fl _style=
Where
.Em name
and
.Em style
are those passed to
.Fl -primary
and
.Fl -shadow
this allows certain flags to be passed only to certain classes of compiler.
.Pp
For historical reasons, the
.Fl _style=
option is also translated such that a style of
.Em sun
may use the flag
.Fl _cc=
and a style of
.Em gnu
may use the flag
.Fl _gcc= ,
and when the
.Fl C
option is given and C++ is in use the style of
.Em sun
may use the flag
.Fl _CC=
and the style of
.Em gnu
may use the flag
.Fl _g++= .
.El
.Sh SHADOW COMPILATION
```

```
128 If
129 .Fl -shadow
130 compilers are specified
131 .Nm cw
132 will invoke shadow compiler, with the outputs modified (as well as any
133 translation for compiler style) as follows:
134 .Bl -enum
135 .It
136 If none of
137 .Fl c ,
138 .Fl E ,
139 .Fl P ,
140 or
141 .Fl S
142 appears in the argument list (that is, linking is attempted), the shadow
143 compilers will not be invoked.
144 This is because the objects built with that compiler which would be linked
145 have been previously discarded.
146 .It
147 If the
147 If an option of the form
148 .Fl o Ar filename
149 option was provided, with or without a separating space, it will be replaced wit
149 was provided, it will be replaced by two options of the form
150 .Fl o Ar tempfile
151 .It
152 If the option
153 .Fl o
154 was not provided,
155 .Fl o Ar tempfile
156 will be added to the end of the argument list used to invoke
157 the shadow compilers.
158 .El
159 When shadow compilation is in effect,
160 .Nm cw
161 writes to standard error each compiler’s standard error output following its
162 argument list.
163 Messages from the compilers will not be interleaved.
164 If
165 .Nm cw
166 is used to invoke the preprocessor and no output location is specified,
167 .Nm cw
168 will write to standard output the primary compiler’s standard output, and the
169 secondary compiler’s standard output will be discarded.
170 .Pp
171 Because the Sun compilers write intermediate objects to fixed
172 filenames in the current directory when instructed to compile and
173 link multiple source files via a single command line, it would be
174 unsafe to invoke more than one compiler in this fashion.
175 Therefore
176 .Nm cw
177 does not accept multiple source files unless the preprocessor is to be
178 invoked.
179 An attempt to invoke
180 .Nm cw
181 in this manner will result in an error.
182 .Sh ARGUMENT TRANSLATION
183 If the compiler to be invoked is a GNU-style C or C++ compiler, a set of
184 default flags is added to the beginning of the argument list, and the
185 remaining arguments are translated to their closest appropriate
186 semantic equivalents and passed in the same order as their
187 counterparts given to
188 .Nm cw .
189 See the comments at the head of
190 .Pa usr/src/tools/cw/cw.c
191 for a detailed list of translations.
```



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

```
**********************************************************
   45568 Sat Feb 17 20:00:54 2018
new/usr/src/tools/cw/cw.c
Review from Robert, and nits
Review from Yuri
**********************************************************
______unchanged_portion_omitted_

 338 #define COMPILER_STYLE(comp) (comp->style == SUN ? "cc" : "gcc")

 338 typedef struct {
 339         char *c_name;
 340         char *c_path;
 341         compiler_style_t c_style;
 341         char *name;
 342         char *path;
 343         compiler_style_t style;
 342 } cw_compiler_t;
______unchanged_portion_omitted_

 452 static void
 453 nomem(void)
 454 {
 455         errx(1, "out of memory");
 457         (void) errx(1, "out of memory");
 456 }
______unchanged_portion_omitted_

 522 /* ARGSUSED */
 523 static void
 524 Xamode(struct aelist __unused *h)
 526 Xamode(struct aelist *h __attribute__((__unused__)))
 525 {
 526 }
______unchanged_portion_omitted_

 614 static void
 615 do_gcc(cw_ictx_t *ctx)
 616 {
 617         int c;
 618         int pic = 0, nolibc = 0;
 619         int in_output = 0, seen_o = 0, c_files = 0;
 620         cw_op_t op = CW_O_LINK;
 621         char *model = NULL;
 622         char *nameflag;
 623         int     mflag = 0;

 625         if (ctx->i_flags & CW_F_PROG) {
 626                 newae(ctx->i_ae, "--version");
 627                 return;
 628         }

 630         newae(ctx->i_ae, "-fident");
 631         newae(ctx->i_ae, "-finline");
 632         newae(ctx->i_ae, "-fno-inline-functions");
 633         newae(ctx->i_ae, "-fno-builtin");
 634         newae(ctx->i_ae, "-fno-asm");
 635         newae(ctx->i_ae, "-fdiagnostics-show-option");
 636         newae(ctx->i_ae, "-nodefaultlibs");

 638 #if defined(__sparc)
 639         /*
 640          * The SPARC ldd and std instructions require 8-byte alignment of
 641          * their address operand.  gcc correctly uses them only when the
 642          * ABI requires 8-byte alignment; unfortunately we have a number of
 643          * pieces of buggy code that doesn't conform to the ABI.  This
```



```
 644          * flag makes gcc work more like Studio with -xmemalign=4.
 645          */
 646         newae(ctx->i_ae, "-mno-integer-ldd-std");
 647 #endif

 649         /*
 650          * This is needed because 'u' is defined
 651          * under a conditional on 'sun'.  Should
 652          * probably just remove the conditional,
 653          * or make it be dependent on '__sun'.
 654          *
 655          * -Dunix is also missing in enhanced ANSI mode
 656          */
 657         newae(ctx->i_ae, "-D__sun");

 659         if (asprintf(&nameflag, "-_%s=", ctx->i_compiler->c_name) == -1)
 661         if (asprintf(&nameflag, "-_%s=", ctx->i_compiler->name) == -1)
 660                 nomem();

