

```

*****
35097 Wed Oct 1 18:40:28 2014
new/usr/src/Makefile.master
5196 The cw wrapper restricts gcc to -O2
*****
1 #
2 # CDDL HEADER START
3 #
4 # The contents of this file are subject to the terms of the
5 # Common Development and Distribution License (the "License").
6 # You may not use this file except in compliance with the License.
7 #
8 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
9 # or http://www.opensolaris.org/os/licensing.
10 # See the License for the specific language governing permissions
11 # and limitations under the License.
12 #
13 # When distributing Covered Code, include this CDDL HEADER in each
14 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
15 # If applicable, add the following below this CDDL HEADER, with the
16 # fields enclosed by brackets "[]" replaced with your own identifying
17 # information: Portions Copyright [yyyy] [name of copyright owner]
18 #
19 # CDDL HEADER END
20 #
21 #
22 #
23 # Copyright (c) 1989, 2010, Oracle and/or its affiliates. All rights reserved.
24 # Copyright (c) 2012 by Delphix. All rights reserved.
25 # Copyright 2014 Garrett D'Amore <garrett@damore.org>
26 # Copyright 2014 Gary Mills
27 #
28 #
29 #
30 # Makefile.master, global definitions for system source
31 #
32 ROOT= /proto
33 #
34 #
35 # Adjunct root, containing an additional proto area to be used for headers
36 # and libraries.
37 #
38 ADJUNCT_PROTO=
39 #
40 #
41 # Adjunct for building things that run on the build machine.
42 #
43 NATIVE_ADJUNCT= /usr
44 #
45 #
46 # RELEASE_BUILD should be cleared for final release builds.
47 # NOT_RELEASE_BUILD is exactly what the name implies.
48 #
49 # __GNUC toggles the building of ON components using gcc and related tools.
50 # Normally set to '#', set it to '' to do gcc build.
51 #
52 # The declaration POUND_SIGN is always '#'. This is needed to get around the
53 # make feature that '#' is always a comment delimiter, even when escaped or
54 # quoted. We use this macro expansion method to get POUND_SIGN rather than
55 # always breaking out a shell because the general case can cause a noticeable
56 # slowdown in build times when so many Makefiles include Makefile.master.
57 #
58 # While the majority of users are expected to override the setting below
59 # with an env file (via nightly or bldenv), if you aren't building that way
60 # (ie, you're using "ws" or some other bootstrapping method) then you need
61 # this definition in order to avoid the subshell invocation mentioned above.

```

```

62 #
63 PRE_POUND= pre\#
64 POUND_SIGN= $(PRE_POUND:pre\%=%)
65 #
66 NOT_RELEASE_BUILD=
67 RELEASE_BUILD= $(POUND_SIGN)
68 $(RELEASE_BUILD)NOT_RELEASE_BUILD= $(POUND_SIGN)
69 PATCH_BUILD= $(POUND_SIGN)
70 #
71 #
72 # SPARC_BLD is '#' for an Intel build.
73 # INTEL_BLD is '#' for a Sparc build.
74 SPARC_BLD_1= $(MACH:i386=$(POUND_SIGN))
75 SPARC_BLD= $(SPARC_BLD_1:sparc=)
76 INTEL_BLD_1= $(MACH:sparc=$(POUND_SIGN))
77 INTEL_BLD= $(INTEL_BLD_1:i386=)
78 #
79 # The variables below control the compilers used during the build.
80 # There are a number of permutations.
81 #
82 # __GNUC and __SUNC control (and indicate) the primary compiler. Whichever
83 # one is not POUND_SIGN is the primary, with the other as the shadow. They
84 # may also be used to control entirely compiler-specific Makefile assignments.
85 # __GNUC and GCC are the default.
86 #
87 # __GNUC64 indicates that the 64bit build should use the GNU C compiler.
88 # There is no Sun C analogue.
89 #
90 # The following version-specific options are operative regardless of which
91 # compiler is primary, and control the versions of the given compilers to be
92 # used. They also allow compiler-version specific Makefile fragments.
93 #
94 #
95 __SUNC= $(POUND_SIGN)
96 $(__SUNC)__GNUC= $(POUND_SIGN)
97 __GNUC64= $(__GNUC)
98 #
99 # CLOSED is the root of the tree that contains source which isn't released
100 # as open source
101 CLOSED= $(SRC)/../closed
102 #
103 # BUILD_TOOLS is the root of all tools including compilers.
104 # ONBLD_TOOLS is the root of all the tools that are part of SUNWonbld.
105 #
106 BUILD_TOOLS= /ws/onnv-tools
107 ONBLD_TOOLS= $(BUILD_TOOLS)/onbld
108 #
109 JAVA_ROOT= /usr/java
110 #
111 SFW_ROOT= /usr/sfw
112 SFWINCDIR= $(SFW_ROOT)/include
113 SFWLIBDIR= $(SFW_ROOT)/lib
114 SFWLIBDIR64= $(SFW_ROOT)/lib/$(MACH64)
115 #
116 GCC_ROOT= /opt/gcc/4.4.4
117 GCCLIBDIR= $(GCC_ROOT)/lib
118 GCCLIBDIR64= $(GCC_ROOT)/lib/$(MACH64)
119 #
120 DOCBOOK_XSL_ROOT= /usr/share/sgml/docbook/xsl-stylesheets
121 #
122 RPCGEN= /usr/bin/rpcgen
123 STABS= $(ONBLD_TOOLS)/bin/$(MACH)/stabs
124 ELFXTRACT= $(ONBLD_TOOLS)/bin/$(MACH)/elfextract
125 MBH_PATCH= $(ONBLD_TOOLS)/bin/$(MACH)/mbh_patch
126 ECHO= echo
127 INS= install

```

```

128 TRUE= true
129 SYMLINK= /usr/bin/ln -s
130 LN= /usr/bin/ln
131 CHMOD= /usr/bin/chmod
132 MV= /usr/bin/mv -f
133 RM= /usr/bin/rm -f
134 CUT= /usr/bin/cut
135 NM= /usr/ccs/bin/nm
136 DIFF= /usr/bin/diff
137 GREP= /usr/bin/grep
138 EGREP= /usr/bin/egrep
139 ELFWRAP= /usr/bin/elfwrap
140 KSH93= /usr/bin/ksh93
141 SED= /usr/bin/sed
142 NAWK= /usr/bin/nawk
143 CP= /usr/bin/cp -f
144 MCS= /usr/ccs/bin/mcs
145 CAT= /usr/bin/cat
146 ELFDUMP= /usr/ccs/bin/elfdump
147 M4= /usr/ccs/bin/m4
148 STRIP= /usr/ccs/bin/strip
149 LEX= /usr/ccs/bin/lex
150 FLEX= $(SFW_ROOT)/bin/flex
151 YACC= /usr/ccs/bin/yacc
152 CPP= /usr/lib/cpp
153 JAVAC= $(JAVA_ROOT)/bin/javac
154 JAVAH= $(JAVA_ROOT)/bin/javah
155 JAVADOC= $(JAVA_ROOT)/bin/javadoc
156 RMIC= $(JAVA_ROOT)/bin/rmic
157 JAR= $(JAVA_ROOT)/bin/jar
158 CTFCONVERT= $(ONBLD_TOOLS)/bin/$(MACH)/ctfconvert
159 CTFMERGE= $(ONBLD_TOOLS)/bin/$(MACH)/ctfmerge
160 CTFSTABS= $(ONBLD_TOOLS)/bin/$(MACH)/ctfstabs
161 CTFSTRIP= $(ONBLD_TOOLS)/bin/$(MACH)/ctfstrip
162 NDRGEN= $(ONBLD_TOOLS)/bin/$(MACH)/ndrgen
163 GENOFFSETS= $(ONBLD_TOOLS)/bin/genoffsets
164 CTFCVTPTBL= $(ONBLD_TOOLS)/bin/ctfcvtptbl
165 CTFINDMOD= $(ONBLD_TOOLS)/bin/ctffindmod
166 XREF= $(ONBLD_TOOLS)/bin/xref
167 FIND= /usr/bin/find
168 PERL= /usr/bin/perl
169 PERL_VERSION= 5.10.0
170 PERL_PKGVERS= -510
171 PYTHON_26= /usr/bin/python2.6
172 PYTHON= $(PYTHON_26)
173 SORT= /usr/bin/sort
174 TOUCH= /usr/bin/touch
175 WC= /usr/bin/wc
176 XARGS= /usr/bin/xargs
177 ELFEDIT= /usr/bin/elfedit
178 ELFSIGN= /usr/bin/elfsign
179 DTRACE= /usr/sbin/dtrace -xnolib
180 UNIQ= /usr/bin/uniq
181 TAR= /usr/bin/tar
182 ASTBINDIR= /usr/ast/bin
183 MSGCC= $(ASTBINDIR)/msgcc

185 FILEMODE= 644
186 DIRMODE= 755

188 #
189 # The version of the patch makeup table optimized for build-time use. Used
190 # during patch builds only.
191 $(PATCH_BUILD)PMTMO_FILE=$(SRC)/patch_makeup_table.mo

193 # Declare that nothing should be built in parallel.

```

```

194 # Individual Makefiles can use the .PARALLEL target to declare otherwise.
195 .NO_PARALLEL:

197 # For stylistic checks
198 #
199 # Note that the X and C checks are not used at this time and may need
200 # modification when they are actually used.
201 #
202 CSTYLE= $(ONBLD_TOOLS)/bin/cstyle
203 CSTYLE_TAIL=
204 HDRCHK= $(ONBLD_TOOLS)/bin/hdrchk
205 HDRCHK_TAIL=
206 JSTYLE= $(ONBLD_TOOLS)/bin/jstyle

208 DOT_H_CHECK= \
209     @$(ECHO) "checking $<"; $(CSTYLE) $< $(CSTYLE_TAIL); \
210     $(HDRCHK) $< $(HDRCHK_TAIL)

212 DOT_X_CHECK= \
213     @$(ECHO) "checking $<"; $(RPCGEN) -C -h $< | $(CSTYLE) $(CSTYLE_TAIL); \
214     $(RPCGEN) -C -h $< | $(HDRCHK) $< $(HDRCHK_TAIL)

216 DOT_C_CHECK= \
217     @$(ECHO) "checking $<"; $(CSTYLE) $< $(CSTYLE_TAIL)

219 MANIFEST_CHECK= \
220     @$(ECHO) "checking $<"; \
221     SVCCFG_DTD=$(SRC)/cmd/svc/dtd/service_bundle.dtd.1 \
222     SVCCFG_REPOSITORY=$(SRC)/cmd/svc/seed/global.db \
223     SVCCFG_CONFIGD_PATH=$(SRC)/cmd/svc/configd/svc.configd-native \
224     $(SRC)/cmd/svc/svccfg/svccfg-native validate $<

226 INS.file= $(RM) $@; $(INS) -s -m $(FILEMODE) -f $(@D) $<
227 INS.dir= $(INS) -s -d -m $(DIRMODE) $@
228 # installs and renames at once
229 #
230 INS.rename= $(INS.file); $(MV) $(@D)/$(<F) $@

232 # install a link
233 INSLINKTARGET= $<
234 INS.link= $(RM) $@; $(LN) $(INSLINKTARGET) $@
235 INS.symlink= $(RM) $@; $(SYMLINK) $(INSLINKTARGET) $@

237 #
238 # Python bakes the mtime of the .py file into the compiled .pyc and
239 # rebuilds if the baked-in mtime != the mtime of the source file
240 # (rather than only if it's less than), thus when installing python
241 # files we must make certain to not adjust the mtime of the source
242 # (.py) file.
243 #
244 INS.pyfile= $(INS.file); $(TOUCH) -r $< $@

246 # MACH must be set in the shell environment per uname -p on the build host
247 # More specific architecture variables should be set in lower makefiles.
248 #
249 # MACH64 is derived from MACH, and BUILD64 is set to '#' for
250 # architectures on which we do not build 64-bit versions.
251 # (There are no such architectures at the moment.)
252 #
253 # Set BUILD64=# in the environment to disable 64-bit amd64
254 # builds on i386 machines.

256 MACH64_1= $(MACH:sparc=sparcv9)
257 MACH64= $(MACH64_1:i386=amd64)

259 MACH32_1= $(MACH:sparc=sparcv7)

```

```

260 MACH32=          $(MACH32_1:i386=i86)

262 sparc_BUILD64=
263 i386_BUILD64=
264 BUILD64=         $$((MACH)_BUILD64)

266 #
267 # C compiler mode. Future compilers may change the default on us,
268 # so force extended ANSI mode globally. Lower level makefiles can
269 # override this by setting CCMODE.
270 #
271 CCMODE=           -Xa
272 CCMODE64=        -Xa

274 #
275 # C compiler verbose mode. This is so we can enable it globally,
276 # but turn it off in the lower level makefiles of things we cannot
277 # (or aren't going to) fix.
278 #
279 CCVERBOSE=       -v

281 # set this to the secret flag "-Wc,-Qiselect-v9abiwarn=1" to get warnings
282 # from the compiler about places the -xarch=v9 may differ from -xarch=v9c.
283 V9ABIWARN=

285 # set this to the secret flag "-Wc,-Qiselect-regsym=0" to disable register
286 # symbols (used to detect conflicts between objects that use global registers)
287 # we disable this now for safety, and because genunix doesn't link with
288 # this feature (the v9 default) enabled.
289 #
290 # REGSYM is separate since the C++ driver syntax is different.
291 CCREGSYM=         -Wc,-Qiselect-regsym=0
292 CCCREGSYM=       -Qoption cg -Qiselect-regsym=0

294 # Prevent the removal of static symbols by the SPARC code generator (cg).
295 # The x86 code generator (ube) does not remove such symbols and as such
296 # using this workaround is not applicable for x86.
297 #
298 CCSTATICSYM=     -Wc,-Qassembler-ounrefsym=0
299 #
300 # generate 32-bit addresses in the v9 kernel. Saves memory.
301 CCABS32=         -Wc,-xcode=abs32
302 #
303 # generate v9 code which tolerates callers using the v7 ABI, for the sake of
304 # system calls.
305 CC32BITCALLERS=  -_gcc=-massume-32bit-callers

307 # GCC, especially, is increasingly beginning to auto-inline functions and
308 # sadly does so separately not under the general -fno-inline-functions
309 # Additionally, we wish to prevent optimisations which cause GCC to clone
310 # functions -- in particular, these may cause unhelpful symbols to be
311 # emitted instead of function names
312 CCNOAUTOINLINE=  -_gcc=-fno-inline-small-functions \
313                 -_gcc=-fno-inline-functions-called-once \
314                 -_gcc=-fno-ipa-cp

316 # One optimization the compiler might perform is to turn this:
317 #     #pragma weak foo
318 #     extern int foo;
319 #     if (&foo)
320 #         foo = 5;
321 # into
322 #     foo = 5;
323 # Since we do some of this (foo might be referenced in common kernel code
324 # but provided only for some cpu modules or platforms), we disable this
325 # optimization.

