```
**********************************************************
    6236 Mon Feb 19 00:52:49 2018
new/usr/src/tools/cw/cw.1onbld
cw: don't shadow pure pre-processing
cw(1onbld): --shadow not --secondary
**********************************************************
.\"
.\" 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 -shadow
.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
If
.Fl -shadow
compilers are specified
.Nm cw
will invoke shadow compiler, with the outputs modified (as well as any
translation for compiler style) as follows:
.Bl -enum
.It
If neither of
If none of
.Fl c ,
.Fl E ,
.Fl P ,
or
.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.
appears in the argument list (that is, linking is attempted), the shadow
compilers will not be invoked.
This is because the objects built with that compiler which would be linked
have been previously discarded.
.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.
will write to standard output the primary compiler’s standard output, and the
secondary compiler’s standard output will be discarded.
.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
If 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, both 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.
```

****
45601 Mon Feb 19 00:52:51 2018
new/usr/src/tools/cw/cw.c
cw: don't shadow pure pre-processing
****
______unchanged_portion_omitted_

```
static void
do_gcc(cw_ictx_t *ctx)
{
	int c;
	int pic = 0, nolibc = 0;
	int in_output = 0, seen_o = 0, c_files = 0;
	cw_op_t op = CW_O_LINK;
	char *model = NULL;
	char *nameflag;
	int	mflag = 0;

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

	newae(ctx->i_ae, "-fident");
	newae(ctx->i_ae, "-finline");
	newae(ctx->i_ae, "-fno-inline-functions");
	newae(ctx->i_ae, "-fno-builtin");
	newae(ctx->i_ae, "-fno-asm");
	newae(ctx->i_ae, "-fdiagnostics-show-option");
	newae(ctx->i_ae, "-nodefaultlibs");

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

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

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

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

		if (*arg == '-') {
			arglen--;
		} else {
			/*
```

```
			 * Discard inline files that gcc doesn't grok
			 */
			if (!in_output && arglen > 3 &&
			    strcmp(arg + arglen - 3, ".il") == 0)
				continue;

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

			/*
			 * Otherwise, filenames and partial arguments
			 * are passed through for gcc to chew on.  However,
			 * output is always discarded for the secondary
			 * compiler.
			 */
			if ((ctx->i_flags & CW_F_SHADOW) && in_output)
				newae(ctx->i_ae, ctx->i_discard);
			else
				newae(ctx->i_ae, arg);
			in_output = 0;
			continue;
		}

		if (ctx->i_flags & CW_F_CXX) {
			if (strncmp(arg, "-_g++=", 6) == 0) {
				newae(ctx->i_ae, strchr(arg, '=') + 1);
				continue;
			}
			if (strncmp(arg, "-compat=", 8) == 0) {
				/* discard -compat=4 and -compat=5 */
				continue;
			}
			if (strcmp(arg, "-Qoption") == 0) {
				/* discard -Qoption and its two arguments */
				if (ctx->i_oldargc < 3)
					error(arg);
				ctx->i_oldargc -= 2;
				ctx->i_oldargv += 2;
				continue;
			}
			if (strcmp(arg, "-xwe") == 0) {
				/* turn warnings into errors */
				newae(ctx->i_ae, "-Werror");
				continue;
			}
			if (strcmp(arg, "-noex") == 0) {
				/* no exceptions */
				newae(ctx->i_ae, "-fno-exceptions");
				/* no run time type descriptor information */
				newae(ctx->i_ae, "-fno-rtti");
				continue;
			}
			if (strcmp(arg, "-pic") == 0) {
				newae(ctx->i_ae, "-fpic");
				pic = 1;
				continue;
			}
			if (strcmp(arg, "-PIC") == 0) {
				newae(ctx->i_ae, "-fPIC");
				pic = 1;
				continue;
			}
			if (strcmp(arg, "-norunpath") == 0) {
```

```
                                /* gcc has no corresponding option */
                                continue;
                        }
                        if (strcmp(arg, "-nolib") == 0) {
                                /* -nodefaultlibs is on by default */
                                nolibc = 1;
                                continue;
                        }
#if defined(__sparc)
                        if (strcmp(arg, "-cg92") == 0) {
                                mflag |= xlate_xtb(ctx->i_ae, "v8");
                                xlate(ctx->i_ae, "super", xchip_tbl);
                                continue;
                        }
#endif  /* __sparc */
                }

                switch ((c = arg[1])) {
                case '_':
                        if ((strncmp(arg, nameflag, strlen(nameflag)) == 0) ||
                            (strncmp(arg, "-_gcc=", 6) == 0) ||
                            (strncmp(arg, "-_gnu=", 6) == 0)) {
                                newae(ctx->i_ae, strchr(arg, '=') + 1);
                        }
