```
**********************************************************
    6241 Wed Oct 17 17:00:27 2018
new/usr/src/tools/cw/cw.1onbld
9899 cw(1onbld) should shadow more compilation
9888 cw shouldn't use __unused
**********************************************************
```

```
.\"
.\" 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.
.\"
.\" Copyright 2018 Joyent, Inc.
.\"
.Dd September 4, 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 all of them.
See
.Sx 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 -shadow
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
If
.Fl -shadow
compilers are specified
.Nm cw
will invoke each shadow compiler, with the outputs modified (as well as any
translation for compiler style) as follows:
.Bl -enum
.It
If
.Nm cw
is invoked to link-edit without compilation (the input files are all objects),
the shadow compiler is not invoked.
If neither of
.Fl c ,
.Fl S
appears in the argument list (that is, linking is attempted or only the
pre-processor is invoked), the shadow compilers will not be invoked.
.It
If the
.Fl o Ar filename
option was provided, with or without a separating space, it will be replaced wit
.Fl o Ar tempfile
.It
If the option
.Fl o
was not provided,
.Fl o Ar tempfile
will be added to the end of the argument list used to invoke
the shadow compilers.
.El
When shadow compilation is in effect,
.Nm cw
writes to standard error each compiler's standard error output following its
argument list.
Messages from the compilers will not be interleaved.
If
.Nm cw
is used to invoke the preprocessor and no output location is specified,
.Nm cw
will write to standard output the primary compiler's standard output.
.Pp
Because the Sun compilers write intermediate objects to fixed
filenames in the current directory when instructed to compile and
link multiple source files via a single command line, it would be
unsafe to invoke more than one compiler in this fashion.
Therefore
.Nm cw
does not accept multiple source files unless the preprocessor is to be
invoked.
An attempt to invoke
.Nm cw
in this manner will result in an error.
.Sh ARGUMENT TRANSLATION
If the compiler to be invoked is a GNU-style C or C++ compiler, a set of
default flags is added to the beginning of the argument list, and the
remaining arguments are translated to their closest appropriate
semantic equivalents and passed in the same order as their
counterparts given to
.Nm cw .
See the comments at the head of
.Pa usr/src/tools/cw/cw.c
```



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

```
**********************************************************
   42328 Wed Oct 17 17:00:35 2018
new/usr/src/tools/cw/cw.c
9899 cw(1onbld) should shadow more compilation
9888 cw shouldn't use __unused
**********************************************************
______unchanged_portion_omitted_

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

 482 /* ARGSUSED */
 483 static void
 484 Xamode(struct aelist __unused *h)
 485 {
 486 }

 482 static void
 483 Xsmode(struct aelist *h)
 484 {
 485         static int xsonce;

 487         if (xsonce++)
 488                 return;

 490         newae(h, "-traditional");
 491         newae(h, "-traditional-cpp");
 492 }
______unchanged_portion_omitted_

 556 /*
 557  * The compiler wants the output file to end in appropriate extension.  If
 558  * we're generating a name from whole cloth (path == NULL), we assume that
 559  * extension to be .o, otherwise we match the extension of the caller.
 560  */
 561 static char *
 562 discard_file_name(const char *path)
 563 {
 564         char *ret, *ext, *file;

 566         if (path == NULL) {
 567                 ext = ".o";
 568         } else {
 569                 ext = strrchr(path, '.');
 570         }

 572         if ((ret = calloc(MAXPATHLEN, sizeof (char))) == NULL)
 573                 nomem();

 575         if ((file = tempnam(NULL, ".cw")) == NULL)
 576                 nomem();

 578         (void) strlcpy(ret, file, MAXPATHLEN);
 579         if (ext != NULL)
 580                 (void) strlcat(ret, ext, MAXPATHLEN);
```



```
 581         free(file);
 582         return (ret);
 583 }

 585 #endif /* ! codereview */
 586 static void
 587 do_gcc(cw_ictx_t *ctx)
 588 {
 589         int c;
 590         int nolibc = 0;
 591         int in_output = 0, seen_o = 0, c_files = 0;
 592         cw_op_t op = CW_O_LINK;
 593         char *model = NULL;
 594         char *nameflag;
 595         int mflag = 0;

 597         if (ctx->i_flags & CW_F_PROG) {
 598                 newae(ctx->i_ae, "--version");
 599                 return;
 600         }

 602         newae(ctx->i_ae, "-fident");
 603         newae(ctx->i_ae, "-finline");
 604         newae(ctx->i_ae, "-fno-inline-functions");
 605         newae(ctx->i_ae, "-fno-builtin");
 606         newae(ctx->i_ae, "-fno-asm");
 607         newae(ctx->i_ae, "-fdiagnostics-show-option");
 608         newae(ctx->i_ae, "-nodefaultlibs");

 610 #if defined(__sparc)
 611         /*
 612          * The SPARC ldd and std instructions require 8-byte alignment of
 613          * their address operand.  gcc correctly uses them only when the
 614          * ABI requires 8-byte alignment; unfortunately we have a number of
 615          * pieces of buggy code that doesn't conform to the ABI.  This
 616          * flag makes gcc work more like Studio with -xmemalign=4.