```

```

326 #
327 sparc_CCUNBOUND = -Wd,-xsafe=unboundsym
328 i386_CCUNBOUND  =
329 CCUNBOUND       = $$((MACH)_CCUNBOUND)

331 #
332 # compiler '-xarch' flag. This is here to centralize it and make it
333 # overridable for testing.
334 sparc_XARCH=    -m32
335 sparcv9_XARCH= -m64
336 i386_XARCH=
337 amd64_XARCH=   -m64 -Ui386 -U__i386

339 # assembler '-xarch' flag. Different from compiler '-xarch' flag.
340 sparc_AS_XARCH= -xarch=v8plus
341 sparcv9_AS_XARCH= -xarch=v9
342 i386_AS_XARCH=
343 amd64_AS_XARCH= -xarch=amd64 -P -Ui386 -U__i386

345 #
346 # These flags define what we need to be 'standalone' i.e. -not- part
347 # of the rather more cosy userland environment. This basically means
348 # the kernel.
349 #
350 # XX64 future versions of gcc will make -mmodel=kernel imply -mno-red-zone
351 #
352 sparc_STAND_FLAGS= -_gcc=-ffreestanding
353 sparcv9_STAND_FLAGS= -_gcc=-ffreestanding
354 # Disabling MMX also disables 3DNow, disabling SSE also disables all later
355 # additions to SSE (SSE2, AVX ,etc.)
356 NO_SIMD=          -_gcc=-mno-mmx -_gcc=-mno-sse
357 i386_STAND_FLAGS= -_gcc=-ffreestanding $(NO_SIMD)
358 amd64_STAND_FLAGS= -xmodel=kernel $(NO_SIMD)

360 SAVEARGS=        -Wu,-save_args
361 amd64_STAND_FLAGS += $(SAVEARGS)

363 STAND_FLAGS_32 = $$((MACH)_STAND_FLAGS)
364 STAND_FLAGS_64 = $$((MACH64)_STAND_FLAGS)

366 #
367 # disable the incremental linker
368 ILDOFF=          -xildoff
369 #
370 XDEPEND=         -xdepend
371 XFFLAG=          -xF=%all
372 KESS=           -xs
373 XSTRCONST=      -xstrconst

375 #
376 # turn warnings into errors (C)
377 CERRWARN = -errtags=yes -errwarn=%all
378 CERRWARN += -erroff=E_EMPTY_TRANSLATION_UNIT
379 CERRWARN += -erroff=E_STATEMENT_NOT_REACHED

381 CERRWARN += -_gcc=-Wno-missing-braces
382 CERRWARN += -_gcc=-Wno-sign-compare
383 CERRWARN += -_gcc=-Wno-unknown-pragmas
384 CERRWARN += -_gcc=-Wno-unused-parameter
385 CERRWARN += -_gcc=-Wno-missing-field-initializers

387 # Unfortunately, this option can misfire very easily and unfixably.
388 CERRWARN += -_gcc=-Wno-array-bounds

390 # Suppress it: this warning generates many false alarms
391 CERRWARN += -_gcc=-Wno-uninitialized

```

```

393 # DEBUG v. -nd make for frequent unused variables, empty conditions, etc. in
394 # -nd builds
395 $(RELEASE_BUILD)CERRWARN += _gcc=-Wno-unused
396 $(RELEASE_BUILD)CERRWARN += _gcc=-Wno-empty-body

398 #
399 # turn warnings into errors (C++)
400 CCERRWARN= -xwe

402 # C99 mode
403 C99_ENABLE= -xc99=%all
404 C99_DISABLE= -xc99=%none
405 C99MODE= $(C99_DISABLE)
406 C99LMODE= $(C99MODE:-xc99%=-Xc99%)

408 # In most places, assignments to these macros should be appended with +=
409 # (CPPFLAGS.master allows values to be prepended to CPPFLAGS).
410 sparc_CFLAGS= $(sparc_XARCH) $(CCSTATICSYM)
411 sparcv9_CFLAGS= $(sparcv9_XARCH) -dalign $(CCVERBOSE) $(V9ABIWARN) $(CCREGSYM) \
412 $(CCSTATICSYM)
413 i386_CFLAGS= $(i386_XARCH)
414 amd64_CFLAGS= $(amd64_XARCH)

416 sparc_ASFLAGS= $(sparc_AS_XARCH)
417 sparcv9_ASFLAGS=$(sparcv9_AS_XARCH)
418 i386_ASFLAGS= $(i386_AS_XARCH)
419 amd64_ASFLAGS= $(amd64_AS_XARCH)

421 # Adjustments to specified optimization level
422 ADJUST_COPT= -_gcc=-fno-strict-aliasing -_gcc=-fno-tree-vrp

424 #
425 sparc_COPTFLAG= $(ADJUST_COPT) -xO3
426 sparcv9_COPTFLAG= $(ADJUST_COPT) -xO3
427 i386_COPTFLAG= $(ADJUST_COPT) -O
428 amd64_COPTFLAG= $(ADJUST_COPT) -xO3
429 sparc_COPTFLAG= -xO3
430 sparcv9_COPTFLAG= -xO3
431 i386_COPTFLAG= -O
432 amd64_COPTFLAG= -xO3

430 COPTFLAG= $($MACH)_COPTFLAG)
431 COPTFLAG64= $($MACH64)_COPTFLAG)

433 # When -g is used, the compiler globalizes static objects
434 # (gives them a unique prefix). Disable that.
435 CNOGLOBAL= -W0,-noglobal

437 # Direct the Sun Studio compiler to use a static globalization prefix based on t
438 # name of the module rather than something unique. Otherwise, objects
439 # will not build deterministically, as subsequent compilations of identical
440 # source will yeild objects that always look different.
441 #
442 # In the same spirit, this will also remove the date from the N_OPT stab.
443 CGLOBALSTATIC= -W0,-xglobalstatic

445 # Sometimes we want all symbols and types in debugging information even
446 # if they aren't used.
447 CALLSYMS= -W0,-xdbggen=no%usedonly

449 #
450 # Default debug format for Sun Studio 11 is dwarf, so force it to
451 # generate stabs.
452 #
453 DEBUGFORMAT= -xdebugformat=stabs

```

```

455 #
456 # Flags used to build in debug mode for ctf generation. Bugs in the Devpro
457 # compilers currently prevent us from building with cc-emitted DWARF.
458 #
459 CTF_FLAGS_sparc = -g -Wc,-Qiselect-T1 $(C99MODE) $(CNOGLOBAL) $(CDWARFSTR)
460 CTF_FLAGS_i386 = -g $(C99MODE) $(CNOGLOBAL) $(CDWARFSTR)

462 CTF_FLAGS_sparcv9 = $(CTF_FLAGS_sparc)
463 CTF_FLAGS_amd64 = $(CTF_FLAGS_i386)

465 # Sun Studio produces broken userland code when saving arguments.
466 $(__GNUCC)CTF_FLAGS_amd64 += $(SAVEARGS)

468 CTF_FLAGS_32 = $(CTF_FLAGS_$(MACH)) $(DEBUGFORMAT)
469 CTF_FLAGS_64 = $(CTF_FLAGS_$(MACH64)) $(DEBUGFORMAT)
470 CTF_FLAGS = $(CTF_FLAGS_32)

472 #
473 # Flags used with genoffsets
474 #
475 GOFLAGS = -_noecho \
476 $(CALLSYMS) \
477 $(CDWARFSTR)

479 OFFSETS_CREATE = $(GENOFFSETS) -s $(CTFSTABS) -r $(CTFCONVERT) \
480 $(CC) $(GOFLAGS) $(CFLAGS) $(CPPFLAGS)

482 OFFSETS_CREATE64 = $(GENOFFSETS) -s $(CTFSTABS) -r $(CTFCONVERT) \
483 $(CC) $(GOFLAGS) $(CFLAGS64) $(CPPFLAGS)

485 #
486 # tradeoff time for space (smaller is better)
487 #
488 sparc_SPACEFLAG = -xspace -W0,-Lt
489 sparcv9_SPACEFLAG = -xspace -W0,-Lt
490 i386_SPACEFLAG = -xspace
491 amd64_SPACEFLAG =

493 SPACEFLAG = $($MACH)_SPACEFLAG)
494 SPACEFLAG64 = $($MACH64)_SPACEFLAG)

496 #
497 # The Sun Studio 11 compiler has changed the behaviour of integer
498 # wrap arounds and so a flag is needed to use the legacy behaviour
499 # (without this flag panics/hangs could be exposed within the source).
500 #
501 sparc_IROPTFLAG = -W2,-xwrap_int
502 sparcv9_IROPTFLAG = -W2,-xwrap_int
503 i386_IROPTFLAG =
504 amd64_IROPTFLAG =

506 IROPTFLAG = $($MACH)_IROPTFLAG)
507 IROPTFLAG64 = $($MACH64)_IROPTFLAG)

509 sparc_XREGSFLAG = -xregs=no%appl
510 sparcv9_XREGSFLAG = -xregs=no%appl
511 i386_XREGSFLAG =
512 amd64_XREGSFLAG =

514 XREGSFLAG = $($MACH)_XREGSFLAG)
515 XREGSFLAG64 = $($MACH64)_XREGSFLAG)

517 # dmake SOURCEDEBUG=yes ... enables source-level debugging information, and
518 # avoids stripping it.
519 SOURCEDEBUG = $(POUND_SIGN)

```

```

520 SRCDBGBLD      = $(SOURCEDEBUG:yes=)

522 #
523 # These variables are intended ONLY for use by developers to safely pass extra
524 # flags to the compilers without unintentionally overriding Makefile-set
525 # flags.  They should NEVER be set to any value in a Makefile.
526 #
527 # They come last in the associated FLAGS variable such that they can
528 # explicitly override things if necessary, there are gaps in this, but it's
529 # the best we can manage.
530 #
531 CUSERFLAGS      =
532 CUSERFLAGS64    = $(CUSERFLAGS)
533 CCUSERFLAGS     =
534 CCUSERFLAGS64   = $(CCUSERFLAGS)

536 CSOURCEDEBUGFLAGS =
537 CCSOURCEDEBUGFLAGS =
538 $(SRCDBGBLD)CSOURCEDEBUGFLAGS = -g -xs
539 $(SRCDBGBLD)CCSOURCEDEBUGFLAGS = -g -xs

541 CFLAGS=          $(COPTFLAG) $($ (MACH)_CFLAGS) $(SPACEFLAG) $(CCMODE) \
542                 $(ILDOFF) $(CERRWARN) $(C99MODE) $(CCUNBOUND) $(IROPTFLAG) \
543                 $(CGLOBALSTATIC) $(CCNOAUTOINLINE) $(CSOURCEDEBUGFLAGS) \
544                 $(CUSERFLAGS)
545 CFLAGS64=        $(COPTFLAG64) $($ (MACH64)_CFLAGS) $(SPACEFLAG64) $(CCMODE64) \
546                 $(ILDOFF) $(CERRWARN) $(C99MODE) $(CCUNBOUND) $(IROPTFLAG64) \
547                 $(CGLOBALSTATIC) $(CCNOAUTOINLINE) $(CSOURCEDEBUGFLAGS) \
548                 $(CUSERFLAGS64)
549 #
550 # Flags that are used to build parts of the code that are subsequently
551 # run on the build machine (also known as the NATIVE_BUILD).
552 #
553 NATIVE_CFLAGS=   $(COPTFLAG) $($ (NATIVE_MACH)_CFLAGS) $(CCMODE) \
554                 $(ILDOFF) $(CERRWARN) $(C99MODE) $(NATIVE_MACH)_CCUNBOUND) \
555                 $(IROPTFLAG) $(CGLOBALSTATIC) $(CCNOAUTOINLINE) \
556                 $(CSOURCEDEBUGFLAGS) $(CUSERFLAGS)

558 DTEXTDOM=-DTEXT_DOMAIN="\$(TEXT_DOMAIN)"      # For messaging.
559 DTS_ERRNO=-D_TS_ERRNO
560 CPPFLAGS.master=$(DTEXTDOM) $(DTS_ERRNO) \
561                $(ENVCPPFLAGS1) $(ENVCPPFLAGS2) $(ENVCPPFLAGS3) $(ENVCPPFLAGS4) \
562                $(ADJUNCT_PROTO:=-I%/usr/include)
563 CPPFLAGS.native=$(ENVCPPFLAGS1) $(ENVCPPFLAGS2) $(ENVCPPFLAGS3) \
564                 $(ENVCPPFLAGS4) -I$(NATIVE_ADJUNCT)/include
565 CPPFLAGS=        $(CPPFLAGS.master)
566 AS_CPPFLAGS=     $(CPPFLAGS.master)
567 JAVAFLAGS=       -deprecation

569 #
570 # For source message catalogue
571 #
572 .SUFFIXES: $(SUFFIXES) .i .po
573 MSGROOT= $(ROOT)/catalog
574 MSGDOMAIN= $(MSGROOT)/$(TEXT_DOMAIN)
575 MSGDOMAINPOFILE = $(MSGDOMAIN)/$(POFILE)
576 DCMSGDOMAIN= $(MSGROOT)/LC_TIME/$(TEXT_DOMAIN)
577 DCMSGDOMAINPOFILE = $(DCMSGDOMAIN)/$(DCFILE:.dc=.po)

579 CLOBBERFILES += $(POFILE) $(POFILES)
580 COMPILER.cpp= $(CC) -E -C $(CFLAGS) $(CPPFLAGS)
581 XGETTEXT= /usr/bin/xgettext
582 XGETTEXTFLAGS= -c TRANSLATION_NOTE
583 GNUXGETTEXT= /usr/gnu/bin/xgettext
584 GNUXGETTEXTFLAGS= --add-comments=TRANSLATION_NOTE --keyword=_ \
585                 --strict --no-location --omit-header