                        break;
                case '#':
                        if (arglen == 1) {
                                newae(ctx->i_ae, "-v");
                                break;
                        }
                        error(arg);
                        break;
                case 'g':
                        newae(ctx->i_ae, "-gdwarf-2");
                        break;
                case 'E':
                        if (arglen == 1) {
                                newae(ctx->i_ae, "-xc");
                                newae(ctx->i_ae, arg);
                                op = CW_O_PREPROCESS;
                                nolibc = 1;
                                break;
                        }
                        error(arg);
                        break;
                case 'c':
                case 'S':
                        if (arglen == 1) {
                                op = CW_O_COMPILE;
                                nolibc = 1;
                        }
                        /* FALLTHROUGH */
                case 'C':
                case 'H':
                case 'p':
                        if (arglen == 1) {
                                newae(ctx->i_ae, arg);
                                break;
                        }
                        error(arg);
                        break;
                case 'A':
                case 'h':
                case 'I':
                case 'i':
                case 'L':
```

```
                case 'l':
                case 'R':
                case 'U':
                case 'u':
                case 'w':
                        newae(ctx->i_ae, arg);
                        break;
                case 'o':
                        seen_o = 1;
                        if (arglen == 1) {
                                in_output = 1;
                                newae(ctx->i_ae, arg);
                        } else if (ctx->i_flags & CW_F_SHADOW) {
                                newae(ctx->i_ae, "-o");
                                newae(ctx->i_ae, ctx->i_discard);
                        } else {
                                newae(ctx->i_ae, arg);
                        }
                        break;
                case 'D':
                        newae(ctx->i_ae, arg);
                        /*
                         * XXX  Clearly a hack ... do we need _KADB too?
                         */
                        if (strcmp(arg, "-D_KERNEL") == 0 ||
                            strcmp(arg, "-D_BOOT") == 0)
                                newae(ctx->i_ae, "-ffreestanding");
                        break;
                case 'd':
                        if (arglen == 2) {
                                if (strcmp(arg, "-dy") == 0) {
                                        newae(ctx->i_ae, "-Wl,-dy");
                                        break;
                                }
                                if (strcmp(arg, "-dn") == 0) {
                                        newae(ctx->i_ae, "-Wl,-dn");
                                        break;
                                }
                        }
                        if (strcmp(arg, "-dalign") == 0) {
                                /*
                                 * -dalign forces alignment in some cases;
                                 * gcc does not need any flag to do this.
                                 */
                                break;
                        }
                        error(arg);
                        break;
                case 'e':
                        if (strcmp(arg,
                            "-erroff=E_EMPTY_TRANSLATION_UNIT") == 0) {
                                /*
                                 * Accept but ignore this -- gcc doesn't
                                 * seem to complain about empty translation
                                 * units
                                 */
                                break;
                        }
                        /* XX64 -- ignore all -erroff= options, for now */
                        if (strncmp(arg, "-erroff=", 8) == 0)
                                break;
                        if (strcmp(arg, "-errtags=yes") == 0) {
                                warnings(ctx->i_ae);
                                break;
                        }
                        if (strcmp(arg, "-errwarn=%all") == 0) {
```

```
				newae(ctx->i_ae, "-Werror");
				break;
			}
			error(arg);
			break;
		case 'f':
			if (strcmp(arg, "-flags") == 0) {
				newae(ctx->i_ae, "--help");
				break;
			}
			if (strncmp(arg, "-features=zla", 13) == 0) {
				/*
				 * Accept but ignore this -- gcc allows
				 * zero length arrays.