 617          */
 618         newae(ctx->i_ae, "-mno-integer-ldd-std");
 619 #endif

 621         /*
 622          * This is needed because 'u' is defined
 623          * under a conditional on 'sun'.  Should
 624          * probably just remove the conditional,
 625          * or make it be dependent on '__sun'.
 626          *
 627          * -Dunix is also missing in enhanced ANSI mode
 628          */
 629         newae(ctx->i_ae, "-D__sun");

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

 634         /*
 635          * Walk the argument list, translating as we go ..
 636          */
 637         while (--ctx->i_oldargc > 0) {
 638                 char *arg = *++ctx->i_oldargv;
 639                 size_t arglen = strlen(arg);

 641                 if (*arg == '-') {
 642                         arglen--;
 643                 } else {
 644                         /*
 645                          * Discard inline files that gcc doesn't grok
 646                          */
```

```
647                         if (!in_output && arglen > 3 &&
648                             strcmp(arg + arglen - 3, ".il") == 0)
649                                 continue;

651                         if (!in_output && arglen > 2 &&
652                             arg[arglen - 2] == '.' &&
653                             (arg[arglen - 1] == 'S' || arg[arglen - 1] == 's' ||
654                             arg[arglen - 1] == 'c' || arg[arglen - 1] == 'i'))
655                                 c_files++;

657                         /*
658                          * Otherwise, filenames and partial arguments
659                          * are passed through for gcc to chew on.  However,
660                          * output is always discarded for the secondary
661                          * compiler.
662                          */
663                         if ((ctx->i_flags & CW_F_SHADOW) && in_output) {
664                                 ctx->i_discard = discard_file_name(arg);

666                                 if (ctx->i_discard == NULL)
667                                         nomem();
562                         if ((ctx->i_flags & CW_F_SHADOW) && in_output)
668                                 newae(ctx->i_ae, ctx->i_discard);
669                         } else {
564                         else
670                                 newae(ctx->i_ae, arg);
671                         }
672 #endif /* ! codereview */
673                         in_output = 0;
674                         continue;
675                 }

677                 if (ctx->i_flags & CW_F_CXX) {
678                         if (strncmp(arg, "-_g++=", 6) == 0) {
679                                 newae(ctx->i_ae, strchr(arg, '=') + 1);
680                                 continue;
681                         }
682                         if (strncmp(arg, "-compat=", 8) == 0) {
683                                 /* discard -compat=4 and -compat=5 */
684                                 continue;
685                         }
686                         if (strcmp(arg, "-Qoption") == 0) {
687                                 /* discard -Qoption and its two arguments */
688                                 if (ctx->i_oldargc < 3)
689                                         error(arg);
690                                 ctx->i_oldargc -= 2;
691                                 ctx->i_oldargv += 2;
692                                 continue;
693                         }
694                         if (strcmp(arg, "-xwe") == 0) {
695                                 /* turn warnings into errors */
696                                 newae(ctx->i_ae, "-Werror");
697                                 continue;
698                         }
699                         if (strcmp(arg, "-norunpath") == 0) {
700                                 /* gcc has no corresponding option */
701                                 continue;
702                         }
703                         if (strcmp(arg, "-nolib") == 0) {
704                                 /* -nodefaultlibs is on by default */
705                                 nolibc = 1;
706                                 continue;
707                         }
708 #if defined(__sparc)
709                         if (strcmp(arg, "-cg92") == 0) {
710                                 mflag |= xlate_xtb(ctx->i_ae, "v8");
```



```
711                                 xlate(ctx->i_ae, "super", xchip_tbl);
712                                 continue;
713                         }
714 #endif  /* __sparc */
715                 }

717                 switch ((c = arg[1])) {
718                 case '_':
719                         if ((strncmp(arg, nameflag, strlen(nameflag)) == 0) ||
720                             (strncmp(arg, "-_gcc=", 6) == 0) ||
721                             (strncmp(arg, "-_gnu=", 6) == 0)) {
722                                 newae(ctx->i_ae, strchr(arg, '=') + 1);
723                         }
724                         break;
725                 case '#':
726                         if (arglen == 1) {
727                                 newae(ctx->i_ae, "-v");
728                                 break;
729                         }
730                         error(arg);
731                         break;
732                 case 'f':
733                         if ((strcmp(arg, "-fpic") == 0) ||
734                             (strcmp(arg, "-fPIC") == 0)) {
735                                 newae(ctx->i_ae, arg);
736                                 break;
737                         }
738                         error(arg);
739                         break;
740                 case 'g':
741                         newae(ctx->i_ae, "-gdwarf-2");
742                         break;
743                 case 'E':
744                         if (arglen == 1) {
745                                 newae(ctx->i_ae, "-xc");
746                                 newae(ctx->i_ae, arg);
747                                 op = CW_O_PREPROCESS;
748                                 nolibc = 1;
749                                 break;
750                         }
751                         error(arg);
752                         break;
753                 case 'c':
754                 case 'S':
755                         if (arglen == 1) {
756                                 op = CW_O_COMPILE;
757                                 nolibc = 1;
758                         }
759                         /* FALLTHROUGH */
760                 case 'C':
761                 case 'H':
762                 case 'p':
763                         if (arglen == 1) {
764                                 newae(ctx->i_ae, arg);
765                                 break;
766                         }
767                         error(arg);
768                         break;
769                 case 'A':
770                 case 'h':
771                 case 'I':
772                 case 'i':
773                 case 'L':
774                 case 'l':
775                 case 'R':
776                 case 'U':
```

```
777                 case 'u':
778                 case 'w':
779                         newae(ctx->i_ae, arg);
780                         break;
781                 case 'o':
782                         seen_o = 1;
783                         if (arglen == 1) {
784                                 in_output = 1;
785                                 newae(ctx->i_ae, arg);
786                         } else if (ctx->i_flags & CW_F_SHADOW) {
787                                 newae(ctx->i_ae, "-o");
788                                 ctx->i_discard = discard_file_name(arg);
789                                 if (ctx->i_discard == NULL)
790                                         nomem();

792 #endif /* ! codereview */
793                                 newae(ctx->i_ae, ctx->i_discard);
794                         } else {
795                                 newae(ctx->i_ae, arg);
796                         }
797                         break;
798                 case 'D':
799                         newae(ctx->i_ae, arg);
800                         /*
801                          * XXX  Clearly a hack ... do we need _KADB too?