```

```

586 BUILD.po= $(XGETTEXT) $(XGETTEXTFLAGS) -d $(<F) $<.i ;\
587           $(RM) $@ ;\
588           $(SED) "/^domain/d" < $(<F).po > $@ ;\
589           $(RM) $(<F).po $<.i

591 #
592 # This is overwritten by local Makefile when PROG is a list.
593 #
594 POFILE= $(PROG).po

596 sparc_CCFLAGS=      -cg92 -compat=4 \
597                    -Qoption ccfe -messages=no%anachronism \
598                    $(CCERRWARN)
599 sparcv9_CCFLAGS=    $(sparcv9_XARCH) -dalign -compat=5 \
600                    -Qoption ccfe -messages=no%anachronism \
601                    -Qoption ccfe -features=no%conststrings \
602                    $(CCREGSYM) \
603                    $(CCERRWARN)
604 i386_CCFLAGS=        -compat=4 \
605                    -Qoption ccfe -messages=no%anachronism \
606                    -Qoption ccfe -features=no%conststrings \
607                    $(CCERRWARN)
608 amd64_CCFLAGS=      $(amd64_XARCH) -compat=5 \
609                    -Qoption ccfe -messages=no%anachronism \
610                    -Qoption ccfe -features=no%conststrings \
611                    $(CCERRWARN)

613 sparc_CCOPTFLAG=    -O
614 sparcv9_CCOPTFLAG=  -O
615 i386_CCOPTFLAG=     -O
616 amd64_CCOPTFLAG=    -O

618 CCOPTFLAG=          $($ (MACH)_CCOPTFLAG)
619 CCOPTFLAG64=        $($ (MACH64)_CCOPTFLAG)
620 CCFLAGS=            $($ (MACH)_CCFLAGS) $(CCSOURCEDEBUGFLAGS) \
621                    $(CUSERFLAGS)
622 CCFLAGS64=          $($ (MACH64)_CCFLAGS) $(CCSOURCEDEBUGFLAGS) \
623                    $(CUSERFLAGS64)

625 #
626 #
627 #
628 ELFWRAP_FLAGS =
629 ELFWRAP_FLAGS64 = -64

631 #
632 # Various mapfiles that are used throughout the build, and delivered to
633 # /usr/lib/ld.
634 #
635 MAPFILE.NED_i386 = $(SRC)/common/mapfiles/common/map.noexdata
636 MAPFILE.NED_sparc =
637 MAPFILE.NED = $(MAPFILE.NED_$(MACH))
638 MAPFILE.PGA = $(SRC)/common/mapfiles/common/map.pagealign
639 MAPFILE.NES = $(SRC)/common/mapfiles/common/map.noexstk
640 MAPFILE.FLT = $(SRC)/common/mapfiles/common/map.filter
641 MAPFILE.LEX = $(SRC)/common/mapfiles/common/map.lex.yy

643 #
644 # Generated mapfiles that are compiler specific, and used throughout the
645 # build.  These mapfiles are not delivered in /usr/lib/ld.
646 #
647 MAPFILE.NGB_sparc= $(SRC)/common/mapfiles/gen/sparc_cc_map.noexglobs
648 $(__GNUC64)MAPFILE.NGB_sparc= \
649 $(SRC)/common/mapfiles/gen/sparc_gcc_map.noexglobs
650 MAPFILE.NGB_sparcv9= $(SRC)/common/mapfiles/gen/sparcv9_cc_map.noexglobs
651 $(__GNUC64)MAPFILE.NGB_sparcv9= \

```

```

652 $(SRC)/common/mapfiles/gen/sparcv9_gcc_map.noexeglobs
653 MAPFILE.NGB_i386= $(SRC)/common/mapfiles/gen/i386_cc_map.noexeglobs
654 $(__GNUC64)MAPFILE.NGB_i386= \
655 $(SRC)/common/mapfiles/gen/i386_gcc_map.noexeglobs
656 MAPFILE.NGB_amd64= $(SRC)/common/mapfiles/gen/amd64_cc_map.noexeglobs
657 $(__GNUC64)MAPFILE.NGB_amd64= \
658 $(SRC)/common/mapfiles/gen/amd64_gcc_map.noexeglobs
659 MAPFILE.NGB = $(MAPFILE.NGB_$(MACH))

661 #
662 # A generic interface mapfile name, used by various dynamic objects to define
663 # the interfaces and interposers the object must export.
664 #
665 MAPFILE.INT = mapfile-intf

667 #
668 # LDLIBS32 and LDLIBS64 can be set in the environment to override the following
669 # assignments.
670 #
671 # These environment settings make sure that no libraries are searched outside
672 # of the local workspace proto area:
673 # LDLIBS32=-YP,$ROOT/lib:$ROOT/usr/lib
674 # LDLIBS64=-YP,$ROOT/lib:$MACH64:$ROOT/usr/lib:$MACH64
675 #
676 LDLIBS32 = $(ENVLDLIBS1) $(ENVLDLIBS2) $(ENVLDLIBS3)
677 LDLIBS32 += $(ADJUNCT_PROTO:%=-L%/usr/lib -L%/lib)
678 LDLIBS.cmd = $(LDLIBS32)
679 LDLIBS.lib = $(LDLIBS32)

681 LDLIBS64 = $(ENVLDLIBS1:%=/%$(MACH64)) \
682 $(ENVLDLIBS2:%=/%$(MACH64)) \
683 $(ENVLDLIBS3:%=/%$(MACH64))
684 LDLIBS64 += $(ADJUNCT_PROTO:%=-L%/usr/lib/%$(MACH64) -L%/lib/%$(MACH64))

686 #
687 # Define compilation macros.
688 #
689 COMPILER.c= $(CC) $(CFLAGS) $(CPPFLAGS) -c
690 COMPILER64.c= $(CC) $(CFLAGS64) $(CPPFLAGS) -c
691 COMPILER.cc= $(CCC) $(CCFLAGS) $(CPPFLAGS) -c
692 COMPILER64.cc= $(CCC) $(CCFLAGS64) $(CPPFLAGS) -c
693 COMPILER.s= $(AS) $(ASFLAGS) $(AS_CPPFLAGS)
694 COMPILER64.s= $(AS) $(ASFLAGS) $(MACH64)_AS_XARCH $(AS_CPPFLAGS)
695 COMPILER.d= $(DTRACE) -G -32
696 COMPILER64.d= $(DTRACE) -G -64
697 COMPILER.b= $(ELFWRAP) $(ELFWRAP_FLAGS$(CLASS))
698 COMPILER64.b= $(ELFWRAP) $(ELFWRAP_FLAGS$(CLASS))

700 CLASSPATH= .
701 COMPILER.java= $(JAVAC) $(JAVAFLAGS) -classpath $(CLASSPATH)

703 #
704 # Link time macros
705 #
706 CCNEEDED = -lC
707 CCEXTNEEDED = -lCrun -lCstd
708 $(__GNUC)CCNEEDED = -L$(GCCLIBDIR) -lstc++ -lgcc_s
709 $(__GNUC)CCEXTNEEDED = $(CCNEEDED)

711 LINK.c= $(CC) $(CFLAGS) $(CPPFLAGS) $(LDFLAGS)
712 LINK64.c= $(CC) $(CFLAGS64) $(CPPFLAGS) $(LDFLAGS)
713 NORUNPATH= -norunpath -nolib
714 LINK.cc= $(CCC) $(CCFLAGS) $(CPPFLAGS) $(NORUNPATH) \
715 $(LDFLAGS) $(CCNEEDED)
716 LINK64.cc= $(CCC) $(CCFLAGS64) $(CPPFLAGS) $(NORUNPATH) \
717 $(LDFLAGS) $(CCNEEDED)

```

```

719 #
720 # lint macros
721 #
722 # Note that the undefine of __PRAGMA_REDEFINE_EXTNAME can be removed once
723 # ON is built with a version of lint that has the fix for 4484186.
724 #
725 ALWAYS_LINT_DEFS = -errtags=yes -s
726 ALWAYS_LINT_DEFS += -erroff=E_PTRDIFF_OVERFLOW
727 ALWAYS_LINT_DEFS += -erroff=E_ASSIGN_NARROW_CONV
728 ALWAYS_LINT_DEFS += -U__PRAGMA_REDEFINE_EXTNAME
729 ALWAYS_LINT_DEFS += $(C99LMODE)
730 ALWAYS_LINT_DEFS += -errsecurity=$(SECLEVEL)
731 ALWAYS_LINT_DEFS += -erroff=E_SEC_CREAT_WITHOUT_EXCL
732 ALWAYS_LINT_DEFS += -erroff=E_SEC_FORBIDDEN_WARN_CREAT
733 # XX64 -- really only needed for amd64 lint
734 ALWAYS_LINT_DEFS += -erroff=E_ASSIGN_INT_TO_SMALL_INT
735 ALWAYS_LINT_DEFS += -erroff=E_CAST_INT_CONST_TO_SMALL_INT
736 ALWAYS_LINT_DEFS += -erroff=E_CAST_INT_TO_SMALL_INT
737 ALWAYS_LINT_DEFS += -erroff=E_CAST_TO_PTR_FROM_INT
738 ALWAYS_LINT_DEFS += -erroff=E_COMP_INT_WITH_LARGE_INT
739 ALWAYS_LINT_DEFS += -erroff=E_INTEGRAL_CONST_EXP_EXPECTED
740 ALWAYS_LINT_DEFS += -erroff=E_PASS_INT_TO_SMALL_INT
741 ALWAYS_LINT_DEFS += -erroff=E_PTR_CONV_LOSES_BITS

743 # This forces lint to pick up note.h and sys/note.h from Devpro rather than
744 # from the proto area. The note.h that ON delivers would disable NOTE().
745 ONLY_LINT_DEFS = -I$(SPRO_VROOT)/prod/include/lint

747 SECLEVEL= core
748 LINT.c= $(LINT) $(ONLY_LINT_DEFS) $(LINTFLAGS) $(CPPFLAGS) \
749 $(ALWAYS_LINT_DEFS)
750 LINT64.c= $(LINT) $(ONLY_LINT_DEFS) $(LINTFLAGS64) $(CPPFLAGS) \
751 $(ALWAYS_LINT_DEFS)
752 LINT.s= $(LINT.c)

754 # For some future builds, NATIVE_MACH and MACH might be different.
755 # Therefore, NATIVE_MACH needs to be redefined in the
756 # environment as 'uname -p' to override this macro.
757 #
758 # For now at least, we cross-compile amd64 on i386 machines.
759 NATIVE_MACH= $(MACH:amd64=i386)

761 # Define native compilation macros
762 #

764 # Base directory where compilers are loaded.
765 # Defined here so it can be overridden by developer.
766 #
767 SPRO_ROOT= $(BUILD_TOOLS)/SUNwspro
768 SPRO_VROOT= $(SPRO_ROOT)/SS12
769 GNU_ROOT= $(SFW_ROOT)

771 # Till SS12ul formally becomes the NV CBE, LINT is hard
772 # coded to be picked up from the $SPRO_ROOT/sunstudio12.1/
773 # location. Impacted variables are sparc_LINT, sparcv9_LINT,
774 # i386_LINT, amd64_LINT.
775 # Reset them when SS12ul is rolled out.
776 #

778 # Specify platform compiler versions for languages
779 # that we use (currently only c and c++).
780 #
781 sparc_CC= $(ONBLD_TOOLS)/bin/$(MACH)/cw -_cc
782 $(__GNUC)sparc_CC= $(ONBLD_TOOLS)/bin/$(MACH)/cw -_gcc
783 sparc_CCC= $(ONBLD_TOOLS)/bin/$(MACH)/cw -_CC

```

```

784 $(__GNUCC)sparc_CCC= $(ONBLD_TOOLS)/bin/$(MACH)/cw -_g++
785 sparc_CPP= /usr/ccs/lib/cpp
786 sparc_AS= /usr/ccs/bin/as -xregsym=no
787 sparc_LD= /usr/ccs/bin/ld
788 sparc_LINT= $(SPRO_ROOT)/sunstudio12.1/bin/lint

790 sparcv9_CC= $(ONBLD_TOOLS)/bin/$(MACH)/cw -_cc
791 $(__GNUCC64)sparcv9_CCC= $(ONBLD_TOOLS)/bin/$(MACH)/cw -_gcc
792 sparcv9_CCC= $(ONBLD_TOOLS)/bin/$(MACH)/cw -_CC
793 $(__GNUCC64)sparcv9_CCC= $(ONBLD_TOOLS)/bin/$(MACH)/cw -_g++
794 sparcv9_CPP= /usr/ccs/lib/cpp
795 sparcv9_AS= /usr/ccs/bin/as -xregsym=no
796 sparcv9_LD= /usr/ccs/bin/ld
797 sparcv9_LINT= $(SPRO_ROOT)/sunstudio12.1/bin/lint

799 i386_CC= $(ONBLD_TOOLS)/bin/$(MACH)/cw -_cc
800 $(__GNUCC)i386_CCC= $(ONBLD_TOOLS)/bin/$(MACH)/cw -_gcc
801 i386_CCC= $(ONBLD_TOOLS)/bin/$(MACH)/cw -_CC
802 $(__GNUCC)i386_CCC= $(ONBLD_TOOLS)/bin/$(MACH)/cw -_g++
803 i386_CPP= /usr/ccs/lib/cpp
804 i386_AS= /usr/ccs/bin/as
805 $(__GNUCC)i386_AS= $(ONBLD_TOOLS)/bin/$(MACH)/aw
806 i386_LD= /usr/ccs/bin/ld
807 i386_LINT= $(SPRO_ROOT)/sunstudio12.1/bin/lint

809 amd64_CC= $(ONBLD_TOOLS)/bin/$(MACH)/cw -_cc
810 $(__GNUCC64)amd64_CCC= $(ONBLD_TOOLS)/bin/$(MACH)/cw -_gcc
811 amd64_CCC= $(ONBLD_TOOLS)/bin/$(MACH)/cw -_CC
812 $(__GNUCC64)amd64_CCC= $(ONBLD_TOOLS)/bin/$(MACH)/cw -_g++
813 amd64_CPP= /usr/ccs/lib/cpp
814 amd64_AS= $(ONBLD_TOOLS)/bin/$(MACH)/aw
815 amd64_LD= /usr/ccs/bin/ld
816 amd64_LINT= $(SPRO_ROOT)/sunstudio12.1/bin/lint

818 NATIVECC= $($ (NATIVE_MACH)_CC)
819 NATIVECCC= $($ (NATIVE_MACH)_CCC)
820 NATIVECPP= $($ (NATIVE_MACH)_CPP)
821 NATIVEAS= $($ (NATIVE_MACH)_AS)
822 NATIVELD= $($ (NATIVE_MACH)_LD)
823 NATIVELINT= $($ (NATIVE_MACH)_LINT)

825 #
826 # Makefile.master.64 overrides these settings
827 #
828 CC= $(NATIVECC)
829 CCC= $(NATIVECCC)
830 CPP= $(NATIVECPP)
831 AS= $(NATIVEAS)
832 LD= $(NATIVELD)
833 LINT= $(NATIVELINT)

835 # The real compilers used for this build
836 CW_CC_CMD= $(CC) -_compiler
837 CW_CCC_CMD= $(CCC) -_compiler
838 REAL_CC= $(CW_CC_CMD:sh)
839 REAL_CCC= $(CW_CCC_CMD:sh)

841 # Pass -Y flag to cpp (method of which is release-dependent)
842 CCYFLAG= -Y I,

844 BDIRECT= -Bdirect
845 BDYNAMIC= -Bdynamic
846 BLOCAL= -Blocal
847 BNODIRECT= -Bnodirect
848 BREDUCE= -Breduce
849 BSTATIC= -Bstatic