				 */
				break;
			}
			error(arg);
			break;
		case 'G':
			newae(ctx->i_ae, "-shared");
			nolibc = 1;
			break;
		case 'k':
			if (strcmp(arg, "-keeptmp") == 0) {
				newae(ctx->i_ae, "-save-temps");
				break;
			}
			error(arg);
			break;
		case 'K':
			if (arglen == 1) {
				if ((arg = *++ctx->i_oldargv) == NULL ||
				    *arg == '\0')
					error("-K");
				ctx->i_oldargc--;
			} else {
				arg += 2;
			}
			if (strcmp(arg, "pic") == 0) {
				newae(ctx->i_ae, "-fpic");
				pic = 1;
				break;
			}
			if (strcmp(arg, "PIC") == 0) {
				newae(ctx->i_ae, "-fPIC");
				pic = 1;
				break;
			}
			error("-K");
			break;
		case 'm':
			if (strcmp(arg, "-mt") == 0) {
				newae(ctx->i_ae, "-D_REENTRANT");
				break;
			}
			if (strcmp(arg, "-m64") == 0) {
				newae(ctx->i_ae, "-m64");
#if defined(__x86)
				newae(ctx->i_ae, "-mtune=opteron");
#endif
				mflag |= M64;
				break;
			}
			if (strcmp(arg, "-m32") == 0) {
				newae(ctx->i_ae, "-m32");
```

```
				mflag |= M32;
				break;
			}
			error(arg);
			break;
		case 'B':	/* linker options */
		case 'M':
		case 'z':
			{
				char *opt;
				size_t len;
				char *s;

				if (arglen == 1) {
					opt = *++ctx->i_oldargv;
					if (opt == NULL || *opt == '\0')
						error(arg);
					ctx->i_oldargc--;
				} else {
					opt = arg + 2;
				}
				len = strlen(opt) + 7;
				if ((s = malloc(len)) == NULL)
					nomem();
				(void) snprintf(s, len, "-Wl,-%c%s", c, opt);
				newae(ctx->i_ae, s);
				free(s);
			}
			break;
		case 'n':
			if (strcmp(arg, "-noqueue") == 0) {
				/*
				 * Horrid license server stuff - n/a
				 */
				break;
			}
			error(arg);
			break;
		case 'O':
			if (arglen == 1) {
				newae(ctx->i_ae, "-O");
				break;
			}
			error(arg);
			break;
		case 'P':
			/*
			 * We could do '-E -o filename.i', but that's hard,
			 * and we don't need it for the case that's triggering
			 * this addition.  We'll require the user to specify
			 * -o in the Makefile.  If they don't they'll find out
			 * in a hurry.
			 */
			newae(ctx->i_ae, "-E");
			op = CW_O_PREPROCESS;
			nolibc = 1;
			break;
		case 'q':
			if (strcmp(arg, "-qp") == 0) {
				newae(ctx->i_ae, "-p");
				break;
			}
			error(arg);
			break;
		case 's':
			if (arglen == 1) {
				newae(ctx->i_ae, "-Wl,-s");
				break;
			}
			error(arg);
			break;
		case 't':
			if (arglen == 1) {
				newae(ctx->i_ae, "-Wl,-t");
				break;
			}
			error(arg);
			break;
		case 'V':
			if (arglen == 1) {
				ctx->i_flags &= ~CW_F_ECHO;
				newae(ctx->i_ae, "--version");
				break;
			}
			error(arg);
			break;
		case 'v':
			if (arglen == 1) {
				warnings(ctx->i_ae);
				break;
			}
			error(arg);
			break;
		case 'W':
			if (strncmp(arg, "-Wp,-xc99", 9) == 0) {
				/*
				 * gcc's preprocessor will accept c99
				 * regardless, so accept and ignore.
				 */
				break;
			}
			if (strncmp(arg, "-Wa,", 4) == 0 ||
			    strncmp(arg, "-Wp,", 4) == 0 ||
			    strncmp(arg, "-Wl,", 4) == 0) {
				newae(ctx->i_ae, arg);
				break;
			}
			if (strcmp(arg, "-W0,-xc99=pragma") == 0) {
				/* (undocumented) enables _Pragma */
				break;
			}
			if (strcmp(arg, "-W0,-xc99=%none") == 0) {
				/*
				 * This is a polite way of saying
				 * "no c99 constructs allowed!"