802                          */
803                         if (strcmp(arg, "-D_KERNEL") == 0 ||
804                             strcmp(arg, "-D_BOOT") == 0)
805                                 newae(ctx->i_ae, "-ffreestanding");
806                         break;
807                 case 'd':
808                         if (arglen == 2) {
809                                 if (strcmp(arg, "-dy") == 0) {
810                                         newae(ctx->i_ae, "-Wl,-dy");
811                                         break;
812                                 }
813                                 if (strcmp(arg, "-dn") == 0) {
814                                         newae(ctx->i_ae, "-Wl,-dn");
815                                         break;
816                                 }
817                         }
818                         if (strcmp(arg, "-dalign") == 0) {
819                                 /*
820                                  * -dalign forces alignment in some cases;
821                                  * gcc does not need any flag to do this.
822                                  */
823                                 break;
824                         }
825                         error(arg);
826                         break;
827                 case 'e':
828                         if (strcmp(arg,
829                             "-erroff=E_EMPTY_TRANSLATION_UNIT") == 0) {
830                                 /*
831                                  * Accept but ignore this -- gcc doesn't
832                                  * seem to complain about empty translation
833                                  * units
834                                  */
835                                 break;
836                         }
837                         /* XX64 -- ignore all -erroff= options, for now */
838                         if (strncmp(arg, "-erroff=", 8) == 0)
839                                 break;
840                         if (strcmp(arg, "-errtags=yes") == 0) {
841                                 warnings(ctx->i_ae);
842                                 break;
```



```
843                         }
844                         if (strcmp(arg, "-errwarn=%all") == 0) {
845                                 newae(ctx->i_ae, "-Werror");
846                                 break;
847                         }
848                         error(arg);
849                         break;
850                 case 'G':
851                         newae(ctx->i_ae, "-shared");
852                         nolibc = 1;
853                         break;
854                 case 'k':
855                         if (strcmp(arg, "-keeptmp") == 0) {
856                                 newae(ctx->i_ae, "-save-temps");
857                                 break;
858                         }
859                         error(arg);
860                         break;
861                 case 'm':
862                         if (strcmp(arg, "-mt") == 0) {
863                                 newae(ctx->i_ae, "-D_REENTRANT");
864                                 break;
865                         }
866                         if (strcmp(arg, "-m64") == 0) {
867                                 newae(ctx->i_ae, "-m64");
868 #if defined(__x86)
869                                 newae(ctx->i_ae, "-mtune=opteron");
870 #endif
871                                 mflag |= M64;
872                                 break;
873                         }
874                         if (strcmp(arg, "-m32") == 0) {
875                                 newae(ctx->i_ae, "-m32");
876                                 mflag |= M32;
877                                 break;
878                         }
879                         error(arg);
880                         break;
881                 case 'B':       /* linker options */
882                 case 'M':
883                 case 'z':
884                         {
885                                 char *opt;
886                                 size_t len;
887                                 char *s;

889                                 if (arglen == 1) {
890                                         opt = *++ctx->i_oldargv;
891                                         if (opt == NULL || *opt == '\0')
892                                                 error(arg);
893                                         ctx->i_oldargc--;
894                                 } else {
895                                         opt = arg + 2;
896                                 }
897                                 len = strlen(opt) + 7;
898                                 if ((s = malloc(len)) == NULL)
899                                         nomem();
900                                 (void) snprintf(s, len, "-Wl,-%c%s", c, opt);
901                                 newae(ctx->i_ae, s);
902                                 free(s);
903                         }
904                         break;
905                 case 'O':
906                         if (arglen == 1) {
907                                 newae(ctx->i_ae, "-O");
908                                 break;
```

```
909                         }
910                         error(arg);
911                         break;
912                 case 'P':
913                         /*
914                          * We could do '-E -o filename.i', but that's hard,
915                          * and we don't need it for the case that's triggering
916                          * this addition.  We'll require the user to specify
917                          * -o in the Makefile.  If they don't they'll find out
918                          * in a hurry.