```

```

851 ZDEFS= -zdefs
852 ZDIRECT= -zdirect
853 ZIGNORE= -zignore
854 ZINITFIRST= -zinitfirst
855 ZINTERPOSE= -zinterpose
856 ZLAZYLOAD= -zlazyload
857 ZLOADFLTR= -zloadfltr
858 ZMULDEFS= -zmuldefs
859 ZNODEFAULTLIB= -zndefaultlib
860 ZNODEFS= -znodefs
861 ZNODELETE= -znodelete
862 ZNODLOPEN= -znodlopen
863 ZNODUMP= -znodump
864 ZNOLAZYLOAD= -znolazyload
865 ZNOLDYNSYM= -znolddynsym
866 ZNORELOC= -znoreloc
867 ZNOVERSION= -znoversion
868 ZRECORD= -zrecord
869 ZREDLOCSYM= -zredlocsymb
870 ZTEXT= -ztext
871 ZVERBOSE= -zverbose

873 GSHARED= -G
874 CCMT= -mt

876 # Handle different PIC models on different ISAs
877 # (May be overridden by lower-level Makefiles)

879 sparc_C_PICFLAGS = -K pic
880 sparcv9_C_PICFLAGS = -K pic
881 i386_C_PICFLAGS = -K pic
882 amd64_C_PICFLAGS = -K pic
883 C_PICFLAGS = $($ (MACH)_C_PICFLAGS)
884 C_PICFLAGS64 = $($ (MACH64)_C_PICFLAGS)

886 sparc_C_BIGPICFLAGS = -K PIC
887 sparcv9_C_BIGPICFLAGS = -K PIC
888 i386_C_BIGPICFLAGS = -K PIC
889 amd64_C_BIGPICFLAGS = -K PIC
890 C_BIGPICFLAGS = $($ (MACH)_C_BIGPICFLAGS)
891 C_BIGPICFLAGS64 = $($ (MACH64)_C_BIGPICFLAGS)

893 # CC requires there to be no space between '-K' and 'pic' or 'PIC'.
894 sparc_CC_PICFLAGS = -Kpic
895 sparcv9_CC_PICFLAGS = -Kpic
896 i386_CC_PICFLAGS = -Kpic
897 amd64_CC_PICFLAGS = -Kpic
898 CC_PICFLAGS = $($ (MACH)_CC_PICFLAGS)
899 CC_PICFLAGS64 = $($ (MACH64)_CC_PICFLAGS)

901 AS_PICFLAGS= $(C_PICFLAGS)
902 AS_BIGPICFLAGS= $(C_BIGPICFLAGS)

904 #
905 # Default label for CTF sections
906 #
907 CTFCVTFLAGS= -i -L VERSION
908 $(SRCDGBLD)CTFCVTFLAGS += -g

910 #
911 # Override to pass module-specific flags to ctfmerge. Currently used only by
912 # krtld to turn on fuzzy matching, and source-level debugging to inhibit
913 # stripping.
914 #
915 CTFMRGFLAGS=

```

```

916 $(SRCDBGBLD)CTFMRGFLAGS      += -g

919 CTFCONVERT_O                  = $(CTFCONVERT) $(CTFCVIFLAGS) $@

921 ELFSIGN_O=                    $(TRUE)
922 ELFSIGN_CRYPTO= $(ELFSIGN_O)
923 ELFSIGN_OBJECT= $(ELFSIGN_O)

925 # Rules (normally from make.rules) and macros which are used for post
926 # processing files. Normally, these do stripping of the comment section
927 # automatically.
928 #   RELEASE_CM:           Should be edited to reflect the release.
929 #   POST_PROCESS_O:      Post-processing for '.o' files.
930 #   POST_PROCESS_A:      Post-processing for '.a' files (currently null).
931 #   POST_PROCESS_SO:     Post-processing for '.so' files.
932 #   POST_PROCESS:        Post-processing for executable files (no suffix).
933 # Note that these macros are not completely generalized as they are to be
934 # used with the file name to be processed following.
935 #
936 # It is left as an exercise to Release Engineering to embellish the generation
937 # of the release comment string.
938 #
939 #   If this is a standard development build:
940 #       compress the comment section (mcs -c)
941 #       add the standard comment (mcs -a $(RELEASE_CM))
942 #       add the development specific comment (mcs -a $(DEV_CM))
943 #
944 #   If this is an installation build:
945 #       delete the comment section (mcs -d)
946 #       add the standard comment (mcs -a $(RELEASE_CM))
947 #       add the development specific comment (mcs -a $(DEV_CM))
948 #
949 #   If this is an release build:
950 #       delete the comment section (mcs -d)
951 #       add the standard comment (mcs -a $(RELEASE_CM))
952 #
953 # The following list of macros are used in the definition of RELEASE_CM
954 # which is used to label all binaries in the build:
955 #
956 #   RELEASE             Specific release of the build, eg: 5.2
957 #   RELEASE_MAJOR       Major version number part of $(RELEASE)
958 #   RELEASE_MINOR       Minor version number part of $(RELEASE)
959 #   VERSION             Version of the build (alpha, beta, Generic)
960 #   PATCHID            If this is a patch this value should contain
961 #                     the patchid value (eg: "Generic 100832-01"), otherwise
962 #                     it will be set to $(VERSION)
963 #   RELEASE_DATE        Date of the Release Build
964 #   PATCH_DATE          Date the patch was created, if this is blank it
965 #                     will default to the RELEASE_DATE
966 #
967 RELEASE_MAJOR= 5
968 RELEASE_MINOR= 11
969 RELEASE= $(RELEASE_MAJOR).$(RELEASE_MINOR)
970 VERSION= SunOS Development
971 PATCHID= $(VERSION)
972 RELEASE_DATE= release date not set
973 PATCH_DATE= $(RELEASE_DATE)
974 RELEASE_CM= "@$(POUND_SIGN)SunOS $(RELEASE) $(PATCHID) $(PATCH_DATE)"
975 DEV_CM= "@$(POUND_SIGN)SunOS Internal Development: non-nightly build"

977 PROCESS_COMMENT= @?${MCS} -d -a $(RELEASE_CM) -a $(DEV_CM)
978 $(RELEASE_BUILD)PROCESS_COMMENT= @?${MCS} -d -a $(RELEASE_CM)

980 STRIP_STABS= :
981 $(RELEASE_BUILD)STRIP_STABS= $(STRIP) -x $@

```

```

982 $(SRCDBGBLD)STRIP_STABS= :

984 POST_PROCESS_O= $(PROCESS_COMMENT) $@
985 POST_PROCESS_A=
986 POST_PROCESS_SO= $(PROCESS_COMMENT) $@ ; $(STRIP_STABS) ; \
987 $(ELFSIGN_OBJECT)
988 POST_PROCESS= $(PROCESS_COMMENT) $@ ; $(STRIP_STABS) ; \
989 $(ELFSIGN_OBJECT)

991 #
992 # chk4ubin is a tool that inspects a module for a symbol table
993 # ELF section size which can trigger an OBP bug on older platforms.
994 # This problem affects only specific sun4u bootable modules.
995 #
996 CHK4UBIN= $(ONBLD_TOOLS)/bin/$(MACH)/chk4ubin
997 CHK4UBINFLAGS=
998 CHK4UBINARY= $(CHK4UBIN) $(CHK4UBINFLAGS) $@

1000 #
1001 # PKGARCHIVE specifies the default location where packages should be
1002 # placed if built.
1003 #
1004 $(RELEASE_BUILD)PKGARCHIVESUFFIX= -nd
1005 PKGARCHIVE=$(SRC)/../../packages/$(MACH)/nightly$(PKGARCHIVESUFFIX)

1007 #
1008 # The repositories will be created with these publisher settings. To
1009 # update an image to the resulting repositories, this must match the
1010 # publisher name provided to "pkg set-publisher."
1011 #
1012 PKGPUBLISHER_REDIST= on-nightly
1013 PKGPUBLISHER_NONREDIST= on-extra

1015 #   Default build rules which perform comment section post-processing.
1016 #
1017 .c:
1018     $(LINK.c) -o $@ $< $(LDLIBS)
1019     $(POST_PROCESS)
1020 .c.o:
1021     $(COMPILE.c) $(OUTPUT_OPTION) $< $(CTFCONVERT_HOOK)
1022     $(POST_PROCESS_O)
1023 .c.a:
1024     $(COMPILE.c) -o $% $<
1025     $(PROCESS_COMMENT) $%
1026     $(AR) $(ARFLAGS) $@ $%
1027     $(RM) $%
1028 .s.o:
1029     $(COMPILE.s) -o $@ $<
1030     $(POST_PROCESS_O)
1031 .s.a:
1032     $(COMPILE.s) -o $% $<
1033     $(PROCESS_COMMENT) $%
1034     $(AR) $(ARFLAGS) $@ $%
1035     $(RM) $%
1036 .cc:
1037     $(LINK.cc) -o $@ $< $(LDLIBS)
1038     $(POST_PROCESS)
1039 .cc.o:
1040     $(COMPILE.cc) $(OUTPUT_OPTION) $<
1041     $(POST_PROCESS_O)
1042 .cc.a:
1043     $(COMPILE.cc) -o $% $<
1044     $(AR) $(ARFLAGS) $@ $%
1045     $(PROCESS_COMMENT) $%
1046     $(RM) $%
1047 .y:

```



```

1048      $(YACC.y) $<
1049      $(LINK.c) -o $@ y.tab.c $(LDLIBS)
1050      $(POST_PROCESS)
1051      $(RM) y.tab.c
1052 .y.o:
1053      $(YACC.y) $<
1054      $(COMPILE.c) -o $@ y.tab.c $(CTFCONVERT_HOOK)
1055      $(POST_PROCESS_0)
1056      $(RM) y.tab.c
1057 .l:
1058      $(RM) $*.c
1059      $(LEX.l) $< > $*.c
1060      $(LINK.c) -o $@ $*.c -ll $(LDLIBS)
1061      $(POST_PROCESS)
1062      $(RM) $*.c
1063 .l.o:
1064      $(RM) $*.c
1065      $(LEX.l) $< > $*.c
1066      $(COMPILE.c) -o $@ $*.c $(CTFCONVERT_HOOK)
1067      $(POST_PROCESS_0)
1068      $(RM) $*.c

1070 .bin.o:
1071      $(COMPILE.b) -o $@ $<
1072      $(POST_PROCESS_0)

1074 .java.class:
1075      $(COMPILE.java) $<

1077 # Bourne and Korn shell script message catalog build rules.
1078 # We extract all gettext strings with sed(1) (being careful to permit
1079 # multiple gettext strings on the same line), weed out the dups, and
1080 # build the catalogue with awk(1).

1082 .sh.po .ksh.po:
1083      $(SED) -n -e ":a" \
1084              -e "h" \
1085              -e "s/.*gettext *\([^\"]*\).*\|/p" \
1086              -e "x" \
1087              -e "s/\(.*\)gettext *\([^\"]*\).*\|/1\2/" \
1088              -e "t a" \
1089      $< | sort -u | awk '{ print "msgid\t" $$0 "\nmsgstr" }' > $@

1091 #
1092 # Python and Perl executable and message catalog build rules.
1093 #
1094 .SUFFIXES: .pl .pm .py .pyc

1096 .pl:
1097      $(RM) $@;
1098      $(SED) -e "s@TEXT_DOMAIN@\"$(TEXT_DOMAIN)\@" $< > $@;
1099      $(CHMOD) +x $@

1101 .py:
1102      $(RM) $@; $(CAT) $< > $@; $(CHMOD) +x $@

1104 .py.pyc:
1105      $(RM) $@
1106      $(PYTHON) -mpy_compile $<
1107      @[ $(<)c = $@ ] || $(MV) $(<)c $@

1109 .py.po:
1110      $(GNUXGETTEXT) $(GNUXGETFLAGS) -d $(<F:%.py=) $< ;

1112 .pl.po .pm.po:
1113      $(XGETTEXT) $(XGETFLAGS) -d $(<F) $< ;

```

```

1114      $(RM) $@ ;
1115      $(SED) "/^domain/d" < $(<F).po > $@ ;
1116      $(RM) $(<F).po

1118 #
1119 # When using xgettext, we want messages to go to the default domain,
1120 # rather than the specified one. This special version of the
1121 # COMPILE.cpp macro effectively prevents expansion of TEXT_DOMAIN,
1122 # causing xgettext to put all messages into the default domain.
1123 #
1124 CPPFORPO=$(COMPILE.cpp:\ "$(TEXT_DOMAIN)\ "=TEXT_DOMAIN)

1126 .c.i:
1127      $(CPPFORPO) $< > $@

1129 .h.i:
1130      $(CPPFORPO) $< > $@

1132 .y.i:
1133      $(YACC) -d $<
1134      $(CPPFORPO) y.tab.c > $@
1135      $(RM) y.tab.c

1137 .l.i:
1138      $(LEX) $<
1139      $(CPPFORPO) lex.yy.c > $@
1140      $(RM) lex.yy.c

1142 .c.po:
1143      $(CPPFORPO) $< > $<.i
1144      $(BUILD.po)

1146 .y.po:
1147      $(YACC) -d $<
1148      $(CPPFORPO) y.tab.c > $<.i
1149      $(BUILD.po)
1150      $(RM) y.tab.c

1152 .l.po:
1153      $(LEX) $<
1154      $(CPPFORPO) lex.yy.c > $<.i
1155      $(BUILD.po)
1156      $(RM) lex.yy.c

1158 #
1159 # Rules to perform stylistic checks
1160 #
1161 .SUFFIXES: .x .xml .check .xmlchk

1163 .h.check:
1164      $(DOT_H_CHECK)

1166 .x.check:
1167      $(DOT_X_CHECK)

1169 .xml.xmlchk:
1170      $(MANIFEST_CHECK)

1172 #
1173 # Include rules to render automated sccs get rules "safe".
1174 #
1175 include $(SRC)/Makefile.noget

```

```

*****
45264 Wed Oct 1 18:40:29 2014
new/usr/src/tools/cw/cw.c
5196 The cw wrapper restricts gcc to -O2
*****
1 /*
2  * CDDL HEADER START
3  *
4  * The contents of this file are subject to the terms of the
5  * Common Development and Distribution License (the "License").
6  * You may not use this file except in compliance with the License.
7  *
8  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
9  * or http://www.opensolaris.org/os/licensing.
10 * See the License for the specific language governing permissions
11 * and limitations under the License.
12 *
13 * When distributing Covered Code, include this CDDL HEADER in each
14 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
15 * If applicable, add the following below this CDDL HEADER, with the
16 * fields enclosed by brackets "[]" replaced with your own identifying
17 * information: Portions Copyright [yyyy] [name of copyright owner]
18 *
19 * CDDL HEADER END
20 */

22 /*
23  * Copyright 2014 Gary Mills
24  *
25  * Copyright 2010 Sun Microsystems, Inc. All rights reserved.
26  * Use is subject to license terms.
27  */

29 /*
30  * Wrapper for the GNU C compiler to make it accept the Sun C compiler
31  * arguments where possible.
32  *
33  * Since the translation is inexact, this is something of a work-in-progress.
34  *
35  */

37 /* If you modify this file, you must increment CW_VERSION */
38 #define CW_VERSION "1.30"
36 #define CW_VERSION "1.29"

40 /*
41  * -# Verbose mode
42  * -### Show compiler commands built by driver, no compilation
43  * -A<name[(tokens)]> Preprocessor predicate assertion
44  * -B[static|dynamic]> Specify dynamic or static binding
45  * -C Prevent preprocessor from removing comments
46  * -c Compile only - produce .o files, suppress linking
47  * -cg92 Alias for -xtarget=ss1000
48  * -D<name=[token]> Associate name with token as if by #define
49  * -d[y|n] dynamic [-dy] or static [-dn] option to linker
50  * -E Compile source through preprocessor only, output to stdout
51  * -errorf=<t> Suppress warnings specified by tags t(%none, %all, <tag list>)
52  * -errtags=<a> Display messages with tags a(no, yes)
53  * -errwarn=<t> Treats warnings specified by tags t(%none, %all, <tag list>)
54  * as errors
55  * -fast Optimize using a selection of options
56  * -fd Report old-style function definitions and declarations
57  * -features=zla Allow zero-length arrays
58  * -flags Show this summary of compiler options
59  * -fnonstd Initialize floating-point hardware to non-standard preferences
60  * -fns[=<yes|no>] Select non-standard floating point mode