				 * For now, just accept and ignore this.
				 */
				break;
			}
			if (strcmp(arg, "-W0,-noglobal") == 0 ||
			    strcmp(arg, "-W0,-xglobalstatic") == 0) {
				/*
				 * gcc doesn't prefix local symbols
				 * in debug mode, so this is not needed.
				 */
				break;
			}
			if (strcmp(arg, "-W0,-Lt") == 0) {
				/*
				 * Generate tests at the top of loops.
				 * There is no direct gcc equivalent, ignore.
				 */
```

```
				break;
			}
			if (strcmp(arg, "-W0,-xdbggen=no%usedonly") == 0) {
				newae(ctx->i_ae,
				    "-fno-eliminate-unused-debug-symbols");
				newae(ctx->i_ae,
				    "-fno-eliminate-unused-debug-types");
				break;
			}
			if (strcmp(arg, "-W2,-xwrap_int") == 0) {
				/*
				 * Use the legacy behaviour (pre-SS11)
				 * for integer wrapping.
				 * gcc does not need this.
				 */
				break;
			}
			if (strcmp(arg, "-W2,-Rcond_elim") == 0) {
				/*
				 * Elimination and expansion of conditionals;
				 * gcc has no direct equivalent.
				 */
				break;
			}
			if (strcmp(arg, "-Wd,-xsafe=unboundsym") == 0) {
				/*
				 * Prevents optimizing away checks for
				 * unbound weak symbol addresses.  gcc does
				 * not do this, so it's not needed.
				 */
				break;
			}
			if (strncmp(arg, "-Wc,-xcode=", 11) == 0) {
				xlate(ctx->i_ae, arg + 11, xcode_tbl);
				if (strncmp(arg + 11, "pic", 3) == 0)
					pic = 1;
				break;
			}
			if (strncmp(arg, "-Wc,-Qiselect", 13) == 0) {
				/*
				 * Prevents insertion of register symbols.
				 * gcc doesn't do this, so ignore it.
				 */
				break;
			}
			if (strcmp(arg, "-Wc,-Qassembler-ounrefsym=0") == 0) {
				/*
				 * Prevents optimizing away of static variables.
				 * gcc does not do this, so it's not needed.
				 */
				break;
			}
#if defined(__x86)
			if (strcmp(arg, "-Wu,-xmodel=kernel") == 0) {
				newae(ctx->i_ae, "-ffreestanding");
				newae(ctx->i_ae, "-mno-red-zone");
				model = "-mcmodel=kernel";
				nolibc = 1;
				break;
			}
			if (strcmp(arg, "-Wu,-save_args") == 0) {
				newae(ctx->i_ae, "-msave-args");
				break;
			}
#endif	/* __x86 */
			error(arg);
```

```
                        break;
                case ’X’:
                        if (strcmp(arg, "-Xa") == 0 ||
                            strcmp(arg, "-Xt") == 0) {
                                Xamode(ctx->i_ae);
                                break;
                        }
                        if (strcmp(arg, "-Xc") == 0) {
                                Xcmode(ctx->i_ae);
                                break;
                        }
                        if (strcmp(arg, "-Xs") == 0) {
                                Xsmode(ctx->i_ae);
                                break;
                        }
                        error(arg);
                        break;
                case ’x’:
                        if (arglen == 1)
                                error(arg);
                        switch (arg[2]) {
#if defined(__x86)
                        case ’3’:
                                if (strcmp(arg, "-x386") == 0) {
                                        newae(ctx->i_ae, "-march=i386");
                                        break;
                                }
                                error(arg);
                                break;
                        case ’4’:
                                if (strcmp(arg, "-x486") == 0) {
                                        newae(ctx->i_ae, "-march=i486");
                                        break;
                                }
                                error(arg);
                                break;
#endif  /* __x86 */
                        case ’a’:
                                if (strncmp(arg, "-xarch=", 7) == 0) {
                                        mflag |= xlate_xtb(ctx->i_ae, arg + 7);
                                        break;
                                }
                                error(arg);
                                break;
                        case ’b’:
                                if (strncmp(arg, "-xbuiltin=", 10) == 0) {
                                        if (strcmp(arg + 10, "%all"))
                                                newae(ctx->i_ae, "-fbuiltin");
                                        break;
                                }
                                error(arg);
                                break;
                        case ’C’:
                                /* Accept C++ style comments -- ignore */
                                if (strcmp(arg, "-xCC") == 0)
                                        break;
                                error(arg);