919                          */
920                         newae(ctx->i_ae, "-E");
921                         op = CW_O_PREPROCESS;
922                         nolibc = 1;
923                         break;
924                 case 's':
925                         if (arglen == 1) {
926                                 newae(ctx->i_ae, "-Wl,-s");
927                                 break;
928                         }
929                         error(arg);
930                         break;
931                 case 't':
932                         if (arglen == 1) {
933                                 newae(ctx->i_ae, "-Wl,-t");
934                                 break;
935                         }
936                         error(arg);
937                         break;
938                 case 'V':
939                         if (arglen == 1) {
940                                 ctx->i_flags &= ~CW_F_ECHO;
941                                 newae(ctx->i_ae, "--version");
942                                 break;
943                         }
944                         error(arg);
945                         break;
946                 case 'v':
947                         if (arglen == 1) {
948                                 warnings(ctx->i_ae);
949                                 break;
950                         }
951                         error(arg);
952                         break;
953                 case 'W':
954                         if (strncmp(arg, "-Wp,-xc99", 9) == 0) {
955                                 /*
956                                  * gcc's preprocessor will accept c99
957                                  * regardless, so accept and ignore.
958                                  */
959                                 break;
960                         }
961                         if (strncmp(arg, "-Wa,", 4) == 0 ||
962                             strncmp(arg, "-Wp,", 4) == 0 ||
963                             strncmp(arg, "-Wl,", 4) == 0) {
964                                 newae(ctx->i_ae, arg);
965                                 break;
966                         }
967                         if (strcmp(arg, "-W0,-noglobal") == 0 ||
968                             strcmp(arg, "-W0,-xglobalstatic") == 0) {
969                                 /*
970                                  * gcc doesn't prefix local symbols
971                                  * in debug mode, so this is not needed.
972                                  */
973                                 break;
974                         }
```



```
975                         if (strcmp(arg, "-W0,-Lt") == 0) {
976                                 /*
977                                  * Generate tests at the top of loops.
978                                  * There is no direct gcc equivalent, ignore.
979                                  */
980                                 break;
981                         }
982                         if (strcmp(arg, "-W0,-xdbggen=no%usedonly") == 0) {
983                                 newae(ctx->i_ae,
984                                     "-fno-eliminate-unused-debug-symbols");
985                                 newae(ctx->i_ae,
986                                     "-fno-eliminate-unused-debug-types");
987                                 break;
988                         }
989                         if (strcmp(arg, "-W2,-xwrap_int") == 0) {
990                                 /*
991                                  * Use the legacy behaviour (pre-SS11)
992                                  * for integer wrapping.
993                                  * gcc does not need this.
994                                  */
995                                 break;
996                         }
997                         if (strcmp(arg, "-Wd,-xsafe=unboundsym") == 0) {
998                                 /*
999                                  * Prevents optimizing away checks for
1000                                 * unbound weak symbol addresses.  gcc does
1001                                 * not do this, so it's not needed.
1002                                 */
1003                                break;
1004                        }
1005                        if (strncmp(arg, "-Wc,-xcode=", 11) == 0) {
1006                                xlate(ctx->i_ae, arg + 11, xcode_tbl);
1007                                break;
1008                        }
1009                        if (strncmp(arg, "-Wc,-Qiselect", 13) == 0) {
1010                                /*
1011                                 * Prevents insertion of register symbols.
1012                                 * gcc doesn't do this, so ignore it.
1013                                 */
1014                                break;
1015                        }
1016                        if (strcmp(arg, "-Wc,-Qassembler-ounrefsym=0") == 0) {
1017                                /*
1018                                 * Prevents optimizing away of static variables.
1019                                 * gcc does not do this, so it's not needed.
1020                                 */
1021                                break;
1022                        }
1023 #if defined(__x86)
1024                        if (strcmp(arg, "-Wu,-save_args") == 0) {
1025                                newae(ctx->i_ae, "-msave-args");
1026                                break;
1027                        }
1028 #endif  /* __x86 */
1029                        error(arg);
1030                        break;
1031                case 'X':
1032                        if (strcmp(arg, "-Xa") == 0 ||
1033                            strcmp(arg, "-Xt") == 0) {
 566                                Xamode(ctx->i_ae);
1034                                break;
1035                        }
1036                        if (strcmp(arg, "-Xs") == 0) {
1037                                Xsmode(ctx->i_ae);
1038                                break;
1039                        }
```

```
1040                            error(arg);
1041                            break;
1042                    case 'x':
1043                            if (arglen == 1)
1044                                    error(arg);
1045                            switch (arg[2]) {
1046                            case 'a':
1047                                    if (strncmp(arg, "-xarch=", 7) == 0) {
1048                                            mflag |= xlate_xtb(ctx->i_ae, arg + 7);
1049                                            break;
1050                                    }
1051                                    error(arg);
1052                                    break;
1053                            case 'b':
1054                                    if (strncmp(arg, "-xbuiltin=", 10) == 0) {
1055                                            if (strcmp(arg + 10, "%all"))
1056                                                    newae(ctx->i_ae, "-fbuiltin");
1057                                            break;
1058                                    }
1059                                    error(arg);
1060                                    break;
1061                            case 'C':
1062                                    /* Accept C++ style comments -- ignore */
1063                                    if (strcmp(arg, "-xCC") == 0)
1064                                            break;
1065                                    error(arg);
1066                                    break;
1067                            case 'c':
1068                                    if (strncmp(arg, "-xc99=%all", 10) == 0) {
1069                                            newae(ctx->i_ae, "-std=gnu99");
1070                                            break;
1071                                    }
1072                                    if (strncmp(arg, "-xc99=%none", 11) == 0) {
1073                                            newae(ctx->i_ae, "-std=gnu89");
1074                                            break;
1075                                    }
1076                                    if (strncmp(arg, "-xchip=", 7) == 0) {
1077                                            xlate(ctx->i_ae, arg + 7, xchip_tbl);
1078                                            break;
1079                                    }
1080                                    if (strncmp(arg, "-xcode=", 7) == 0) {
1081                                            xlate(ctx->i_ae, arg + 7, xcode_tbl);
1082                                            break;
1083                                    }
1084                                    if (strncmp(arg, "-xcrossfile", 11) == 0)
1085                                            break;
1086                                    error(arg);
1087                                    break;
1088                            case 'd':
1089                                    if (strncmp(arg, "-xdebugformat=", 14) == 0)
1090                                            break;
1091                                    error(arg);
1092                                    break;
1093                            case 'F':
1094                                    /*
1095                                     * Compile for mapfile reordering, or unused
1096                                     * section elimination, syntax can be -xF or
1097                                     * more complex, like -xF=%all -- ignore.