 662         /*
 663          * Walk the argument list, translating as we go ..
 664          */
 665         while (--ctx->i_oldargc > 0) {
 666                 char *arg = *++ctx->i_oldargv;
 667                 size_t arglen = strlen(arg);

 669                 if (*arg == '-') {
 670                         arglen--;
 671                 } else {
 672                         /*
 673                          * Discard inline files that gcc doesn't grok
 674                          */
 675                         if (!in_output && arglen > 3 &&
 676                             strcmp(arg + arglen - 3, ".il") == 0)
 677                                 continue;

 679                         if (!in_output && arglen > 2 &&
 680                             arg[arglen - 2] == '.' &&
 681                             (arg[arglen - 1] == 'S' || arg[arglen - 1] == 's' ||
 682                             arg[arglen - 1] == 'c' || arg[arglen - 1] == 'i'))
 683                                 c_files++;

 685                         /*
 686                          * Otherwise, filenames and partial arguments
 687                          * are passed through for gcc to chew on.  However,
 688                          * output is always discarded for the secondary
 689                          * compiler.
 690                          */
 691                         if ((ctx->i_flags & CW_F_SHADOW) && in_output)
 692                                 newae(ctx->i_ae, ctx->i_discard);
 693                         else
 694                                 newae(ctx->i_ae, arg);
 695                         in_output = 0;
 696                         continue;
 697                 }

 699                 if (ctx->i_flags & CW_F_CXX) {
 700                         if (strncmp(arg, "-_g++=", 6) == 0) {
 701                                 newae(ctx->i_ae, strchr(arg, '=') + 1);
 702                                 continue;
 703                         }
 704                         if (strncmp(arg, "-compat=", 8) == 0) {
 705                                 /* discard -compat=4 and -compat=5 */
 706                                 continue;
 707                         }
 708                         if (strcmp(arg, "-Qoption") == 0) {
```

```
709                                 /* discard -Qoption and its two arguments */
710                                 if (ctx->i_oldargc < 3)
711                                         error(arg);
712                                 ctx->i_oldargc -= 2;
713                                 ctx->i_oldargv += 2;
714                                 continue;
715                         }
716                         if (strcmp(arg, "-xwe") == 0) {
717                                 /* turn warnings into errors */
718                                 newae(ctx->i_ae, "-Werror");
719                                 continue;
720                         }
721                         if (strcmp(arg, "-noex") == 0) {
722                                 /* no exceptions */
723                                 newae(ctx->i_ae, "-fno-exceptions");
724                                 /* no run time type descriptor information */
725                                 newae(ctx->i_ae, "-fno-rtti");
726                                 continue;
727                         }
728                         if (strcmp(arg, "-pic") == 0) {
729                                 newae(ctx->i_ae, "-fpic");
730                                 pic = 1;
731                                 continue;
732                         }
733                         if (strcmp(arg, "-PIC") == 0) {
734                                 newae(ctx->i_ae, "-fPIC");
735                                 pic = 1;
736                                 continue;
737                         }
738                         if (strcmp(arg, "-norunpath") == 0) {
739                                 /* gcc has no corresponding option */
740                                 continue;
741                         }
742                         if (strcmp(arg, "-nolib") == 0) {
743                                 /* -nodefaultlibs is on by default */
744                                 nolibc = 1;
745                                 continue;
746                         }
747 #if defined(__sparc)
748                         if (strcmp(arg, "-cg92") == 0) {
749                                 mflag |= xlate_xtb(ctx->i_ae, "v8");
750                                 xlate(ctx->i_ae, "super", xchip_tbl);
751                                 continue;
752                         }
753 #endif  /* __sparc */
754                 }

756                 switch ((c = arg[1])) {
757                 case '_':
758                         if ((strncmp(arg, nameflag, strlen(nameflag)) == 0) ||
759                             (strncmp(arg, "-_gcc=", 6) == 0) ||
760                             (strncmp(arg, "-_gnu=", 6) == 0)) {
761                                 newae(ctx->i_ae, strchr(arg, '=') + 1);
762                         }
763                         break;
764                 case '#':
765                         if (arglen == 1) {
766                                 newae(ctx->i_ae, "-v");
767                                 break;
768                         }
769                         error(arg);
770                         break;
771                 case 'g':
772                         newae(ctx->i_ae, "-gdwarf-2");
773                         break;
774                 case 'E':
```



```
775                         if (arglen == 1) {
776                                 newae(ctx->i_ae, "-xc");
777                                 newae(ctx->i_ae, arg);
778                                 op = CW_O_PREPROCESS;
779                                 nolibc = 1;
780                                 break;
781                         }
782                         error(arg);
783                         break;
784                 case 'c':
785                 case 'S':
786                         if (arglen == 1) {
787                                 op = CW_O_COMPILE;
788                                 nolibc = 1;
789                         }
790                         /* FALLTHROUGH */
791                 case 'C':
792                 case 'H':
793                 case 'p':
794                         if (arglen == 1) {
795                                 newae(ctx->i_ae, arg);
796                                 break;
797                         }
798                         error(arg);
799                         break;
800                 case 'A':
801                 case 'h':
802                 case 'I':
803                 case 'i':
804                 case 'L':
805                 case 'l':
806                 case 'R':
807                 case 'U':
808                 case 'u':
809                 case 'w':
810                         newae(ctx->i_ae, arg);
811                         break;
812                 case 'o':
813                         seen_o = 1;
814                         if (arglen == 1) {
815                                 in_output = 1;
816                                 newae(ctx->i_ae, arg);
817                         } else if (ctx->i_flags & CW_F_SHADOW) {
818                                 newae(ctx->i_ae, "-o");
819                                 newae(ctx->i_ae, ctx->i_discard);
820                         } else {
821                                 newae(ctx->i_ae, arg);
822                         }
823                         break;
824                 case 'D':
825                         newae(ctx->i_ae, arg);
826                         /*
827                          * XXX  Clearly a hack ... do we need _KADB too?
828                          */
829                         if (strcmp(arg, "-D_KERNEL") == 0 ||
830                             strcmp(arg, "-D_BOOT") == 0)
831                                 newae(ctx->i_ae, "-ffreestanding");
832                         break;
833                 case 'd':
834                         if (arglen == 2) {
835                                 if (strcmp(arg, "-dy") == 0) {
836                                         newae(ctx->i_ae, "-Wl,-dy");
837                                         break;
838                                 }
839                                 if (strcmp(arg, "-dn") == 0) {
840                                         newae(ctx->i_ae, "-Wl,-dn");
```