                                break;
                        case ’c’:
                                if (strncmp(arg, "-xc99=%all", 10) == 0) {
                                        newae(ctx->i_ae, "-std=gnu99");
                                        break;
                                }
                                if (strncmp(arg, "-xc99=%none", 11) == 0) {
                                        newae(ctx->i_ae, "-std=gnu89");
                                        break;
```

```
                                }
                                if (strncmp(arg, "-xchip=", 7) == 0) {
                                        xlate(ctx->i_ae, arg + 7, xchip_tbl);
                                        break;
                                }
                                if (strncmp(arg, "-xcode=", 7) == 0) {
                                        xlate(ctx->i_ae, arg + 7, xcode_tbl);
                                        if (strncmp(arg + 7, "pic", 3) == 0)
                                                pic = 1;
                                        break;
                                }
                                if (strncmp(arg, "-xcache=", 8) == 0)
                                        break;
                                if (strncmp(arg, "-xcrossfile", 11) == 0)
                                        break;
                                error(arg);
                                break;
                        case ’d’:
                                if (strcmp(arg, "-xdepend") == 0)
                                        break;
                                if (strncmp(arg, "-xdebugformat=", 14) == 0)
                                        break;
                                error(arg);
                                break;
                        case ’F’:
                                /*
                                 * Compile for mapfile reordering, or unused
                                 * section elimination, syntax can be -xF or
                                 * more complex, like -xF=%all -- ignore.
                                 */
                                if (strncmp(arg, "-xF", 3) == 0)
                                        break;
                                error(arg);
                                break;
                        case ’i’:
                                if (strncmp(arg, "-xinline", 8) == 0)
                                        /* No inlining; ignore */
                                        break;
                                if (strcmp(arg, "-xildon") == 0 ||
                                    strcmp(arg, "-xildoff") == 0)
                                        /* No incremental linking; ignore */
                                        break;
                                error(arg);
                                break;
#if defined(__x86)
                        case ’m’:
                                if (strcmp(arg, "-xmodel=kernel") == 0) {
                                        newae(ctx->i_ae, "-ffreestanding");
                                        newae(ctx->i_ae, "-mno-red-zone");
                                        model = "-mcmodel=kernel";
                                        nolibc = 1;
                                        break;
                                }
                                error(arg);
                                break;
#endif  /* __x86 */
                        case ’M’:
                                if (strcmp(arg, "-xM") == 0) {
                                        newae(ctx->i_ae, "-M");
                                        break;
                                }
                                if (strcmp(arg, "-xM1") == 0) {
                                        newae(ctx->i_ae, "-MM");
                                        break;
                                }
                                error(arg);
```

```
				break;
			case 'n':
				if (strcmp(arg, "-xnolib") == 0) {
					nolibc = 1;
					break;
				}
				error(arg);
				break;
			case 'O':
				if (strncmp(arg, "-xO", 3) == 0) {
					size_t len = strlen(arg);
					char *s;
					int c = *(arg + 3);
					int level;

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

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

					level = atoi(arg + 3);
					if (level > 5)
						error(arg);
					if (level >= 2) {
						/*
						 * For gcc-3.4.x at -O2 we
						 * need to disable optimizations
						 * that break ON.
						 */
						optim_disable(ctx->i_ae, level);
						/*
						 * limit -xO3 to -O2 as well.