1098                                     */
1099                                    if (strncmp(arg, "-xF", 3) == 0)
1100                                            break;
1101                                    error(arg);
1102                                    break;
1103                            case 'i':
1104                                    if (strncmp(arg, "-xinline", 8) == 0)
1105                                            /* No inlining; ignore */
```



```
1106                                            break;
1107                                    if (strcmp(arg, "-xildon") == 0 ||
1108                                        strcmp(arg, "-xildoff") == 0)
1109                                            /* No incremental linking; ignore */
1110                                            break;
1111                                    error(arg);
1112                                    break;
1113 #if defined(__x86)
1114                            case 'm':
1115                                    if (strcmp(arg, "-xmodel=kernel") == 0) {
1116                                            newae(ctx->i_ae, "-ffreestanding");
1117                                            newae(ctx->i_ae, "-mno-red-zone");
1118                                            model = "-mcmodel=kernel";
1119                                            nolibc = 1;
1120                                            break;
1121                                    }
1122                                    error(arg);
1123                                    break;
1124 #endif  /* __x86 */
1125                            case 'O':
1126                                    if (strncmp(arg, "-xO", 3) == 0) {
1127                                            size_t len = strlen(arg);
1128                                            char *s = NULL;
1129                                            int c = *(arg + 3);
1130                                            int level;
1131
1132                                            if (len != 4 || !isdigit(c))
1133                                                    error(arg);
1134
1135                                            level = atoi(arg + 3);
1136                                            if (level > 5)
1137                                                    error(arg);
1138                                            if (level >= 2) {
1139                                                    /*
1140                                                     * For gcc-3.4.x at -O2 we
1141                                                     * need to disable optimizations
1142                                                     * that break ON.
1143                                                     */
1144                                                    optim_disable(ctx->i_ae, level);
1145                                                    /*
1146                                                     * limit -xO3 to -O2 as well.
1147                                                     */
1148                                                    level = 2;
1149                                            }
1150                                            if (asprintf(&s, "-O%d", level) == -1)
1151                                                    nomem();
1152                                            newae(ctx->i_ae, s);
1153                                            free(s);
1154                                            break;
1155                                    }
1156                                    error(arg);
1157                                    break;
1158                            case 'r':
1159                                    if (strncmp(arg, "-xregs=", 7) == 0) {
1160                                            xlate(ctx->i_ae, arg + 7, xregs_tbl);
1161                                            break;
1162                                    }
1163                                    error(arg);
1164                                    break;
1165                            case 's':
1166                                    if (strcmp(arg, "-xs") == 0 ||
1167                                        strcmp(arg, "-xspace") == 0 ||
1168                                        strcmp(arg, "-xstrconst") == 0)
1169                                            break;
1170                                    error(arg);
1171                                    break;
```

```
1172                        case 't':
1173                                if (strncmp(arg, "-xtarget=", 9) == 0) {
1174                                        xlate(ctx->i_ae, arg + 9, xtarget_tbl);
1175                                        break;
1176                                }
1177                                error(arg);
1178                                break;
1179                        case 'e':
1180                        case 'h':
1181                        case 'l':
1182                        default:
1183                                error(arg);
1184                                break;
1185                        }
1186                        break;
1187                case 'Y':
1188                        if (arglen == 1) {
1189                                if ((arg = *++ctx->i_oldargv) == NULL ||
1190                                    *arg == '\0')
1191                                        error("-Y");
1192                                ctx->i_oldargc--;
1193                                arglen = strlen(arg + 1);
1194                        } else {
1195                                arg += 2;
1196                        }
1197                        /* Just ignore -YS,... for now */
1198                        if (strncmp(arg, "S,", 2) == 0)
1199                                break;
1200                        if (strncmp(arg, "l,", 2) == 0) {
1201                                char *s = strdup(arg);
1202                                s[0] = '-';
1203                                s[1] = 'B';
1204                                newae(ctx->i_ae, s);
1205                                free(s);
1206                                break;
1207                        }
1208                        if (strncmp(arg, "I,", 2) == 0) {
1209                                char *s = strdup(arg);
1210                                s[0] = '-';
1211                                s[1] = 'I';
1212                                newae(ctx->i_ae, "-nostdinc");
1213                                newae(ctx->i_ae, s);
1214                                free(s);
1215                                break;
1216                        }
1217                        error(arg);
1218                        break;
1219                case 'Q':
1220                        /*
1221                         * We could map -Qy into -Wl,-Qy etc.