```

```

61  * -fprecision=<p> Set FP rounding precision mode p(single, double, extended)
62  * -fround=<r> Select the IEEE rounding mode in effect at startup
63  * -fsimple[=<n>] Select floating-point optimization preferences <n>
64  * -fsingle Use single-precision arithmetic (-Xt and -Xs modes only)
65  * -ftrap=<t> Select floating-point trapping mode in effect at startup
66  * -fstore force floating pt. values to target precision on assignment
67  * -G Build a dynamic shared library
68  * -g Compile for debugging
69  * -H Print path name of each file included during compilation
70  * -h <name> Assign <name> to generated dynamic shared library
71  * -I<dir> Add <dir> to preprocessor #include file search path
72  * -i Passed to linker to ignore any LD_LIBRARY_PATH setting
73  * -keeptmp Keep temporary files created during compilation
74  * -KPIC Compile position independent code with 32-bit addresses
75  * -Kpic Compile position independent code
76  * -L<dir> Pass to linker to add <dir> to the library search path
77  * -l<name> Link with library lib<name>.a or lib<name>.so
78  * -mc Remove duplicate strings from .comment section of output files
79  * -mr Remove all strings from .comment section of output files
80  * -mr,"string" Remove all strings and append "string" to .comment section
81  * -mt Specify options needed when compiling multi-threaded code
82  * -native Find available processor, generate code accordingly
83  * -nofstore Do not force floating pt. values to target precision
84  * on assignment
85  * -nolib Same as -xnolib
86  * -noqueue Disable queuing of compiler license requests
87  * -norunpath Do not build in a runtime path for shared libraries
88  * -O Use default optimization level (-xO2 or -xO3. Check man page.)
89  * -o <outputfile> Set name of output file to <outputfile>
90  * -P Compile source through preprocessor only, output to .i file
91  * -PIC Alias for -KPIC or -xcode=pic32
92  * -p Compile for profiling with prof
93  * -pic Alias for -Kpic or -xcode=pic13
94  * -Q[y|n] Emit/don't emit identification info to output file
95  * -qp Compile for profiling with prof
96  * -R<dir[:dir]> Build runtime search path list into executable
97  * -S Compile and only generate assembly code (.s)
98  * -s Strip symbol table from the executable file
99  * -t Turn off duplicate symbol warnings when linking
100 * -U<name> Delete initial definition of preprocessor symbol <name>
101 * -V Report version number of each compilation phase
102 * -v Do stricter semantic checking
103 * -W<c>,<arg> Pass <arg> to specified component <c> (a,l,m,p,0,2,h,i,u)
104 * -w Suppress compiler warning messages
105 * -Xa Compile assuming ANSI C conformance, allow K & R extensions
106 * (default mode)
107 * -Xc Compile assuming strict ANSI C conformance
108 * -Xs Compile assuming (pre-ANSI) K & R C style code
109 * -Xt Compile assuming K & R conformance, allow ANSI C
110 * -x386 Generate code for the 80386 processor
111 * -x486 Generate code for the 80486 processor
112 * -xarch=<a> Specify target architecture instruction set
113 * -xbuiltin[=<b>] When profitable inline, or substitute intrinsic functions
114 * for system functions, b={%all,%none}
115 * -xCC Accept C++ style comments
116 * -xchar_byte_order=<o> Specify multi-char byte order <o> (default, high, low)
117 * -xchip=<c> Specify the target processor for use by the optimizer
118 * -xcode=<c> Generate different code for forming addresses
119 * -xcrossfile[=<n>] Enable optimization and inlining across source files,
120 * n={0|1}
121 * -xe Perform only syntax/semantic checking, no code generation
122 * -xF Compile for later mapfile reordering or unused section
123 * elimination
124 * -xhelp=<f> Display on-line help information f(flags, readme, errors)
125 * -xildoff Cancel -xildon
126 * -xildon Enable use of the incremental linker, ild

```

```

127 * -xinline=[<a>,...,<a>] Attempt inlining of specified user routines,
128 * <a>={%auto,func,no%func}
129 * -xlibmieee Force IEEE 754 return values for math routines in
130 * exceptional cases
131 * -xlibmil Inline selected libm math routines for optimization
132 * -xlic_lib=sunperf Link in the Sun supplied performance libraries
133 * -xlicinfo Show license server information
134 * -xM Generate makefile dependencies
135 * -xMl Generate makefile dependencies, but exclude /usr/include
136 * -xmaxopt=[off,1,2,3,4,5] maximum optimization level allowed on #pragma opt
137 * -xnolib Do not link with default system libraries
138 * -xnolibmil Cancel -xlibmil on command line
139 * -xO<n> Generate optimized code (n={1|2|3|4|5})
140 * -xP Print prototypes for function definitions
141 * -xpentium Generate code for the pentium processor
142 * -xpg Compile for profiling with gprof
143 * -xprofile=<p> Collect data for a profile or use a profile to optimize
144 * <p>={{collect,use}[:<path>],tcov}
145 * -xregs=<r> Control register allocation
146 * -xs Allow debugging without object (.o) files
147 * -xsb Compile for use with the WorkShop source browser
148 * -xsbfast Generate only WorkShop source browser info, no compilation
149 * -xsfpconst Represent unsuffixed floating point constants as single
150 * precision
151 * -xspace Do not do optimizations that increase code size
152 * -xstrconst Place string literals into read-only data segment
153 * -xtarget=<t> Specify target system for optimization
154 * -xtemp=<dir> Set directory for temporary files to <dir>
155 * -xtime Report the execution time for each compilation phase
156 * -xtransition Emit warnings for differences between K&R C and ANSI C
157 * -xtrigraphs=[<yes|no>] Enable|disable trigraph translation
158 * -xunroll=n Enable unrolling loops n times where possible
159 * -Y<c>,<dir> Specify <dir> for location of component <c> (a,l,m,p,0,h,i,u)
160 * -YA,<dir> Change default directory searched for components
161 * -YI,<dir> Change default directory searched for include files
162 * -YP,<dir> Change default directory for finding libraries files
163 * -YS,<dir> Change default directory for startup object files
164 */

166 /*
167 * Translation table:
168 */
169 /*
170 * -# -v
171 * -### error
172 * -A<name[(tokens)]> pass-thru
173 * -B<[static|dynamic]> pass-thru (syntax error for anything else)
174 * -C pass-thru
175 * -c pass-thru
176 * -cg92 -m32 -mcpu=v8 -mtune=supersparc (SPARC only)
177 * -D<name[=token]> pass-thru
178 * -dy or -dn -Wl,-dy or -Wl,-dn
179 * -E pass-thru
180 * -erroff=E_EMPTY_TRANSLATION_UNIT ignore
181 * -errtags=%all -Wall
182 * -errwarn=%all -Werror else -Wno-error
183 * -fast error
184 * -fd error
185 * -features=zla ignore
186 * -flags --help
187 * -fnonstd error
188 * -fns[=<yes|no>] error
189 * -fprecision=<p> error
190 * -fround=<r> error
191 * -fsimple[=<n>] error
192 * -fsingle[=<n>] error

```

```

193 * -ftrap=<t> error
194 * -fstore error
195 * -G pass-thru
196 * -g pass-thru
197 * -H pass-thru
198 * -h <name> pass-thru
199 * -I<dir> pass-thru
200 * -i pass-thru
201 * -keeptmp -save-temps
202 * -KPIC -fPIC
203 * -Kpic -fpic
204 * -L<dir> pass-thru
205 * -l<name> pass-thru
206 * -mc error
207 * -mr error
208 * -mr,"string" error
209 * -mt -D_REENTRANT
210 * -native error
211 * -nofstore error
212 * -nolib -nodefaultlibs
213 * -noqueue ignore
214 * -norunpath ignore
215 * -O -O1 (Check the man page to be certain)
216 * -o <outputfile> pass-thru
217 * -p -E -o filename.i (or error)
218 * -PIC -fPIC (C++ only)
219 * -p pass-thru
220 * -pic -fpic (C++ only)
221 * -Q[y|n] error
222 * -qp -p
223 * -R<dir[:dir]> pass-thru
224 * -S pass-thru
225 * -s -Wl,-s
226 * -t -Wl,-t
227 * -U<name> pass-thru
228 * -V --version
229 * -v -Wall
230 * -Wa,<arg> pass-thru
231 * -Wp,<arg> pass-thru except -xc99=<a>
232 * -Wl,<arg> pass-thru
233 * -W{m,0,2,h,i,u} error/ignore
234 * -Wu,-xmodel=kernel -ffreestanding -mcmmodel=kernel -mno-red-zone
235 * -xmodel=kernel -ffreestanding -mcmmodel=kernel -mno-red-zone
236 * -Wu,-save_args msave-args
237 * -w pass-thru
238 * -Xa -std=iso9899:199409 or -ansi
239 * -Xc -ansi -pedantic
240 * -Xt error
241 * -Xs -traditional -std=c89
242 * -x386 -march=i386 (x86 only)
243 * -x486 -march=i486 (x86 only)
244 * -xarch=<a> table
245 * -xbuiltin[=<b>] -fbuiltin (-fno-builtin otherwise)
246 * -xCC ignore
247 * -xchar_byte_order=<o> error
248 * -xchip=<c> table
249 * -xcode=<c> table
250 * -xdebugformat=<format> ignore (always use dwarf-2 for gcc)
251 * -xcrossfile[=<n>] ignore
252 * -xe error
253 * -xF error
254 * -xhelp=<f> error
255 * -xildoff ignore
256 * -xildon ignore
257 * -xinline ignore
258 * -xlibmieee error

```

```

259 * -xlibmil          error
260 * -xlic_lib=sunperf error
261 * -xM               -M
262 * -xM1             -MM
263 * -xmaxopt=[...]   error
264 * -xnolib          -ndefaultlibs
265 * -xnolibmil       error
266 * -xO<n>           -O<n>
267 * -xP             error
268 * -xpentium        -march=pentium (x86 only)
269 * -xpg            error
270 * -xprofile=<p>    error
271 * -xregs=<r>      table
272 * -xs            error
273 * -xsb          error
274 * -xsbfast     error
275 * -xsfpconst   error
276 * -xspace      ignore (-not -Os)
277 * -xstrconst   ignore
278 * -xtarget=<t>  table
279 * -xtemp=<dir> error
280 * -xtime       error
281 * -xtransition -Wtransition
282 * -xtrigraphs=<yes|no> -trigraphs -notrigraphs
283 * -xunroll=n    error
284 * -W0,-xdbggen=no%usedonly -fno-eliminate-unused-debug-symbols
285 *                -fno-eliminate-unused-debug-types
286 * -Y<c>,<dir>   error
287 * -YA,<dir>     error
288 * -YI,<dir>     -nostdinc -I<dir>
289 * -YP,<dir>     error
290 * -YS,<dir>     error
291 */

293 #include <stdio.h>
294 #include <sys/types.h>
295 #include <unistd.h>
296 #include <string.h>
297 #include <stdlib.h>
298 #include <ctype.h>
299 #include <fcntl.h>
300 #include <errno.h>
301 #include <stdarg.h>
302 #include <sys/utsname.h>
303 #include <sys/param.h>
304 #include <sys/isa_defs.h>
305 #include <sys/wait.h>
306 #include <sys/stat.h>

308 #define CW_F_CXX      0x01
309 #define CW_F_SHADOW  0x02
310 #define CW_F_EXEC    0x04
311 #define CW_F_ECHO    0x08
312 #define CW_F_XLATE   0x10
313 #define CW_F_PROG    0x20

315 typedef enum cw_compiler {
316     CW_C_CC = 0,
317     CW_C_GCC
318 } cw_compiler_t;

```

unchanged portion omitted

```

537 static void
538 optim_disable(struct aelist *h, int level)
539 {
540     if (level >= 2) {

```

```

541         newae(h, "-fno-strict-aliasing");
542         newae(h, "-fno-unit-at-a-time");
543         newae(h, "-fno-optimize-sibling-calls");
544     }
545 }

539 /* ARGSUSED */
540 static void
541 Xamode(struct aelist *h)
542 {
543 }

```

unchanged portion omitted

```

625 static void
626 do_gcc(cw_ictx_t *ctx)
627 {
628     int c;
629     int pic = 0, nolibs = 0;
630     int in_output = 0, seen_o = 0, c_files = 0;
631     cw_op_t op = CW_O_LINK;
632     char *model = NULL;
633     int mflag = 0;

635     if (ctx->i_flags & CW_F_PROG) {
636         newae(ctx->i_ae, "--version");
637         return;
638     }

640     newae(ctx->i_ae, "-fident");
641     newae(ctx->i_ae, "-finline");
642     newae(ctx->i_ae, "-fno-inline-functions");
643     newae(ctx->i_ae, "-fno-builtin");
644     newae(ctx->i_ae, "-fno-asm");
645     newae(ctx->i_ae, "-fdiagnostics-show-option");
646     newae(ctx->i_ae, "-ndefaultlibs");

648 #if defined(__sparc)
649     /*
650     * The SPARC ldd and std instructions require 8-byte alignment of
651     * their address operand. gcc correctly uses them only when the
652     * ABI requires 8-byte alignment; unfortunately we have a number of
653     * pieces of buggy code that doesn't conform to the ABI. This
654     * flag makes gcc work more like Studio with -xmemalign=4.
655     */
656     newae(ctx->i_ae, "-mno-integer-ldd-std");
657 #endif

659     /*
660     * This is needed because 'u' is defined
661     * under a conditional on 'sun'. Should
662     * probably just remove the conditional,
663     * or make it be dependent on '__sun'.
664     *
665     * -Dunix is also missing in enhanced ANSI mode
666     */
667     newae(ctx->i_ae, "-D__sun");

669     /*
670     * Walk the argument list, translating as we go ..
671     */

673     while (--ctx->i_oldargc > 0) {
674         char *arg = **ctx->i_oldargv;
675         size_t arglen = strlen(arg);

677         if (*arg == '-') {