```
841                                     break;
842                             }
843                     }
844                     if (strcmp(arg, "-dalign") == 0) {
845                             /*
846                              * -dalign forces alignment in some cases;
847                              * gcc does not need any flag to do this.
848                              */
849                             break;
850                     }
851                     error(arg);
852                     break;
853             case 'e':
854                     if (strcmp(arg,
855                         "-erroff=E_EMPTY_TRANSLATION_UNIT") == 0) {
856                             /*
857                              * Accept but ignore this -- gcc doesn't
858                              * seem to complain about empty translation
859                              * units
860                              */
861                             break;
862                     }
863                     /* XX64 -- ignore all -erroff= options, for now */
864                     if (strncmp(arg, "-erroff=", 8) == 0)
865                             break;
866                     if (strcmp(arg, "-errtags=yes") == 0) {
867                             warnings(ctx->i_ae);
868                             break;
869                     }
870                     if (strcmp(arg, "-errwarn=%all") == 0) {
871                             newae(ctx->i_ae, "-Werror");
872                             break;
873                     }
874                     error(arg);
875                     break;
876             case 'f':
877                     if (strcmp(arg, "-flags") == 0) {
878                             newae(ctx->i_ae, "--help");
879                             break;
880                     }
881                     if (strncmp(arg, "-features=zla", 13) == 0) {
882                             /*
883                              * Accept but ignore this -- gcc allows
884                              * zero length arrays.
885                              */
886                             break;
887                     }
888                     error(arg);
889                     break;
890             case 'G':
891                     newae(ctx->i_ae, "-shared");
892                     nolibc = 1;
893                     break;
894             case 'k':
895                     if (strcmp(arg, "-keeptmp") == 0) {
896                             newae(ctx->i_ae, "-save-temps");
897                             break;
898                     }
899                     error(arg);
900                     break;
901             case 'K':
902                     if (arglen == 1) {
903                             if ((arg = *++ctx->i_oldargv) == NULL ||
904                                 *arg == '\0')
905                                     error("-K");
906                             ctx->i_oldargc--;
```



```
907                     } else {
908                             arg += 2;
909                     }
910                     if (strcmp(arg, "pic") == 0) {
911                             newae(ctx->i_ae, "-fpic");
912                             pic = 1;
913                             break;
914                     }
915                     if (strcmp(arg, "PIC") == 0) {
916                             newae(ctx->i_ae, "-fPIC");
917                             pic = 1;
918                             break;
919                     }
920                     error("-K");
921                     break;
922             case 'm':
923                     if (strcmp(arg, "-mt") == 0) {
924                             newae(ctx->i_ae, "-D_REENTRANT");
925                             break;
926                     }
927                     if (strcmp(arg, "-m64") == 0) {
928                             newae(ctx->i_ae, "-m64");
929 #if defined(__x86)
930                             newae(ctx->i_ae, "-mtune=opteron");
931 #endif
932                             mflag |= M64;
933                             break;
934                     }
935                     if (strcmp(arg, "-m32") == 0) {
936                             newae(ctx->i_ae, "-m32");
937                             mflag |= M32;
938                             break;
939                     }
940                     error(arg);
941                     break;
942             case 'B':       /* linker options */
943             case 'M':
944             case 'z':
945                     {
946                             char *opt;
947                             size_t len;
948                             char *s;

950                             if (arglen == 1) {
951                                     opt = *++ctx->i_oldargv;
952                                     if (opt == NULL || *opt == '\0')
953                                             error(arg);
954                                     ctx->i_oldargc--;
955                             } else {
956                                     opt = arg + 2;
957                             }
958                             len = strlen(opt) + 7;
959                             if ((s = malloc(len)) == NULL)
960                                     nomem();
961                             (void) snprintf(s, len, "-Wl,-%c%s", c, opt);
962                             newae(ctx->i_ae, s);
963                             free(s);
964                     }
965                     break;
966             case 'n':
967                     if (strcmp(arg, "-noqueue") == 0) {
968                             /*
969                              * Horrid license server stuff - n/a
970                              */
971                             break;
972                     }
```