1222                         */
1223                default:
1224                        error(arg);
1225                        break;
1226                }
1227        }

1229        free(nameflag);

1231        /*
1232         * When compiling multiple source files in a single invocation some
1233         * compilers output objects into the current directory with
1234         * predictable and conventional names.
1235         *
1236         * We prevent any attempt to compile multiple files at once so that
1237         * any such objects created by a shadow can't escape into a later
```



```
1238         * link-edit.
1239         */
1240        if (c_files > 1 && op != CW_O_PREPROCESS) {
 764        if (c_files > 1 && (ctx->i_flags & CW_F_SHADOW) &&
 765            op != CW_O_PREPROCESS) {
1241                errx(2, "multiple source files are "
1242                    "allowed only with -E or -P");
1243        }

1245        /*
1246         * Make sure that we do not have any unintended interactions between
1247         * the xarch options passed in and the version of the Studio compiler
1248         * used.
1249         */
1250        if ((mflag & (SS11|SS12)) == (SS11|SS12)) {
1251                errx(2,
1252                    "Conflicting \"-xarch=\" flags (both Studio 11 and 12)\n");
1253        }

1255        switch (mflag) {
1256        case 0:
1257                /* FALLTHROUGH */
1258        case M32:
1259 #if defined(__sparc)
1260                /*
1261                 * Only -m32 is defined and so put in the missing xarch
1262                 * translation.
1263                 */
1264                newae(ctx->i_ae, "-mcpu=v8");
1265                newae(ctx->i_ae, "-mno-v8plus");
1266 #endif
1267                break;
1268        case M64:
1269 #if defined(__sparc)
1270                /*
1271                 * Only -m64 is defined and so put in the missing xarch
1272                 * translation.
1273                 */
1274                newae(ctx->i_ae, "-mcpu=v9");
1275 #endif
1276                break;
1277        case SS12:
1278 #if defined(__sparc)
1279                /* no -m32/-m64 flag used - this is an error for sparc builds */
1280                (void) fprintf(stderr, "No -m32/-m64 flag defined\n");
1281                exit(2);
1282 #endif
1283                break;
1284        case SS11:
1285                /* FALLTHROUGH */
1286        case (SS11|M32):
1287        case (SS11|M64):
1288                break;
1289        case (SS12|M32):
1290 #if defined(__sparc)
1291                /*
1292                 * Need to add in further 32 bit options because with SS12
1293                 * the xarch=sparcvis option can be applied to 32 or 64
1294                 * bit, and so the translatation table (xtbl) cannot handle
1295                 * that.
1296                 */
1297                newae(ctx->i_ae, "-mv8plus");
1298 #endif
1299                break;
1300        case (SS12|M64):
1301                break;
```

```
1302         default:
1303                 (void) fprintf(stderr,
1304                     "Incompatible -xarch= and/or -m32/-m64 options used.\n");
1305                 exit(2);
1306         }

1308         if (ctx->i_flags & CW_F_SHADOW) {
1309                 if (op == CW_O_PREPROCESS)
1310                         exit(0);
1311                 else if (op == CW_O_LINK && c_files == 0)
 833         if ((op == CW_O_LINK || op == CW_O_PREPROCESS) &&
 834             (ctx->i_flags & CW_F_SHADOW))
1312                         exit(0);
1313         }
1314 #endif /* ! codereview */

1316         if (model != NULL)
1317                 newae(ctx->i_ae, model);
1318         if (!nolibc)
1319                 newae(ctx->i_ae, "-lc");
1320         if (!seen_o && (ctx->i_flags & CW_F_SHADOW)) {
1321                 ctx->i_discard = discard_file_name(NULL);

1323                 if (ctx->i_discard == NULL)
1324                         nomem();
1325 #endif /* ! codereview */
1326                 newae(ctx->i_ae, "-o");
1327                 newae(ctx->i_ae, ctx->i_discard);
1328         }
1329 }

1331 static void
1332 do_cc(cw_ictx_t *ctx)
1333 {
1334         int in_output = 0, seen_o = 0, c_files = 0;
 836         int in_output = 0, seen_o = 0;
1335         cw_op_t op = CW_O_LINK;
1336         char *nameflag;

1338         if (ctx->i_flags & CW_F_PROG) {
1339                 newae(ctx->i_ae, "-V");
1340                 return;
1341         }

1343         if (asprintf(&nameflag, "-_%s=", ctx->i_compiler->c_name) == -1)
1344                 nomem();

1346         while (--ctx->i_oldargc > 0) {
1347                 char *arg = *++ctx->i_oldargv;