```

```

678     arglen--;
679 } else {
680     /*
681     * Discard inline files that gcc doesn't grok
682     */
683     if (!in_output && arglen > 3 &&
684         strcmp(arg + arglen - 3, ".il") == 0)
685         continue;
687     if (!in_output && arglen > 2 &&
688         arg[arglen - 2] == '.' &&
689         (arg[arglen - 1] == 'S' || arg[arglen - 1] == 's' ||
690          arg[arglen - 1] == 'c' || arg[arglen - 1] == 'i'))
691         c_files++;
693     /*
694     * Otherwise, filenames and partial arguments
695     * are passed through for gcc to chew on. However,
696     * output is always discarded for the secondary
697     * compiler.
698     */
699     if ((ctx->i_flags & CW_F_SHADOW) && in_output)
700         newae(ctx->i_ae, ctx->i_discard);
701     else
702         newae(ctx->i_ae, arg);
703     in_output = 0;
704     continue;
705 }
707 if (ctx->i_flags & CW_F_CXX) {
708     if (strncmp(arg, "-compat=", 8) == 0) {
709         /* discard -compat=4 and -compat=5 */
710         continue;
711     }
712     if (strcmp(arg, "-Qoption") == 0) {
713         /* discard -Qoption and its two arguments */
714         if (ctx->i_oldargc < 3)
715             error(arg);
716         ctx->i_oldargc -= 2;
717         ctx->i_oldargv += 2;
718         continue;
719     }
720     if (strcmp(arg, "-xwe") == 0) {
721         /* turn warnings into errors */
722         newae(ctx->i_ae, "-Werror");
723         continue;
724     }
725     if (strcmp(arg, "-noex") == 0) {
726         /* no exceptions */
727         newae(ctx->i_ae, "-fno-exceptions");
728         /* no run time type descriptor information */
729         newae(ctx->i_ae, "-fno-rtti");
730         continue;
731     }
732     if (strcmp(arg, "-pic") == 0) {
733         newae(ctx->i_ae, "-fpic");
734         pic = 1;
735         continue;
736     }
737     if (strcmp(arg, "-PIC") == 0) {
738         newae(ctx->i_ae, "-fPIC");
739         pic = 1;
740         continue;
741     }
742     if (strcmp(arg, "-norunpath") == 0) {
743         /* gcc has no corresponding option */

```

```

744         continue;
745     }
746     if (strcmp(arg, "-nolib") == 0) {
747         /* -nodefaultlibs is on by default */
748         nolibc = 1;
749         continue;
750     }
751     #if defined(__sparc)
752     if (strcmp(arg, "-cg92") == 0) {
753         mflag |= xlate_xtb(ctx->i_ae, "v8");
754         xlate(ctx->i_ae, "super", xchip_tbl);
755         continue;
756     }
757     #endif /* __sparc */
758 }
760 switch ((c = arg[1])) {
761 case '-':
762     if (strcmp(arg, "-noecho") == 0)
763         ctx->i_flags &= ~CW_F_ECHO;
764     else if (strncmp(arg, "-cc=", 5) == 0 ||
765             strncmp(arg, "-_CC=", 5) == 0)
766         /* EMPTY */;
767     else if (strncmp(arg, "-gcc=", 6) == 0 ||
768             strncmp(arg, "-g++=", 6) == 0)
769         newae(ctx->i_ae, arg + 6);
770     else
771         error(arg);
772     break;
773 case '#':
774     if (arglen == 1) {
775         newae(ctx->i_ae, "-v");
776         break;
777     }
778     error(arg);
779     break;
780 case 'g':
781     newae(ctx->i_ae, "-gdwarf-2");
782     break;
783 case 'E':
784     if (arglen == 1) {
785         newae(ctx->i_ae, "-xc");
786         newae(ctx->i_ae, arg);
787         op = CW_O_PREPROCESS;
788         nolibc = 1;
789         break;
790     }
791     error(arg);
792     break;
793 case 'c':
794 case 'S':
795     if (arglen == 1) {
796         op = CW_O_COMPILE;
797         nolibc = 1;
798     }
799     /* FALLTHROUGH */
800 case 'C':
801 case 'H':
802 case 'p':
803     if (arglen == 1) {
804         newae(ctx->i_ae, arg);
805         break;
806     }
807     error(arg);
808     break;
809 case 'A':

```

```

810     case 'h':
811     case 'I':
812     case 'i':
813     case 'L':
814     case 'l':
815     case 'R':
816     case 'U':
817     case 'u':
818     case 'w':
819         newae(ctx->i_ae, arg);
820         break;
821     case 'o':
822         seen_o = 1;
823         if (arglen == 1) {
824             in_output = 1;
825             newae(ctx->i_ae, arg);
826         } else if (ctx->i_flags & CW_F_SHADOW) {
827             newae(ctx->i_ae, "-o");
828             newae(ctx->i_ae, ctx->i_discard);
829         } else {
830             newae(ctx->i_ae, arg);
831         }
832         break;
833     case 'D':
834         newae(ctx->i_ae, arg);
835         /*
836          * XXX Clearly a hack ... do we need _KADB too?
837          */
838         if (strcmp(arg, "-D_KERNEL") == 0 ||
839             strcmp(arg, "-D_BOOT") == 0)
840             newae(ctx->i_ae, "-ffreestanding");
841         break;
842     case 'd':
843         if (arglen == 2) {
844             if (strcmp(arg, "-dy") == 0) {
845                 newae(ctx->i_ae, "-Wl,-dy");
846                 break;
847             }
848             if (strcmp(arg, "-dn") == 0) {
849                 newae(ctx->i_ae, "-Wl,-dn");
850                 break;
851             }
852         }
853         if (strcmp(arg, "-dalign") == 0) {
854             /*
855              * -dalign forces alignment in some cases;
856              * gcc does not need any flag to do this.
857              */
858             break;
859         }
860         error(arg);
861         break;
862     case 'e':
863         if (strcmp(arg,
864             "--errorf=E_EMPTY_TRANSLATION_UNIT") == 0) {
865             /*
866              * Accept but ignore this -- gcc doesn't
867              * seem to complain about empty translation
868              * units
869              */
870             break;
871         }
872         /* XX64 -- ignore all -errorf= options, for now */
873         if (strncmp(arg, "--errorf=", 8) == 0)
874             break;
875         if (strcmp(arg, "-errtags=yes") == 0) {

```

```

876             warnings(ctx->i_ae);
877             break;
878         }
879         if (strcmp(arg, "-errwarn=all") == 0) {
880             newae(ctx->i_ae, "-Werror");
881             break;
882         }
883         error(arg);
884         break;
885     case 'f':
886         if (strcmp(arg, "-flags") == 0) {
887             newae(ctx->i_ae, "--help");
888             break;
889         }
890         if (strncmp(arg, "--features=zla", 13) == 0) {
891             /*
892              * Accept but ignore this -- gcc allows
893              * zero length arrays.
894              */
895             break;
896         }
897         error(arg);
898         break;
899     case 'G':
900         newae(ctx->i_ae, "-shared");
901         nolibc = 1;
902         break;
903     case 'k':
904         if (strcmp(arg, "-keeptmp") == 0) {
905             newae(ctx->i_ae, "-save-temps");
906             break;
907         }
908         error(arg);
909         break;
910     case 'K':
911         if (arglen == 1) {
912             if ((arg = **++ctx->i_oldargv) == NULL ||
913                 *arg == '\0')
914                 error("-K");
915             ctx->i_oldargc--;
916         } else {
917             arg += 2;
918         }
919         if (strcmp(arg, "pic") == 0) {
920             newae(ctx->i_ae, "-fpic");
921             pic = 1;
922             break;
923         }
924         if (strcmp(arg, "PIC") == 0) {
925             newae(ctx->i_ae, "-fPIC");
926             pic = 1;
927             break;
928         }
929         error("-K");
930         break;
931     case 'm':
932         if (strcmp(arg, "-mt") == 0) {
933             newae(ctx->i_ae, "-D_REENTRANT");
934             break;
935         }
936         if (strcmp(arg, "-m64") == 0) {
937             newae(ctx->i_ae, "-m64");
938             #if defined(__x86)
939             newae(ctx->i_ae, "-mtune=opteron");
940             #endif
941             mflag |= M64;

```

```

942         break;
943     }
944     if (strcmp(arg, "-m32") == 0) {
945         newae(ctx->i_ae, "-m32");
946         mflag |= M32;
947         break;
948     }
949     error(arg);
950     break;
951 case 'B': /* linker options */
952 case 'M':
953 case 'z':
954     {
955         char *opt;
956         size_t len;
957         char *s;
958
959         if (arglen == 1) {
960             opt = *++ctx->i_oldargv;
961             if (opt == NULL || *opt == '\\0')
962                 error(arg);
963             ctx->i_oldargc--;
964         } else {
965             opt = arg + 2;
966         }
967         len = strlen(opt) + 7;
968         if ((s = malloc(len)) == NULL)
969             nomem();
970         (void) snprintf(s, len, "-Wl,-%c%s", c, opt);
971         newae(ctx->i_ae, s);
972         free(s);
973     }
974     break;
975 case 'n':
976     if (strcmp(arg, "-noqueue") == 0) {
977         /*
978          * Horrid license server stuff - n/a
979          */
980         break;
981     }
982     error(arg);
983     break;
984 case 'O':
985     if (arglen == 1) {
986         newae(ctx->i_ae, "-O");
987         break;
988     }
989     error(arg);
990     break;
991 case 'P':
992     /*
993     * We could do '-E -o filename.i', but that's hard,
994     * and we don't need it for the case that's triggering
995     * this addition. We'll require the user to specify
996     * -o in the Makefile. If they don't they'll find out
997     * in a hurry.
998     */
999     newae(ctx->i_ae, "-E");
1000     op = CW_O_PREPROCESS;
1001     nolibc = 1;
1002     break;
1003 case 'q':
1004     if (strcmp(arg, "-qp") == 0) {
1005         newae(ctx->i_ae, "-p");
1006         break;
1007     }

```

```

1008         error(arg);
1009         break;
1010 case 's':
1011     if (arglen == 1) {
1012         newae(ctx->i_ae, "-Wl,-s");
1013         break;
1014     }
1015     error(arg);
1016     break;
1017 case 't':
1018     if (arglen == 1) {
1019         newae(ctx->i_ae, "-Wl,-t");
1020         break;
1021     }
1022     error(arg);
1023     break;
1024 case 'V':
1025     if (arglen == 1) {
1026         ctx->i_flags &= ~CW_F_ECHO;
1027         newae(ctx->i_ae, "--version");
1028         break;
1029     }
1030     error(arg);
1031     break;
1032 case 'v':
1033     if (arglen == 1) {
1034         warnings(ctx->i_ae);
1035         break;
1036     }
1037     error(arg);
1038     break;
1039 case 'W':
1040     if (strncmp(arg, "-Wp,-xc99", 9) == 0) {
1041         /*
1042          * gcc's preprocessor will accept c99
1043          * regardless, so accept and ignore.
1044          */
1045         break;
1046     }
1047     if (strncmp(arg, "-Wa,", 4) == 0 ||
1048         strncmp(arg, "-Wp,", 4) == 0 ||
1049         strncmp(arg, "-Wl,", 4) == 0) {
1050         newae(ctx->i_ae, arg);
1051         break;
1052     }
1053     if (strcmp(arg, "-W0,-xc99=pragma") == 0) {
1054         /* (undocumented) enables _Pragma */
1055         break;
1056     }
1057     if (strcmp(arg, "-W0,-xc99=%none") == 0) {
1058         /*
1059          * This is a polite way of saying
1060          * "no c99 constructs allowed!"
1061          * For now, just accept and ignore this.
1062          */
1063         break;
1064     }
1065     if (strcmp(arg, "-W0,-noglobal") == 0 ||
1066         strcmp(arg, "-W0,-xglobalstatic") == 0) {
1067         /*
1068          * gcc doesn't prefix local symbols
1069          * in debug mode, so this is not needed.
1070          */
1071         break;
1072     }
1073     if (strcmp(arg, "-W0,-Lt") == 0) {

```

```

1074     /*
1075     * Generate tests at the top of loops.
1076     * There is no direct gcc equivalent, ignore.
1077     */
1078     break;
1079 }
1080 if (strcmp(arg, "-W0,-xdbggen=no%usedonly") == 0) {
1081     newae(ctx->i_ae,
1082         "-fno-eliminate-unused-debug-symbols");
1083     newae(ctx->i_ae,
1084         "-fno-eliminate-unused-debug-types");
1085     break;
1086 }
1087 if (strcmp(arg, "-W2,-xwrap_int") == 0) {
1088     /*
1089     * Use the legacy behaviour (pre-SS11)
1090     * for integer wrapping.
1091     * gcc does not need this.
1092     */
1093     break;
1094 }
1095 if (strcmp(arg, "-W2,-Rcond_elim") == 0) {
1096     /*
1097     * Elimination and expansion of conditionals;
1098     * gcc has no direct equivalent.
1099     */
1100     break;
1101 }
1102 if (strcmp(arg, "-Wd,-xsafe=unboundsym") == 0) {
1103     /*
1104     * Prevents optimizing away checks for
1105     * unbound weak symbol addresses. gcc does
1106     * not do this, so it's not needed.
1107     */
1108     break;
1109 }
1110 if (strncmp(arg, "-Wc,-xcode=", 11) == 0) {
1111     xlate(ctx->i_ae, arg + 11, xcode_tbl);
1112     if (strncmp(arg + 11, "pic", 3) == 0)
1113         pic = 1;
1114     break;
1115 }
1116 if (strncmp(arg, "-Wc,-Qiselect", 13) == 0) {
1117     /*
1118     * Prevents insertion of register symbols.
1119     * gcc doesn't do this, so ignore it.
1120     */
1121     break;
1122 }
1123 if (strcmp(arg, "-Wc,-Qassembler-ounrefsym=0") == 0) {
1124     /*
1125     * Prevents optimizing away of static variables.
1126     * gcc does not do this, so it's not needed.
1127     */
1128     break;
1129 }
1130 #if defined(__x86)
1131 if (strcmp(arg, "-Wu,-xmodel=kernel") == 0) {
1132     newae(ctx->i_ae, "-ffreestanding");
1133     newae(ctx->i_ae, "-mno-red-zone");
1134     model = "-mcmmodel=kernel";
1135     nolibc = 1;
1136     break;
1137 }
1138 if (strcmp(arg, "-Wu,-save_args") == 0) {
1139     newae(ctx->i_ae, "-msave-args");