						 */
						level = 2;
					}
					(void) snprintf(s, len, "-O%d", level);
					newae(ctx->i_ae, s);
					free(s);
					break;
				}
				error(arg);
				break;
			case 'p':
				if (strcmp(arg, "-xpentium") == 0) {
					newae(ctx->i_ae, "-march=pentium");
					break;
				}
				if (strcmp(arg, "-xpg") == 0) {
					newae(ctx->i_ae, "-pg");
					break;
				}
				error(arg);
				break;
			case 'r':
				if (strncmp(arg, "-xregs=", 7) == 0) {
					xlate(ctx->i_ae, arg + 7, xregs_tbl);
					break;
				}
				error(arg);
				break;
			case 's':
				if (strcmp(arg, "-xs") == 0 ||
				    strcmp(arg, "-xspace") == 0 ||
				    strcmp(arg, "-xstrconst") == 0)
					break;
```

```
					error(arg);
					break;
			case 't':
				if (strcmp(arg, "-xtransition") == 0) {
					newae(ctx->i_ae, "-Wtransition");
					break;
				}
				if (strcmp(arg, "-xtrigraphs=yes") == 0) {
					newae(ctx->i_ae, "-trigraphs");
					break;
				}
				if (strcmp(arg, "-xtrigraphs=no") == 0) {
					newae(ctx->i_ae, "-notrigraphs");
					break;
				}
				if (strncmp(arg, "-xtarget=", 9) == 0) {
					xlate(ctx->i_ae, arg + 9, xtarget_tbl);
					break;
				}
				error(arg);
				break;
			case 'e':
			case 'h':
			case 'l':
			default:
				error(arg);
				break;
			}
			break;
		case 'Y':
			if (arglen == 1) {
				if ((arg = *++ctx->i_oldargv) == NULL ||
				    *arg == '\0')
					error("-Y");
				ctx->i_oldargc--;
				arglen = strlen(arg + 1);
			} else {
				arg += 2;
			}
			/* Just ignore -YS,... for now */
			if (strncmp(arg, "S,", 2) == 0)
				break;
			if (strncmp(arg, "l,", 2) == 0) {
				char *s = strdup(arg);
				s[0] = '-';
				s[1] = 'B';
				newae(ctx->i_ae, s);
				free(s);
				break;
			}
			if (strncmp(arg, "I,", 2) == 0) {
				char *s = strdup(arg);
				s[0] = '-';
				s[1] = 'I';
				newae(ctx->i_ae, "-nostdinc");
				newae(ctx->i_ae, s);
				free(s);
				break;
			}
			error(arg);
			break;
		case 'Q':
			/*
			 * We could map -Qy into -Wl,-Qy etc.
			 */
		default:
```

```
                        error(arg);
                        break;
                }
        }

        free(nameflag);

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

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

        switch (mflag) {
        case 0:
                /* FALLTHROUGH */
        case M32:
#if defined(__sparc)
                /*
                 * Only -m32 is defined and so put in the missing xarch
                 * translation.
                 */
                newae(ctx->i_ae, "-mcpu=v8");
                newae(ctx->i_ae, "-mno-v8plus");
#endif
                break;
        case M64:
#if defined(__sparc)
                /*
                 * Only -m64 is defined and so put in the missing xarch
                 * translation.
                 */
                newae(ctx->i_ae, "-mcpu=v9");
#endif
                break;
        case SS12:
#if defined(__sparc)
                /* no -m32/-m64 flag used - this is an error for sparc builds */
                (void) fprintf(stderr, "No -m32/-m64 flag defined\n");
                exit(2);
#endif
                break;
        case SS11:
                /* FALLTHROUGH */
        case (SS11|M32):
        case (SS11|M64):
                break;
        case (SS12|M32):
#if defined(__sparc)
                /*
                 * Need to add in further 32 bit options because with SS12
                 * the xarch=sparcvis option can be applied to 32 or 64
                 * bit, and so the translatation table (xtbl) cannot handle
                 * that.
                 */
                newae(ctx->i_ae, "-mv8plus");
```

```
#endif
                break;
        case (SS12|M64):
                break;
        default:
                (void) fprintf(stderr,
                    "Incompatible -xarch= and/or -m32/-m64 options used.\n");
                exit(2);
        }

        if ((op == CW_O_LINK || op == CW_O_PREPROCESS) &&
            (ctx->i_flags & CW_F_SHADOW))
        if (op == CW_O_LINK && (ctx->i_flags & CW_F_SHADOW))
                exit(0);

        if (model && !pic)
                newae(ctx->i_ae, model);
        if (!nolibc)
                newae(ctx->i_ae, "-lc");
        if (!seen_o && (ctx->i_flags & CW_F_SHADOW)) {
                newae(ctx->i_ae, "-o");
                newae(ctx->i_ae, ctx->i_discard);
        }
}
```

______***unchanged_portion_omitted_***