```
 973                         error(arg);
 974                         break;
 975                 case 'O':
 976                         if (arglen == 1) {
 977                                 newae(ctx->i_ae, "-O");
 978                                 break;
 979                         }
 980                         error(arg);
 981                         break;
 982                 case 'P':
 983                         /*
 984                          * We could do '-E -o filename.i', but that's hard,
 985                          * and we don't need it for the case that's triggering
 986                          * this addition.  We'll require the user to specify
 987                          * -o in the Makefile.  If they don't they'll find out
 988                          * in a hurry.
 989                          */
 990                         newae(ctx->i_ae, "-E");
 991                         op = CW_O_PREPROCESS;
 992                         nolibc = 1;
 993                         break;
 994                 case 'q':
 995                         if (strcmp(arg, "-qp") == 0) {
 996                                 newae(ctx->i_ae, "-p");
 997                                 break;
 998                         }
 999                         error(arg);
1000                         break;
1001                 case 's':
1002                         if (arglen == 1) {
1003                                 newae(ctx->i_ae, "-Wl,-s");
1004                                 break;
1005                         }
1006                         error(arg);
1007                         break;
1008                 case 't':
1009                         if (arglen == 1) {
1010                                 newae(ctx->i_ae, "-Wl,-t");
1011                                 break;
1012                         }
1013                         error(arg);
1014                         break;
1015                 case 'V':
1016                         if (arglen == 1) {
1017                                 ctx->i_flags &= ~CW_F_ECHO;
1018                                 newae(ctx->i_ae, "--version");
1019                                 break;
1020                         }
1021                         error(arg);
1022                         break;
1023                 case 'v':
1024                         if (arglen == 1) {
1025                                 warnings(ctx->i_ae);
1026                                 break;
1027                         }
1028                         error(arg);
1029                         break;
1030                 case 'W':
1031                         if (strncmp(arg, "-Wp,-xc99", 9) == 0) {
1032                                 /*
1033                                  * gcc's preprocessor will accept c99
1034                                  * regardless, so accept and ignore.
1035                                  */
1036                                 break;
1037                         }
1038                         if (strncmp(arg, "-Wa,", 4) == 0 ||
```



```
1039                             strncmp(arg, "-Wp,", 4) == 0 ||
1040                             strncmp(arg, "-Wl,", 4) == 0) {
1041                                 newae(ctx->i_ae, arg);
1042                                 break;
1043                         }
1044                         if (strcmp(arg, "-W0,-xc99=pragma") == 0) {
1045                                 /* (undocumented) enables _Pragma */
1046                                 break;
1047                         }
1048                         if (strcmp(arg, "-W0,-xc99=%none") == 0) {
1049                                 /*
1050                                  * This is a polite way of saying
1051                                  * "no c99 constructs allowed!"
1052                                  * For now, just accept and ignore this.
1053                                  */
1054                                 break;
1055                         }
1056                         if (strcmp(arg, "-W0,-noglobal") == 0 ||
1057                             strcmp(arg, "-W0,-xglobalstatic") == 0) {
1058                                 /*
1059                                  * gcc doesn't prefix local symbols
1060                                  * in debug mode, so this is not needed.
1061                                  */
1062                                 break;
1063                         }
1064                         if (strcmp(arg, "-W0,-Lt") == 0) {
1065                                 /*
1066                                  * Generate tests at the top of loops.
1067                                  * There is no direct gcc equivalent, ignore.
1068                                  */
1069                                 break;
1070                         }
1071                         if (strcmp(arg, "-W0,-xdbggen=no%usedonly") == 0) {
1072                                 newae(ctx->i_ae,
1073                                     "-fno-eliminate-unused-debug-symbols");
1074                                 newae(ctx->i_ae,
1075                                     "-fno-eliminate-unused-debug-types");
1076                                 break;
1077                         }
1078                         if (strcmp(arg, "-W2,-xwrap_int") == 0) {
1079                                 /*
1080                                  * Use the legacy behaviour (pre-SS11)
1081                                  * for integer wrapping.
1082                                  * gcc does not need this.
1083                                  */
1084                                 break;
1085                         }
1086                         if (strcmp(arg, "-W2,-Rcond_elim") == 0) {
1087                                 /*
1088                                  * Elimination and expansion of conditionals;
1089                                  * gcc has no direct equivalent.
1090                                  */
1091                                 break;
1092                         }
1093                         if (strcmp(arg, "-Wd,-xsafe=unboundsym") == 0) {
1094                                 /*
1095                                  * Prevents optimizing away checks for
1096                                  * unbound weak symbol addresses.  gcc does
1097                                  * not do this, so it's not needed.
1098                                  */
1099                                 break;
1100                         }
1101                         if (strncmp(arg, "-Wc,-xcode=", 11) == 0) {
1102                                 xlate(ctx->i_ae, arg + 11, xcode_tbl);
1103                                 if (strncmp(arg + 11, "pic", 3) == 0)
1104                                         pic = 1;
```

```
1105                            break;
1106                    }
1107                    if (strncmp(arg, "-Wc,-Qiselect", 13) == 0) {
1108                            /*
1109                             * Prevents insertion of register symbols.
1110                             * gcc doesn't do this, so ignore it.
1111                             */
1112                            break;
1113                    }
1114                    if (strcmp(arg, "-Wc,-Qassembler-ounrefsym=0") == 0) {
1115                            /*
1116                             * Prevents optimizing away of static variables.
1117                             * gcc does not do this, so it's not needed.
1118                             */
1119                            break;
1120                    }
1121 #if defined(__x86)
1122                    if (strcmp(arg, "-Wu,-xmodel=kernel") == 0) {
1123                            newae(ctx->i_ae, "-ffreestanding");
1124                            newae(ctx->i_ae, "-mno-red-zone");
1125                            model = "-mcmodel=kernel";
1126                            nolibc = 1;
1127                            break;
1128                    }
1129                    if (strcmp(arg, "-Wu,-save_args") == 0) {
1130                            newae(ctx->i_ae, "-msave-args");
1131                            break;
1132                    }
1133 #endif  /* __x86 */
1134                    error(arg);
1135                    break;
1136            case 'X':
1137                    if (strcmp(arg, "-Xa") == 0 ||
1138                        strcmp(arg, "-Xt") == 0) {
1139                            Xamode(ctx->i_ae);
1140                            break;
1141                    }
1142                    if (strcmp(arg, "-Xc") == 0) {
1143                            Xcmode(ctx->i_ae);
1144                            break;
1145                    }
1146                    if (strcmp(arg, "-Xs") == 0) {
1147                            Xsmode(ctx->i_ae);
1148                            break;
1149                    }
1150                    error(arg);
1151                    break;
1152            case 'x':
1153                    if (arglen == 1)
1154                            error(arg);
1155                    switch (arg[2]) {
1156 #if defined(__x86)
1157                    case '3':
1158                            if (strcmp(arg, "-x386") == 0) {
1159                                    newae(ctx->i_ae, "-march=i386");
1160                                    break;
1161                            }
1162                            error(arg);
1163                            break;
1164                    case '4':
1165                            if (strcmp(arg, "-x486") == 0) {
1166                                    newae(ctx->i_ae, "-march=i486");
1167                                    break;
1168                            }
1169                            error(arg);
1170                            break;
```