```

```

1140         break;
1141     }
1142 #endif /* __x86 */
1143     error(arg);
1144     break;
1145 case 'X':
1146     if (strcmp(arg, "-Xa") == 0 ||
1147         strcmp(arg, "-Xt") == 0) {
1148         Xamode(ctx->i_ae);
1149         break;
1150     }
1151     if (strcmp(arg, "-Xc") == 0) {
1152         Xcmode(ctx->i_ae);
1153         break;
1154     }
1155     if (strcmp(arg, "-Xs") == 0) {
1156         Xsmode(ctx->i_ae);
1157         break;
1158     }
1159     error(arg);
1160     break;
1161 case 'x':
1162     if (arglen == 1)
1163         error(arg);
1164     switch (arg[2]) {
1165 #if defined(__x86)
1166     case '3':
1167         if (strcmp(arg, "-x386") == 0) {
1168             newae(ctx->i_ae, "-march=i386");
1169             break;
1170         }
1171         error(arg);
1172         break;
1173     case '4':
1174         if (strcmp(arg, "-x486") == 0) {
1175             newae(ctx->i_ae, "-march=i486");
1176             break;
1177         }
1178         error(arg);
1179         break;
1180 #endif /* __x86 */
1181     case 'a':
1182         if (strncmp(arg, "-xarch=", 7) == 0) {
1183             mflag |= xlate_xtb(ctx->i_ae, arg + 7);
1184             break;
1185         }
1186         error(arg);
1187         break;
1188     case 'b':
1189         if (strncmp(arg, "-xbuiltin=", 10) == 0) {
1190             if (strcmp(arg + 10, "%all")
1191                 newae(ctx->i_ae, "-fbuiltin");
1192             break;
1193         }
1194         error(arg);
1195         break;
1196     case 'C':
1197         /* Accept C++ style comments -- ignore */
1198         if (strcmp(arg, "-xCC") == 0)
1199             break;
1200         error(arg);
1201         break;
1202     case 'c':
1203         if (strncmp(arg, "-xc99=%all", 10) == 0) {
1204             newae(ctx->i_ae, "-std=gnu99");
1205             break;

```



```

1206     }
1207     if (strncmp(arg, "-xc99=%none", 11) == 0) {
1208         newae(ctx->i_ae, "-std=gnu89");
1209         break;
1210     }
1211     if (strncmp(arg, "-xchip=", 7) == 0) {
1212         xlate(ctx->i_ae, arg + 7, xchip_tbl);
1213         break;
1214     }
1215     if (strncmp(arg, "-xcode=", 7) == 0) {
1216         xlate(ctx->i_ae, arg + 7, xcode_tbl);
1217         if (strncmp(arg + 7, "pic", 3) == 0)
1218             pic = 1;
1219         break;
1220     }
1221     if (strncmp(arg, "-xcache=", 8) == 0)
1222         break;
1223     if (strncmp(arg, "-xcrossfile", 11) == 0)
1224         break;
1225     error(arg);
1226     break;
1227 case 'd':
1228     if (strcmp(arg, "-xdepend") == 0)
1229         break;
1230     if (strncmp(arg, "-xdebugformat=", 14) == 0)
1231         break;
1232     error(arg);
1233     break;
1234 case 'F':
1235     /*
1236      * Compile for mapfile reordering, or unused
1237      * section elimination, syntax can be -xF or
1238      * more complex, like -xF=%all -- ignore.
1239      */
1240     if (strncmp(arg, "-xF", 3) == 0)
1241         break;
1242     error(arg);
1243     break;
1244 case 'i':
1245     if (strncmp(arg, "-xinline", 8) == 0)
1246         /* No inlining; ignore */
1247         break;
1248     if (strcmp(arg, "-xildon") == 0 ||
1249         strcmp(arg, "-xildoff") == 0)
1250         /* No incremental linking; ignore */
1251         break;
1252     error(arg);
1253     break;
1254 #if defined(__x86)
1255 case 'm':
1256     if (strcmp(arg, "-xmodel=kernel") == 0) {
1257         newae(ctx->i_ae, "-ffreestanding");
1258         newae(ctx->i_ae, "-mno-red-zone");
1259         model = "-mcmmodel=kernel";
1260         nolibc = 1;
1261         break;
1262     }
1263     error(arg);
1264     break;
1265 #endif /* __x86 */
1266 case 'M':
1267     if (strcmp(arg, "-xM") == 0) {
1268         newae(ctx->i_ae, "-M");
1269         break;
1270     }
1271     if (strcmp(arg, "-xM1") == 0) {

```

```

1272         newae(ctx->i_ae, "-MM");
1273         break;
1274     }
1275     error(arg);
1276     break;
1277 case 'n':
1278     if (strcmp(arg, "-xnolib") == 0) {
1279         nolibc = 1;
1280         break;
1281     }
1282     error(arg);
1283     break;
1284 case 'O':
1285     if (strncmp(arg, "-xO", 3) == 0) {
1286         size_t len = strlen(arg);
1287         char *s;
1288         int c = *(arg + 3);
1289         int level;
1290
1291         if (len != 4 || !isdigit(c))
1292             error(arg);
1293
1294         if ((s = malloc(len)) == NULL)
1295             nomem();
1296
1297         level = atoi(arg + 3);
1298         if (level > 5)
1299             error(arg);
1300         if (level >= 2) {
1301             /*
1302              * For gcc-3.4.x at -O2 we
1303              * need to disable optimizations
1304              * that break ON.
1305              */
1306             optim_disable(ctx->i_ae, level);
1307             /*
1308              * limit -xO3 to -O2 as well.
1309              */
1310             level = 2;
1311         }
1312         (void) snprintf(s, len, "-O%d", level);
1313         newae(ctx->i_ae, s);
1314         free(s);
1315         break;
1316     }
1317     error(arg);
1318     break;
1319 case 'p':
1320     if (strcmp(arg, "-xpentium") == 0) {
1321         newae(ctx->i_ae, "-march=pentium");
1322         break;
1323     }
1324     if (strcmp(arg, "-xpg") == 0) {
1325         newae(ctx->i_ae, "-pg");
1326         break;
1327     }
1328     error(arg);
1329     break;
1330 case 'r':
1331     if (strncmp(arg, "-xregs=", 7) == 0) {
1332         xlate(ctx->i_ae, arg + 7, xregs_tbl);
1333         break;
1334     }
1335     error(arg);
1336     break;
1337 case 's':

```

```

1326         if (strcmp(arg, "-xs") == 0 ||
1327             strcmp(arg, "-xspace") == 0 ||
1328             strcmp(arg, "-xstrconst") == 0)
1329             break;
1330         error(arg);
1331         break;
1332     case 't':
1333         if (strcmp(arg, "-xtransition") == 0) {
1334             newae(ctx->i_ae, "-Wtransition");
1335             break;
1336         }
1337         if (strcmp(arg, "-xtrigraphs=yes") == 0) {
1338             newae(ctx->i_ae, "-trigraphs");
1339             break;
1340         }
1341         if (strcmp(arg, "-xtrigraphs=no") == 0) {
1342             newae(ctx->i_ae, "-notrigraphs");
1343             break;
1344         }
1345         if (strncmp(arg, "-xtarget=", 9) == 0) {
1346             xlate(ctx->i_ae, arg + 9, xtarget_tbl);
1347             break;
1348         }
1349         error(arg);
1350         break;
1351     case 'e':
1352     case 'h':
1353     case 'l':
1354     default:
1355         error(arg);
1356         break;
1357     }
1358     break;
1359 case 'Y':
1360     if (arglen == 1) {
1361         if ((arg = **++ctx->i_oldargv) == NULL ||
1362             *arg == '\0')
1363             error("-Y");
1364         ctx->i_oldargc--;
1365         arglen = strlen(arg + 1);
1366     } else {
1367         arg += 2;
1368     }
1369     /* Just ignore -YS,... for now */
1370     if (strncmp(arg, "S,", 2) == 0)
1371         break;
1372     if (strncmp(arg, "l,", 2) == 0) {
1373         char *s = strdup(arg);
1374         s[0] = '-';
1375         s[1] = 'B';
1376         newae(ctx->i_ae, s);
1377         free(s);
1378         break;
1379     }
1380     if (strncmp(arg, "I,", 2) == 0) {
1381         char *s = strdup(arg);
1382         s[0] = '-';
1383         s[1] = 'I';
1384         newae(ctx->i_ae, "-nostdinc");
1385         newae(ctx->i_ae, s);
1386         free(s);
1387         break;
1388     }
1389     error(arg);
1390     break;
1391 case 'Q':

```

```

1392         /*
1393         * We could map -Qy into -Wl,-Qy etc.
1394         */
1395     default:
1396         error(arg);
1397         break;
1398     }
1399 }
1400
1401 if (c_files > 1 && (ctx->i_flags & CW_F_SHADOW) &&
1402     op != CW_O_PREPROCESS) {
1403     (void) fprintf(stderr, "%s: error: multiple source files are "
1404                    "allowed only with -E or -P\n", progname);
1405     exit(2);
1406 }
1407
1408 /*
1409 * Make sure that we do not have any unintended interactions between
1410 * the xarch options passed in and the version of the Studio compiler
1411 * used.
1412 */
1413 if ((mflag & (SS11|SS12)) == (SS11|SS12)) {
1414     (void) fprintf(stderr,
1415                    "Conflicting \"-xarch=\" flags (both Studio 11 and 12)\n");
1416     exit(2);
1417 }
1418
1419 switch (mflag) {
1420     case 0:
1421         /* FALLTHROUGH */
1422     case M32:
1423         #if defined(__sparc)
1424             /*
1425              * Only -m32 is defined and so put in the missing xarch
1426              * translation.
1427              */
1428             newae(ctx->i_ae, "-mcpu=v8");
1429             newae(ctx->i_ae, "-mno-v8plus");
1430         #endif
1431         break;
1432     case M64:
1433         #if defined(__sparc)
1434             /*
1435              * Only -m64 is defined and so put in the missing xarch
1436              * translation.
1437              */
1438             newae(ctx->i_ae, "-mcpu=v9");
1439         #endif
1440         break;
1441     case SS12:
1442         #if defined(__sparc)
1443             /* no -m32/-m64 flag used - this is an error for sparc builds */
1444             (void) fprintf(stderr, "No -m32/-m64 flag defined\n");
1445             exit(2);
1446         #endif
1447         break;
1448     case SS11:
1449         /* FALLTHROUGH */
1450     case (SS11|M32):
1451     case (SS11|M64):
1452         break;
1453     case (SS12|M32):
1454     #if defined(__sparc)
1455         /*
1456          * Need to add in further 32 bit options because with SS12
1457          * the xarch=sparcvis option can be applied to 32 or 64

```

```
1458         * bit, and so the translation table (xtbl) cannot handle
1459         * that.
1460         */
1461         newae(ctx->i_ae, "-mv8plus");
1462 #endif
1463         break;
1464     case (SSI2|M64):
1465         break;
1466     default:
1467         (void) fprintf(stderr,
1468             "Incompatible -xarch= and/or -m32/-m64 options used.\n");
1469         exit(2);
1470     }
1471     if (op == CW_O_LINK && (ctx->i_flags & CW_F_SHADOW))
1472         exit(0);
1473
1474     if (model && !pic)
1475         newae(ctx->i_ae, model);
1476     if (!nolibc)
1477         newae(ctx->i_ae, "-lc");
1478     if (!seen_o && (ctx->i_flags & CW_F_SHADOW)) {
1479         newae(ctx->i_ae, "-o");
1480         newae(ctx->i_ae, ctx->i_discard);
1481     }
1482 }
unchanged_portion_omitted
```

```

*****
16093 Wed Oct 1 18:40:29 2014
new/usr/src/uts/intel/Makefile.intel
5196 The cw wrapper restricts gcc to -O2
*****
1 # CDDL HEADER START
2 #
3 # The contents of this file are subject to the terms of the
4 # Common Development and Distribution License (the "License").
5 # You may not use this file except in compliance with the License.
6 #
7 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
8 # or http://www.opensolaris.org/os/licensing.
9 # See the License for the specific language governing permissions
10 # and limitations under the License.
11 #
12 # When distributing Covered Code, include this CDDL HEADER in each
13 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
14 # If applicable, add the following below this CDDL HEADER, with the
15 # fields enclosed by brackets "[]" replaced with your own identifying
16 # information: Portions Copyright [yyyy] [name of copyright owner]
17 #
18 # CDDL HEADER END
19 #
21 # Copyright (c) 2005, 2010, Oracle and/or its affiliates. All rights reserved.
22 # Copyright (c) 2012 Nexenta Systems, Inc. All rights reserved.
23 # Copyright (c) 2013 Andrew Stormont. All rights reserved.
24 # Copyright 2014 Gary Mills
25 #
26 #
27 # This makefile contains the common definitions for all intel
28 # implementation architecture independent modules.
29 #
31 #
32 # Machine type (implementation architecture):
33 #
34 PLATFORM = i86pc
35 #
36 #
37 # Everybody needs to know how to build modstubs.o and to locate unix.o.
38 # Note that unix.o must currently be selected from among the possible
39 # "implementation architectures". Note further, that unix.o is only
40 # used as an optional error check for undefines so (theoretically)
41 # any "implementation architectures" could be used. We choose i86pc
42 # because it is the reference port.
43 #
44 UNIX_DIR = $(UTSBASE)/i86pc/unix
45 GENLIB_DIR = $(UTSBASE)/intel/genunix
46 IPDRV_DIR = $(UTSBASE)/intel/ip
47 MODSTUBS_DIR = $(UNIX_DIR)
48 DSF_DIR = $(UTSBASE)/$(PLATFORM)/genassym
49 LINTS_DIR = $(OBJS_DIR)
50 LINT_LIB_DIR = $(UTSBASE)/intel/lint-libs/$(OBJS_DIR)
51 #
52 UNIX_O = $(UNIX_DIR)/$(OBJS_DIR)/unix.o
53 GENLIB = $(GENLIB_DIR)/$(OBJS_DIR)/libgenunix.so
54 MODSTUBS_O = $(MODSTUBS_DIR)/$(OBJS_DIR)/modstubs.o
55 LINT_LIB = $(UTSBASE)/i86pc/lint-libs/$(OBJS_DIR)/llib-lunix.ln
56 GEN_LINT_LIB = $(UTSBASE)/intel/lint-libs/$(OBJS_DIR)/llib-lgenunix.ln
57 #
58 #
59 # Include the makefiles which define build rule templates, the
60 # collection of files per module, and a few specific flags. Note
61 # that order is significant, just as with an include path. The