1348                 size_t arglen = strlen(arg);
1349 #endif /* ! codereview */

1351                 if (strncmp(arg, "-_CC=", 5) == 0) {
1352                         newae(ctx->i_ae, strchr(arg, '=') + 1);
1353                         continue;
1354                 }

1356                 if (*arg != '-') {
1357                         if (!in_output && arglen > 2 &&
1358                             arg[arglen - 2] == '.' &&
1359                             (arg[arglen - 1] == 'S' || arg[arglen - 1] == 's' ||
1360                             arg[arglen - 1] == 'c' || arg[arglen - 1] == 'i'))
1361                                 c_files++;

1363 #endif /* ! codereview */
1364                         if (in_output == 0 || !(ctx->i_flags & CW_F_SHADOW)) {
```



```
1365                                 newae(ctx->i_ae, arg);
1366                         } else {
1367                                 in_output = 0;
1368                                 ctx->i_discard = discard_file_name(arg);
1369                                 if (ctx->i_discard == NULL)
1370                                         nomem();
1371 #endif /* ! codereview */
1372                                 newae(ctx->i_ae, ctx->i_discard);
1373                         }
1374                         continue;
1375                 }
1376                 switch (*(arg + 1)) {
1377                 case '_':
1378                         if ((strncmp(arg, nameflag, strlen(nameflag)) == 0) ||
1379                             (strncmp(arg, "-_cc=", 5) == 0) ||
1380                             (strncmp(arg, "-_sun=", 6) == 0)) {
1381                                 newae(ctx->i_ae, strchr(arg, '=') + 1);
1382                         }
1383                         break;

1385                 case 'V':
1386                         ctx->i_flags &= ~CW_F_ECHO;
1387                         newae(ctx->i_ae, arg);
1388                         break;
1389                 case 'o':
1390                         seen_o = 1;
1391                         if (strlen(arg) == 2) {
1392                                 in_output = 1;
1393                                 newae(ctx->i_ae, arg);
1394                         } else if (ctx->i_flags & CW_F_SHADOW) {
1395                                 newae(ctx->i_ae, "-o");
1396                                 ctx->i_discard = discard_file_name(arg);
1397                                 if (ctx->i_discard == NULL)
1398                                         nomem();
1399 #endif /* ! codereview */
1400                                 newae(ctx->i_ae, ctx->i_discard);
1401                         } else {
1402                                 newae(ctx->i_ae, arg);
1403                         }
1404                         break;
1405                 case 'c':
1406                 case 'S':
1407                         if (strlen(arg) == 2)
1408                                 op = CW_O_COMPILE;
1409                         newae(ctx->i_ae, arg);
1410                         break;
1411                 case 'E':
1412                 case 'P':
1413                         if (strlen(arg) == 2)
1414                                 op = CW_O_PREPROCESS;
1415                 /*FALLTHROUGH*/
1416                 default:
1417                         newae(ctx->i_ae, arg);
1418                 }
1419         }

1421         free(nameflag);

1423         /* See the comment on this same code in do_gcc() */
1424         if (c_files > 1 && op != CW_O_PREPROCESS) {
1425                 errx(2, "multiple source files are "
1426                     "allowed only with -E or -P");
1427         }

1429         if (ctx->i_flags & CW_F_SHADOW) {
1430                 if (op == CW_O_PREPROCESS)
```

```
 850         if ((op == CW_O_LINK || op == CW_O_PREPROCESS) &&
 851             (ctx->i_flags & CW_F_SHADOW))
1431                     exit(0);
1432                 else if (op == CW_O_LINK && c_files == 0)
1433                         exit(0);
1434         }
1435 #endif /* ! codereview */

1437         if (!seen_o && (ctx->i_flags & CW_F_SHADOW)) {
1438                 newae(ctx->i_ae, "-o");
1439                 ctx->i_discard = discard_file_name(NULL);

1441                 if (ctx->i_discard == NULL)
1442                         nomem();
1443 #endif /* ! codereview */
1444                 newae(ctx->i_ae, ctx->i_discard);
1445         }
1446 }

1448 static void
1449 prepctx(cw_ictx_t *ctx)
1450 {
1451         newae(ctx->i_ae, ctx->i_compiler->c_path);

1453         if (ctx->i_flags & CW_F_PROG) {
1454                 (void) printf("%s: %s\n", (ctx->i_flags & CW_F_SHADOW) ?