```
1171 #endif  /* __x86 */
1172                    case 'a':
1173                            if (strncmp(arg, "-xarch=", 7) == 0) {
1174                                    mflag |= xlate_xtb(ctx->i_ae, arg + 7);
1175                                    break;
1176                            }
1177                            error(arg);
1178                            break;
1179                    case 'b':
1180                            if (strncmp(arg, "-xbuiltin=", 10) == 0) {
1181                                    if (strcmp(arg + 10, "%all"))
1182                                            newae(ctx->i_ae, "-fbuiltin");
1183                                    break;
1184                            }
1185                            error(arg);
1186                            break;
1187                    case 'C':
1188                            /* Accept C++ style comments -- ignore */
1189                            if (strcmp(arg, "-xCC") == 0)
1190                                    break;
1191                            error(arg);
1192                            break;
1193                    case 'c':
1194                            if (strncmp(arg, "-xc99=%all", 10) == 0) {
1195                                    newae(ctx->i_ae, "-std=gnu99");
1196                                    break;
1197                            }
1198                            if (strncmp(arg, "-xc99=%none", 11) == 0) {
1199                                    newae(ctx->i_ae, "-std=gnu89");
1200                                    break;
1201                            }
1202                            if (strncmp(arg, "-xchip=", 7) == 0) {
1203                                    xlate(ctx->i_ae, arg + 7, xchip_tbl);
1204                                    break;
1205                            }
1206                            if (strncmp(arg, "-xcode=", 7) == 0) {
1207                                    xlate(ctx->i_ae, arg + 7, xcode_tbl);
1208                                    if (strncmp(arg + 7, "pic", 3) == 0)
1209                                            pic = 1;
1210                                    break;
1211                            }
1212                            if (strncmp(arg, "-xcache=", 8) == 0)
1213                                    break;
1214                            if (strncmp(arg, "-xcrossfile", 11) == 0)
1215                                    break;
1216                            error(arg);
1217                            break;
1218                    case 'd':
1219                            if (strcmp(arg, "-xdepend") == 0)
1220                                    break;
1221                            if (strncmp(arg, "-xdebugformat=", 14) == 0)
1222                                    break;
1223                            error(arg);
1224                            break;
1225                    case 'F':
1226                            /*
1227                             * Compile for mapfile reordering, or unused
1228                             * section elimination, syntax can be -xF or
1229                             * more complex, like -xF=%all -- ignore.
1230                             */
1231                            if (strncmp(arg, "-xF", 3) == 0)
1232                                    break;
1233                            error(arg);
1234                            break;
1235                    case 'i':
1236                            if (strncmp(arg, "-xinline", 8) == 0)
```

```
1237                                    /* No inlining; ignore */
1238                                    break;
1239                            if (strcmp(arg, "-xildon") == 0 ||
1240                                strcmp(arg, "-xildoff") == 0)
1241                                    /* No incremental linking; ignore */
1242                                    break;
1243                            error(arg);
1244                            break;
1245 #if defined(__x86)
1246                    case 'm':
1247                            if (strcmp(arg, "-xmodel=kernel") == 0) {
1248                                    newae(ctx->i_ae, "-ffreestanding");
1249                                    newae(ctx->i_ae, "-mno-red-zone");
1250                                    model = "-mcmodel=kernel";
1251                                    nolibc = 1;
1252                                    break;
1253                            }
1254                            error(arg);
1255                            break;
1256 #endif  /* __x86 */
1257                    case 'M':
1258                            if (strcmp(arg, "-xM") == 0) {
1259                                    newae(ctx->i_ae, "-M");
1260                                    break;
1261                            }
1262                            if (strcmp(arg, "-xM1") == 0) {
1263                                    newae(ctx->i_ae, "-MM");
1264                                    break;
1265                            }
1266                            error(arg);
1267                            break;
1268                    case 'n':
1269                            if (strcmp(arg, "-xnolib") == 0) {
1270                                    nolibc = 1;
1271                                    break;
1272                            }
1273                            error(arg);
1274                            break;
1275                    case 'O':
1276                            if (strncmp(arg, "-xO", 3) == 0) {
1277                                    size_t len = strlen(arg);
1278                                    char *s;
1279                                    int c = *(arg + 3);
1280                                    int level;

1282                                    if (len != 4 || !isdigit(c))
1283                                            error(arg);

1285                                    if ((s = malloc(len)) == NULL)
1286                                            nomem();

1288                                    level = atoi(arg + 3);
1289                                    if (level > 5)
1290                                            error(arg);
1291                                    if (level >= 2) {
1292                                            /*
1293                                             * For gcc-3.4.x at -O2 we
1294                                             * need to disable optimizations
1295                                             * that break ON.
1296                                             */
1297                                            optim_disable(ctx->i_ae, level);
1298                                            /*
1299                                             * limit -xO3 to -O2 as well.
1300                                             */
1301                                            level = 2;
1302                                    }
```



```
1303                                    (void) snprintf(s, len, "-O%d", level);
1304                                    newae(ctx->i_ae, s);
1305                                    free(s);
1306                                    break;
1307                            }
1308                            error(arg);
1309                            break;
1310                    case 'p':
1311                            if (strcmp(arg, "-xpentium") == 0) {
1312                                    newae(ctx->i_ae, "-march=pentium");
1313                                    break;
1314                            }
1315                            if (strcmp(arg, "-xpg") == 0) {
1316                                    newae(ctx->i_ae, "-pg");
1317                                    break;
1318                            }
1319                            error(arg);
1320                            break;
1321                    case 'r':
1322                            if (strncmp(arg, "-xregs=", 7) == 0) {
1323                                    xlate(ctx->i_ae, arg + 7, xregs_tbl);
1324                                    break;
1325                            }
1326                            error(arg);
1327                            break;
1328                    case 's':
1329                            if (strcmp(arg, "-xs") == 0 ||
1330                                strcmp(arg, "-xspace") == 0 ||
1331                                strcmp(arg, "-xstrconst") == 0)
1332                                    break;
1333                            error(arg);
1334                            break;
1335                    case 't':
1336                            if (strcmp(arg, "-xtransition") == 0) {
1337                                    newae(ctx->i_ae, "-Wtransition");
1338                                    break;
1339                            }
1340                            if (strcmp(arg, "-xtrigraphs=yes") == 0) {
1341                                    newae(ctx->i_ae, "-trigraphs");
1342                                    break;
1343                            }
1344                            if (strcmp(arg, "-xtrigraphs=no") == 0) {
1345                                    newae(ctx->i_ae, "-notrigraphs");
1346                                    break;
1347                            }
1348                            if (strncmp(arg, "-xtarget=", 9) == 0) {
1349                                    xlate(ctx->i_ae, arg + 9, xtarget_tbl);
1350                                    break;
1351                            }
1352                            error(arg);
1353                            break;
1354                    case 'e':
1355                    case 'h':
1356                    case 'l':
1357                    default:
1358                            error(arg);
1359                            break;
1360                    }
1361                    break;
1362            case 'Y':
1363                    if (arglen == 1) {
1364                            if ((arg = *++ctx->i_oldargv) == NULL ||
1365                                *arg == '\0')
1366                                    error("-Y");
1367                            ctx->i_oldargc--;
1368                            arglen = strlen(arg + 1);
```

```
1369                        } else {
1370                                arg += 2;
1371                        }
1372                        /* Just ignore -YS,... for now */
1373                        if (strncmp(arg, "S,", 2) == 0)
1374                                break;
1375                        if (strncmp(arg, "l,", 2) == 0) {
1376                                char *s = strdup(arg);
1377                                s[0] = '-';
1378                                s[1] = 'B';
1379                                newae(ctx->i_ae, s);
1380                                free(s);
1381                                break;
1382                        }
1383                        if (strncmp(arg, "I,", 2) == 0) {
1384                                char *s = strdup(arg);
1385                                s[0] = '-';
1386                                s[1] = 'I';
1387                                newae(ctx->i_ae, "-nostdinc");
1388                                newae(ctx->i_ae, s);
1389                                free(s);
1390                                break;
1391                        }
1392                        error(arg);
1393                        break;
1394                case 'Q':
1395                        /*
1396                         * We could map -Qy into -Wl,-Qy etc.
1397                         */
1398                default:
1399                        error(arg);
1400                        break;
1401                }
1402        }

1404        free(nameflag);

1406        if (c_files > 1 && (ctx->i_flags & CW_F_SHADOW) &&
1407            op != CW_O_PREPROCESS) {
1408                errx(2, "multiple source files are "
1410                (void) errx(2, "multiple source files are "
1409                    "allowed only with -E or -P");
1410        }

1412        /*
1413         * Make sure that we do not have any unintended interactions between
1414         * the xarch options passed in and the version of the Studio compiler
1415         * used.
1416         */
1417        if ((mflag & (SS11|SS12)) == (SS11|SS12)) {
1418                errx(2,
1420                (void) errx(2,
1419                    "Conflicting \"-xarch=\" flags (both Studio 11 and 12)\n");
1420        }

1422        switch (mflag) {
1423        case 0:
1424                /* FALLTHROUGH */
1425        case M32:
1426 #if defined(__sparc)
1427                /*
1428                 * Only -m32 is defined and so put in the missing xarch
1429                 * translation.
1430                 */
1431                newae(ctx->i_ae, "-mcpu=v8");
1432                newae(ctx->i_ae, "-mno-v8plus");
```



```
1433 #endif
1434                 break;
1435         case M64:
1436 #if defined(__sparc)
1437                 /*
1438                  * Only -m64 is defined and so put in the missing xarch
1439                  * translation.
1440                  */
1441                 newae(ctx->i_ae, "-mcpu=v9");
1442 #endif
1443                 break;
1444         case SS12:
1445 #if defined(__sparc)
1446                 /* no -m32/-m64 flag used - this is an error for sparc builds */
1447                 (void) fprintf(stderr, "No -m32/-m64 flag defined\n");
1448                 exit(2);
1449 #endif
1450                 break;
1451         case SS11:
1452                 /* FALLTHROUGH */
1453         case (SS11|M32):
1454         case (SS11|M64):
1455                 break;
1456         case (SS12|M32):
1457 #if defined(__sparc)
1458                 /*
1459                  * Need to add in further 32 bit options because with SS12
1460                  * the xarch=sparcvis option can be applied to 32 or 64
1461                  * bit, and so the translatation table (xtbl) cannot handle
1462                  * that.
1463                  */
1464                 newae(ctx->i_ae, "-mv8plus");
1465 #endif
1466                 break;
1467         case (SS12|M64):
1468                 break;
1469         default:
1470                 (void) fprintf(stderr,
1471                     "Incompatible -xarch= and/or -m32/-m64 options used.\n");
1472                 exit(2);
1473         }
1474         if (op == CW_O_LINK && (ctx->i_flags & CW_F_SHADOW))
1475                 exit(0);