1455                     "shadow" : "primary", ctx->i_compiler->c_path);
1456                 (void) fflush(stdout);
1457         }

1459         if (!(ctx->i_flags & CW_F_XLATE))
1460                 return;

1462         switch (ctx->i_compiler->c_style) {
1463         case SUN:
1464                 do_cc(ctx);
1465                 break;
1466         case GNU:
1467                 do_gcc(ctx);
1468                 break;
1469         }
1470 }

1472 static int
1473 invoke(cw_ictx_t *ctx)
1474 {
1475         char **newargv;
1476         int ac;
1477         struct ae *a;

1479         if ((newargv = calloc(sizeof (*newargv), ctx->i_ae->ael_argc + 1)) ==
1480             NULL)
1481                 nomem();

1483         if (ctx->i_flags & CW_F_ECHO)
1484                 (void) fprintf(stderr, "+ ");

1486         for (ac = 0, a = ctx->i_ae->ael_head; a; a = a->ae_next, ac++) {
1487                 newargv[ac] = a->ae_arg;
1488                 if (ctx->i_flags & CW_F_ECHO)
1489                         (void) fprintf(stderr, "%s ", a->ae_arg);
1490                 if (a == ctx->i_ae->ael_tail)
1491                         break;
1492         }

1494         if (ctx->i_flags & CW_F_ECHO) {
```



```
1495                 (void) fprintf(stderr, "\n");
1496                 (void) fflush(stderr);
1497         }

1499         if (!(ctx->i_flags & CW_F_EXEC))
1500                 return (0);

1502         /*
1503          * We must fix up the environment here so that the dependency files are
1504          * not trampled by the shadow compiler. Also take care of GCC
1505          * environment variables that will throw off gcc. This assumes a primary
1506          * gcc.
1507          */
1508         if ((ctx->i_flags & CW_F_SHADOW) &&
1509             (unsetenv("SUNPRO_DEPENDENCIES") != 0 ||
1510             unsetenv("DEPENDENCIES_OUTPUT") != 0 ||
1511             unsetenv("GCC_ROOT") != 0)) {
1512                 (void) fprintf(stderr, "error: environment setup failed: %s\n",
1513                     strerror(errno));
1514                 return (-1);
1515         }

1517         (void) execv(newargv[0], newargv);
1518         warn("couldn't run %s", newargv[0]);

1520         return (-1);
1521 }

1523 static int
1524 reap(cw_ictx_t *ctx)
1525 {
1526         int status, ret = 0;
1527         char buf[1024];
1528         struct stat s;

1530         /*
1531          * Only wait for one specific child.
1532          */
1533         if (ctx->i_pid <= 0)
1534                 return (-1);

1536         do {
1537                 if (waitpid(ctx->i_pid, &status, 0) < 0) {
1538                         warn("cannot reap child");
1539                         return (-1);
1540                 }
1541                 if (status != 0) {
1542                         if (WIFSIGNALED(status)) {
1543                                 ret = -WTERMSIG(status);
1544                                 break;
1545                         } else if (WIFEXITED(status)) {
1546                                 ret = WEXITSTATUS(status);
1547                                 break;
1548                         }
1549                 }
1550         } while (!WIFEXITED(status) && !WIFSIGNALED(status));

1552         (void) unlink(ctx->i_discard);
1553         free(ctx->i_discard);
1554 #endif /* ! codereview */

1556         if (stat(ctx->i_stderr, &s) < 0) {
1557                 warn("stat failed on child cleanup");
1558                 return (-1);
1559         }
1560         if (s.st_size != 0) {
```

```
1561                 FILE *f;

1563                 if ((f = fopen(ctx->i_stderr, "r")) != NULL) {
1564                         while (fgets(buf, sizeof (buf), f))
1565                                 (void) fprintf(stderr, "%s", buf);
1566                         (void) fflush(stderr);
1567                         (void) fclose(f);
1568                 }
1569         }
1570         (void) unlink(ctx->i_stderr);
1571         free(ctx->i_stderr);

1573         /*
1574          * cc returns an error code when given -V; we want that to succeed.
1575          */
1576         if (ctx->i_flags & CW_F_PROG)
1577                 return (0);

1579         return (ret);
1580 }

1582 static int
1583 exec_ctx(cw_ictx_t *ctx, int block)
1584 {
 853         char *file;

 855         /*
 856          * To avoid offending cc's sensibilities, the name of its output
 857          * file must end in '.o'.
 858          */
 859         if ((file = tempnam(NULL, ".cw")) == NULL) {
 860                 nomem();
 861                 return (-1);
 862         }
 863         (void) strlcpy(ctx->i_discard, file, MAXPATHLEN);
 864         (void) strlcat(ctx->i_discard, ".o", MAXPATHLEN);
 865         free(file);

1585         if ((ctx->i_stderr = tempnam(NULL, ".cw")) == NULL) {
1586                 nomem();
1587                 return (-1);
1588         }

1590         if ((ctx->i_pid = fork()) == 0) {
1591                 int fd;

1593                 (void) fclose(stderr);
1594                 if ((fd = open(ctx->i_stderr, O_WRONLY | O_CREAT | O_EXCL,
1595                     0666)) < 0) {
1596                         err(1, "open failed for standard error");
1597                 }
1598                 if (dup2(fd, 2) < 0) {
1599                         err(1, "dup2 failed for standard error");
1600                 }
1601                 if (fd != 2)
1602                         (void) close(fd);
1603                 if (freopen("/dev/fd/2", "w", stderr) == NULL) {
1604                         err(1, "freopen failed for /dev/fd/2");
1605                 }

1607                 prepctx(ctx);
1608                 exit(invoke(ctx));
1609         }

1611         if (ctx->i_pid < 0) {
1612                 err(1, "fork failed");
```



```
1613         }

1615         if (block)
1616                 return (reap(ctx));

1618         return (0);
1619 }
```

_____***unchanged_portion_omitted_***