1477         if (model && !pic)
1478                 newae(ctx->i_ae, model);
1479         if (!nolibc)
1480                 newae(ctx->i_ae, "-lc");
1481         if (!seen_o && (ctx->i_flags & CW_F_SHADOW)) {
1482                 newae(ctx->i_ae, "-o");
1483                 newae(ctx->i_ae, ctx->i_discard);
1484         }
1485 }

1487 static void
1488 do_cc(cw_ictx_t *ctx)
1489 {
1490         int in_output = 0, seen_o = 0;
1491         cw_op_t op = CW_O_LINK;
1492         char *nameflag;

1494         if (ctx->i_flags & CW_F_PROG) {
1495                 newae(ctx->i_ae, "-V");
1496                 return;
1497         }
```

```
1499          if (asprintf(&nameflag, "-_%s=", ctx->i_compiler->c_name) == -1)
1501          if (asprintf(&nameflag, "-_%s=", ctx->i_compiler->name) == -1)
1500                  nomem();

1502          while (--ctx->i_oldargc > 0) {
1503                  char *arg = *++ctx->i_oldargv;

1505                  if (strncmp(arg, "-_CC=", 5) == 0) {
1506                          newae(ctx->i_ae, strchr(arg, '=') + 1);
1507                          continue;
1508                  }

1510                  if (*arg != '-') {
1511                          if (in_output == 0 || !(ctx->i_flags & CW_F_SHADOW)) {
1512                                  newae(ctx->i_ae, arg);
1513                          } else {
1514                                  in_output = 0;
1515                                  newae(ctx->i_ae, ctx->i_discard);
1516                          }
1517                          continue;
1518                  }
1519                  switch (*(arg + 1)) {
1520                  case '_':
1521                          if ((strncmp(arg, nameflag, strlen(nameflag)) == 0) ||
1522                              (strncmp(arg, "-_cc=", 5) == 0) ||
1523                              (strncmp(arg, "-_sun=", 6) == 0)) {
1524                                  newae(ctx->i_ae, strchr(arg, '=') + 1);
1525                          }
1526                          break;

1528                  case 'V':
1529                          ctx->i_flags &= ~CW_F_ECHO;
1530                          newae(ctx->i_ae, arg);
1531                          break;
1532                  case 'o':
1533                          seen_o = 1;
1534                          if (strlen(arg) == 2) {
1535                                  in_output = 1;
1536                                  newae(ctx->i_ae, arg);
1537                          } else if (ctx->i_flags & CW_F_SHADOW) {
1538                                  newae(ctx->i_ae, "-o");
1539                                  newae(ctx->i_ae, ctx->i_discard);
1540                          } else {
1541                                  newae(ctx->i_ae, arg);
1542                          }
1543                          break;
1544                  case 'c':
1545                  case 'S':
1546                          if (strlen(arg) == 2)
1547                                  op = CW_O_COMPILE;
1548                          newae(ctx->i_ae, arg);
1549                          break;
1550                  case 'E':
1551                  case 'P':
1552                          if (strlen(arg) == 2)
1553                                  op = CW_O_PREPROCESS;
1554                  /*FALLTHROUGH*/
1555                  default:
1556                          newae(ctx->i_ae, arg);
1557                  }
1558          }

1560          free(nameflag);

1562          if ((op == CW_O_LINK || op == CW_O_PREPROCESS) &&
1563              (ctx->i_flags & CW_F_SHADOW))
```



```
1564                  exit(0);

1566          if (!seen_o && (ctx->i_flags & CW_F_SHADOW)) {
1567                  newae(ctx->i_ae, "-o");
1568                  newae(ctx->i_ae, ctx->i_discard);
1569          }
1570  }

1572  static void
1573  prepctx(cw_ictx_t *ctx)
1574  {
1575          newae(ctx->i_ae, ctx->i_compiler->c_path);
1577          newae(ctx->i_ae, ctx->i_compiler->path);

1577          if (ctx->i_flags & CW_F_PROG) {
1578                  (void) printf("%s: %s\n", (ctx->i_flags & CW_F_SHADOW) ?
1579                      "shadow" : "primary", ctx->i_compiler->c_path);
1581                      "shadow" : "primary", ctx->i_compiler->path);
1580                  (void) fflush(stdout);
1581          }

1583          if (!(ctx->i_flags & CW_F_XLATE))
1584                  return;

1586          switch (ctx->i_compiler->c_style) {
1588          switch (ctx->i_compiler->style) {
1587          case SUN:
1588                  do_cc(ctx);
1589                  break;
1590          case GNU:
1591                  do_gcc(ctx);
1592                  break;
1593          }
1594  }
______unchanged_portion_omitted_

1753  static int
1754  parse_compiler(const char *spec, cw_compiler_t *compiler)
1755  {
1756          char *tspec, *token;

1758          if ((tspec = strdup(spec)) == NULL)
1759                  nomem();
1761                  err(1, "out of memory");

1761          if ((token = strsep(&tspec, ",")) == NULL)
1762                  errx(1, "Compiler is missing a name: %s", spec);
1763          compiler->c_name = token;
1765          compiler->name = token;

1765          if ((token = strsep(&tspec, ",")) == NULL)
1766                  errx(1, "Compiler is missing a path: %s", spec);
1767          compiler->c_path = token;
1769          compiler->path = token;

1769          if ((token = strsep(&tspec, ",")) == NULL)
1770                  errx(1, "Compiler is missing a style: %s", spec);

1772          if ((strcasecmp(token, "gnu") == 0) ||
1773              (strcasecmp(token, "gcc") == 0))
1774                  compiler->c_style = GNU;
1776                  compiler->style = GNU;
1775          else if ((strcasecmp(token, "sun") == 0) ||
1776              (strcasecmp(token, "cc") == 0))
1777                  compiler->c_style = SUN;
1779                  compiler->style = SUN;
```

```
1778         else
1779                 errx(1, "unknown compiler style: %s", token);

1781         if (tspec != NULL)
1782                 errx(1, "Excess tokens in compiler: %s", spec);

1784         return (0);
1785 }

1787 int
1788 main(int argc, char **argv)
1789 {
1790         int ch;
1791         cw_compiler_t primary = { NULL, NULL, 0 };
1792         cw_compiler_t shadows[10];
1793         int nshadows = 0;
1794         int ret = 0;
1795         boolean_t do_serial = B_FALSE;
1796         boolean_t do_exec = B_FALSE;
1797         boolean_t vflg = B_FALSE;
1798         boolean_t Cflg = B_FALSE;
1799         boolean_t cflg = B_FALSE;
1800         boolean_t nflg = B_FALSE;

1802         cw_ictx_t *main_ctx;
1804         cw_ictx_t *main_ctx = newictx();

1804         static struct option longopts[] = {
1805                 { "compiler", no_argument, NULL, 'c' },
1806                 { "noecho", no_argument, NULL, 'n' },
1808                 { "noecho", no_argument, NULL ,'n' },
1807                 { "primary", required_argument, NULL, 'p' },
1808                 { "shadow", required_argument, NULL, 's' },
1809                 { "versions", no_argument, NULL, 'v' },
1810                 { NULL, 0, NULL, 0 },
1811         };


1814         if ((main_ctx = newictx()) == NULL)
1815                 nomem();


1818 #endif /* ! codereview */
1819         while ((ch = getopt_long(argc, argv, "C", longopts, NULL)) != -1) {
1820                 switch (ch) {
1821                 case 'c':
1822                         cflg = B_TRUE;
1823                         break;
1824                 case 'C':
1825                         Cflg = B_TRUE;
1826                         break;
1827                 case 'n':
1828                         nflg = B_TRUE;
1829                         break;
1830                 case 'p':
1831                         if (primary.c_path != NULL) {
1832                                 warnx("Only one primary compiler may "
1833                                     "be specified");
1815                         if (primary.path != NULL) {
1816                                 warnx("Only one primary compiler may be specifie
1834                                 usage();
1835                         }

1837                         if (parse_compiler(optarg, &primary) != 0)
1838                                 errx(1, "Couldn't parse %s as a compiler spec",
1839                                     optarg);
```



```
1821                                 errx(1, "Couldn't parse %s as a compiler spec",
1840                         break;
1841                 case 's':
1842                         if (nshadows >= 10)
1843                                 errx(1, "May only use 10 shadows at "
1844                                     "the moment");
1825                                 errx(1, "May only use 10 shadows at the moment")
1845                         if (parse_compiler(optarg, &shadows[nshadows]) != 0)
1846                                 errx(1, "Couldn't parse %s as a compiler spec",
1847                                     optarg);
1827                                 errx(1, "Couldn't parse %s as a compiler spec",
1848                         nshadows++;
1849                         break;
1850                 case 'v':
1851                         vflg = B_TRUE;
1852                         break;
1853                 default:
1854                         (void) fprintf(stderr, "Did you forget '--'?\n");
1834                         fprintf(stderr, "Did you forget '--'?\n");
1855                         usage();
1856                 }
1857         }

1859         if (primary.c_path == NULL) {
1839         if (primary.path == NULL) {
1860                 warnx("A primary compiler must be specified");
1861                 usage();
1862         }

1864         do_serial = (getenv("CW_SHADOW_SERIAL") == NULL) ? B_FALSE : B_TRUE;
1865         do_exec = (getenv("CW_NO_EXEC") == NULL) ? B_TRUE : B_FALSE;

1867         /* Leave room for argv[0] */
1868         argc -= (optind - 1);
1869         argv += (optind - 1);

1851         if (main_ctx == NULL)
1852                 nomem();

1871         main_ctx->i_oldargc = argc;
1872         main_ctx->i_oldargv = argv;
1873         main_ctx->i_flags = CW_F_XLATE;
1874         if (nflg == 0)
1875                 main_ctx->i_flags |= CW_F_ECHO;
1876         if (do_exec)
1877                 main_ctx->i_flags |= CW_F_EXEC;
1878         if (Cflg)
1879                 main_ctx->i_flags |= CW_F_CXX;
1880         main_ctx->i_compiler = &primary;

1882         if (cflg) {
1883                 (void) fputs(primary.c_path, stdout);
1866                 fputs(primary.path, stdout);
1884         }

1886         if (vflg) {
1887                 (void) printf("cw version %s\n", CW_VERSION);
1888                 (void) fflush(stdout);
1889                 main_ctx->i_flags &= ~CW_F_ECHO;
1890                 main_ctx->i_flags |= CW_F_PROG|CW_F_EXEC;
1891                 do_serial = 1;
1892         }

1894         ret |= exec_ctx(main_ctx, do_serial);

1896         for (int i = 0; i < nshadows; i++) {
```

```
1897                    cw_ictx_t *shadow_ctx;
1880                    cw_ictx_t *shadow_ctx = newictx();

1899                    if ((shadow_ctx = newictx()) == NULL)
1882                    if (shadow_ctx == NULL)
1900                            nomem();

1902                    memcpy(shadow_ctx, main_ctx, sizeof (cw_ictx_t));

1904                    shadow_ctx->i_flags |= CW_F_SHADOW;
1905                    shadow_ctx->i_compiler = &shadows[i];

1907                    /* XXX: Would be nice to run these parallel, too */
1908                    ret |= exec_ctx(shadow_ctx, 1);
1909            }

1911            if (!do_serial)
1912                    ret |= reap(main_ctx);

1914            return (ret);
1915 }
```

_____***unchanged_portion_omitted_***