```

```

62 # first build rule template which matches the files name will be
63 # used. By including these in order from most machine dependent
64 # to most machine independent, we allow a machine dependent file
65 # to be used in preference over a machine independent version
66 # (Such as a machine specific optimization, which preserves the
67 # interfaces.)
68 #
69 include $(UTSBASE)/intel/Makefile.files
70 include $(UTSBASE)/common/Makefile.files
71 #
72 #
73 # ----- TRANSITIONAL SECTION -----
74 #
75 #
76 #
77 # Not everything which *should* be a module is a module yet. The
78 # following is a list of such objects which are currently part of
79 # genunix but which might someday become kmods. This must be
80 # defined before we include Makefile.uts, or else genunix's build
81 # won't be as parallel as we might like.
82 #
83 NOT_YET_KMODS = $(OLDPTY_OBJS) $(PTY_OBJS) $(VCONS_CONF_OBJS) $(MOD_OBJS)
84 #
85 #
86 # ----- END OF TRANSITIONAL SECTION -----
87 #
88 # Include machine independent rules. Note that this does not imply
89 # that the resulting module from rules in Makefile.uts is machine
90 # independent. Only that the build rules are machine independent.
91 #
92 include $(UTSBASE)/Makefile.uts
93 #
94 #
95 # The following must be defined for all implementations:
96 #
97 MODSTUBS = $(UTSBASE)/intel/ia32/ml/modstubs.s
98 #
99 #
100 # Define supported builds
101 #
102 DEF_BUILDS = $(DEF_BUILDS64) $(DEF_BUILDS32)
103 ALL_BUILDS = $(ALL_BUILDS64) $(ALL_BUILDS32)
104 #
105 #
106 # x86 or amd64 inline templates
107 #
108 INLINES_32 = $(UTSBASE)/intel/ia32/ml/ia32.il
109 INLINES_64 = $(UTSBASE)/intel/amd64/ml/amd64.il
110 INLINES += $(INLINES_$(CLASS))
111 #
112 #
113 # kernel-specific optimizations; override default in Makefile.master
114 #
115 #
116 CFLAGS_XARCH_32 = $(i386_CFLAGS)
117 CFLAGS_XARCH_64 = $(amd64_CFLAGS)
118 CFLAGS_XARCH = $(CFLAGS_XARCH_$(CLASS))
119 #
120 COPTFLAG_32 = -_gcc=-fno-toplevel-reorder $(COPTFLAG)
121 COPTFLAG_64 = -_gcc=-fno-toplevel-reorder $(COPTFLAG64)
122 COPTFLAG_32 = $(COPTFLAG)
123 COPTFLAG_64 = $(COPTFLAG64)
124 COPTIMIZE = $(COPTFLAG_$(CLASS))
125 #
126 CFLAGS = $(CFLAGS_XARCH)
127 CFLAGS += $(COPTIMIZE)

```

```

126 CFLAGS          += $(INLINES) -D_ASM_INLINES
127 CFLAGS          += $(CCMODE)
128 CFLAGS          += $(SPACEFLAG)
129 CFLAGS          += $(CCUNBOUND)
130 CFLAGS          += $(CFLAGS_uts)
131 CFLAGS          += -xstrconst

133 ASFLAGS_XARCH_32 = $(i386_ASFLAGS)
134 ASFLAGS_XARCH_64 = $(amd64_ASFLAGS)
135 ASFLAGS_XARCH    = $(ASFLAGS_XARCH_$(CLASS))

137 ASFLAGS          += $(ASFLAGS_XARCH)

139 #
140 #       Define the base directory for installation.
141 #
142 BASE_INS_DIR     = $(ROOT)

144 #
145 #       Debugging level
146 #
147 #       Special knowledge of which special debugging options affect which
148 #       file is used to optimize the build if these flags are changed.
149 #
150 DEBUG_DEFS_OBJ32 =
151 DEBUG_DEFS_DBG32 = -DDEBUG
152 DEBUG_DEFS_OBJ64 =
153 DEBUG_DEFS_DBG64 = -DDEBUG
154 DEBUG_DEFS       = $(DEBUG_DEFS_$(BUILD_TYPE))

156 DEBUG_COND_OBJ32 = $(POUND_SIGN)
157 DEBUG_COND_DBG32 =
158 DEBUG_COND_OBJ64 = $(POUND_SIGN)
159 DEBUG_COND_DBG64 =
160 IF_DEBUG_OBJ     = $(DEBUG_COND_$(BUILD_TYPE))$(OBJS_DIR)/

162 $(IF_DEBUG_OBJ)syscall.o      :=      DEBUG_DEFS      += -DSYSCALLTRACE
163 $(IF_DEBUG_OBJ)clock.o       :=      DEBUG_DEFS      += -DKSLICE=1

165 #
166 #       Collect the preprocessor definitions to be associated with *all*
167 #       files.
168 #
169 ALL_DEFS         = $(DEBUG_DEFS) $(OPTION_DEFS)

171 #
172 #       The kernels modules which are "implementation architecture"
173 #       specific for this machine are enumerated below. Note that most
174 #       of these modules must exist (in one form or another) for each
175 #       architecture.
176 #
177 #       Common Drivers (usually pseudo drivers) (/kernel/drv)
178 #       DRV_KMODS are built both 32-bit and 64-bit
179 #       DRV_KMODS_32 are built only 32-bit
180 #       DRV_KMODS_64 are built only 64-bit
181 #
182 DRV_KMODS        += aac
183 DRV_KMODS        += aggr
184 DRV_KMODS        += ahci
185 DRV_KMODS        += amd64_gart
186 DRV_KMODS        += amr
187 DRV_KMODS        += agpgart
188 DRV_KMODS        += srn
189 DRV_KMODS        += agptarget
190 DRV_KMODS        += arn
191 DRV_KMODS        += arp

```

```

192 DRV_KMODS        += asy
193 DRV_KMODS        += ata
194 DRV_KMODS        += ath
195 DRV_KMODS        += atu
196 DRV_KMODS        += audio
197 DRV_KMODS        += audio1575
198 DRV_KMODS        += audio810
199 DRV_KMODS        += audiocmi
200 DRV_KMODS        += audiocmihd
201 DRV_KMODS        += audioemu10k
202 DRV_KMODS        += audioens
203 DRV_KMODS        += audiohd
204 DRV_KMODS        += audioixp
205 DRV_KMODS        += audiols
206 DRV_KMODS        += audiopl6x
207 DRV_KMODS        += audiopci
208 DRV_KMODS        += audiosolo
209 DRV_KMODS        += audiotst
210 DRV_KMODS        += audiovia823x
211 DRV_KMODS_32    += audiovia97
212 DRV_KMODS        += bl
213 DRV_KMODS        += blkdev
214 DRV_KMODS        += bge
215 DRV_KMODS        += bofi
216 DRV_KMODS        += bpf
217 DRV_KMODS        += bridge
218 DRV_KMODS        += bsdbus
219 DRV_KMODS        += bscv
220 DRV_KMODS        += chxge
221 DRV_KMODS        += cxgbe
222 DRV_KMODS        += ntxn
223 DRV_KMODS        += myri10ge
224 DRV_KMODS        += clone
225 DRV_KMODS        += cmdk
226 DRV_KMODS        += cn
227 DRV_KMODS        += conskbd
228 DRV_KMODS        += consms
229 DRV_KMODS        += cpgary3
230 DRV_KMODS        += cpuid
231 DRV_KMODS        += cpunex
232 DRV_KMODS        += crypto
233 DRV_KMODS        += cryptoadm
234 DRV_KMODS        += dca
235 DRV_KMODS        += devinfo
236 DRV_KMODS        += dld
237 DRV_KMODS        += dlpistub
238 DRV_KMODS_32    += dnet
239 DRV_KMODS        += dump
240 DRV_KMODS        += ecpp
241 DRV_KMODS        += emlxs
242 DRV_KMODS        += fd
243 DRV_KMODS        += fdc
244 DRV_KMODS        += fm
245 DRV_KMODS        += fssnap
246 DRV_KMODS        += hxge
247 DRV_KMODS        += i8042
248 DRV_KMODS        += i915
249 DRV_KMODS        += icmp
250 DRV_KMODS        += icmp6
251 DRV_KMODS        += intel_nb5000
252 DRV_KMODS        += intel_nhm
253 DRV_KMODS        += ip
254 DRV_KMODS        += ip6
255 DRV_KMODS        += ipf
256 DRV_KMODS        += ipf
257 DRV_KMODS        += ipnet

```

```

258 DRV_KMODS += ippctl
259 DRV_KMODS += ipsecah
260 DRV_KMODS += ipsecesp
261 DRV_KMODS += ipw
262 DRV_KMODS += iwh
263 DRV_KMODS += iwi
264 DRV_KMODS += iwk
265 DRV_KMODS += iwp
266 DRV_KMODS += iwscn
267 DRV_KMODS += kb8042
268 DRV_KMODS += keysock
269 DRV_KMODS += kssl
270 DRV_KMODS += kstat
271 DRV_KMODS += ksyms
272 DRV_KMODS += kmdb
273 DRV_KMODS += llcl
274 DRV_KMODS += lofi
275 DRV_KMODS += log
276 DRV_KMODS += logindmux
277 DRV_KMODS += mega_sas
278 DRV_KMODS += mc-amd
279 DRV_KMODS += mm
280 DRV_KMODS += mouse8042
281 DRV_KMODS += mpt_sas
282 DRV_KMODS += mr_sas
283 DRV_KMODS += mwl
284 DRV_KMODS += nca
285 DRV_KMODS += nsmb
286 DRV_KMODS += nulldriver
287 DRV_KMODS += nv_sata
288 DRV_KMODS += nxge
289 DRV_KMODS += oce
290 DRV_KMODS += openepr
291 DRV_KMODS += pci_pci
292 DRV_KMODS += pcic
293 DRV_KMODS += pcieb
294 DRV_KMODS += phymem
295 DRV_KMODS += pit_bEEP
296 DRV_KMODS += pm
297 DRV_KMODS += poll
298 DRV_KMODS += pool
299 DRV_KMODS += power
300 DRV_KMODS += pseudo
301 DRV_KMODS += ptc
302 DRV_KMODS += ptm
303 DRV_KMODS += pts
304 DRV_KMODS += ptsl
305 DRV_KMODS += qlge
306 DRV_KMODS += radeon
307 DRV_KMODS += ral
308 DRV_KMODS += ramdisk
309 DRV_KMODS += random
310 DRV_KMODS += rds
311 DRV_KMODS += rdsv3
312 DRV_KMODS += rpcib
313 DRV_KMODS += rsm
314 DRV_KMODS += rts
315 DRV_KMODS += rtw
316 DRV_KMODS += rum
317 DRV_KMODS += rwd
318 DRV_KMODS += rwn
319 DRV_KMODS += sad
320 DRV_KMODS += sd
321 DRV_KMODS += sdhost
322 DRV_KMODS += sgen
323 DRV_KMODS += si3124

```

```

324 DRV_KMODS += smbios
325 DRV_KMODS += softmac
326 DRV_KMODS += spdssock
327 DRV_KMODS += smbsrv
328 DRV_KMODS += smp
329 DRV_KMODS += spps
330 DRV_KMODS += sppptun
331 DRV_KMODS += srpt
332 DRV_KMODS += st
333 DRV_KMODS += sy
334 DRV_KMODS += sysevent
335 DRV_KMODS += sysmsg
336 DRV_KMODS += tcp
337 DRV_KMODS += tcp6
338 DRV_KMODS += tl
339 DRV_KMODS += tnf
340 DRV_KMODS += tpm
341 DRV_KMODS += trill
342 DRV_KMODS += udp
343 DRV_KMODS += udp6
344 DRV_KMODS += ucode
345 DRV_KMODS += ural
346 DRV_KMODS += uath
347 DRV_KMODS += urtw
348 DRV_KMODS += vgatext
349 DRV_KMODS += heci
350 DRV_KMODS += vnica
351 DRV_KMODS += vscan
352 DRV_KMODS += wc
353 DRV_KMODS += winlock
354 DRV_KMODS += wpi
355 DRV_KMODS += xge
356 DRV_KMODS += yge
357 DRV_KMODS += zcons
358 DRV_KMODS += zyd
359 DRV_KMODS += simnet
360 DRV_KMODS += stmf
361 DRV_KMODS += stmf_sbd
362 DRV_KMODS += fct
363 DRV_KMODS += fcoe
364 DRV_KMODS += fcoet
365 DRV_KMODS += fcoel
366 DRV_KMODS += qlt
367 DRV_KMODS += iscsit
368 DRV_KMODS += pppt
369 DRV_KMODS += ncall nsctl sdbc nskern sv
370 DRV_KMODS += ii rdc rdcsrv rdcstub
371 DRV_KMODS += iptun

373 #
374 # Common code drivers
375 #

377 DRV_KMODS += afe
378 DRV_KMODS += atge
379 DRV_KMODS += bfe
380 DRV_KMODS += dmfe
381 DRV_KMODS += el000g
382 DRV_KMODS += efe
383 DRV_KMODS += elxl
384 DRV_KMODS += hme
385 DRV_KMODS += mxfe
386 DRV_KMODS += nge
387 DRV_KMODS += pcn
388 DRV_KMODS += rge
389 DRV_KMODS += rtls

```

```

390 DRV_KMODS      += sfe
391 DRV_KMODS      += amd811ls
392 DRV_KMODS      += igb
393 DRV_KMODS      += ipmi
394 DRV_KMODS      += iprb
395 DRV_KMODS      += ixgbe
396 DRV_KMODS      += vr

398 #
399 # Virtio drivers
400 #

402 # Virtio core
403 DRV_KMODS      += virtio

405 # Virtio block driver
406 DRV_KMODS      += vioblk

408 #
409 #      DTrace and DTrace Providers
410 #
411 DRV_KMODS      += dtrace
412 DRV_KMODS      += fbt
413 DRV_KMODS      += lockstat
414 DRV_KMODS      += profile
415 DRV_KMODS      += sdt
416 DRV_KMODS      += systrace
417 DRV_KMODS      += fasttrap
418 DRV_KMODS      += dcpc

420 #
421 #      I/O framework test drivers
422 #
423 DRV_KMODS      += pshot
424 DRV_KMODS      += gen_drv
425 DRV_KMODS      += tvhci tphci tclient
426 DRV_KMODS      += emul64

428 #
429 #      Machine Specific Driver Modules (/kernel/drv):
430 #
431 DRV_KMODS      += options
432 DRV_KMODS      += scsi_vhci
433 DRV_KMODS      += pmcs
434 DRV_KMODS      += pmcs8001fw
435 DRV_KMODS      += arcmsr
436 DRV_KMODS      += fcp
437 DRV_KMODS      += fcip
438 DRV_KMODS      += fcsm
439 DRV_KMODS      += fp
440 DRV_KMODS      += qlc
441 DRV_KMODS      += iscsi

443 #
444 #      PCMCIA specific module(s)
445 #
446 DRV_KMODS      += pcs
447 MISC_KMODS     += cardbus

449 #
450 #      SCSI Enclosure Services driver
451 #
452 DRV_KMODS      += ses

454 #
455 #      USB specific modules

```

```

456 #
457 DRV_KMODS      += hid
458 DRV_KMODS      += hwarc hwahc
459 DRV_KMODS      += hubd
460 DRV_KMODS      += uhci
461 DRV_KMODS      += ehci
462 DRV_KMODS      += ohci
463 DRV_KMODS      += usb_mid
464 DRV_KMODS      += usb_ia
465 DRV_KMODS      += scsa2usb
466 DRV_KMODS      += usbprn
467 DRV_KMODS      += ugen
468 DRV_KMODS      += usbser
469 DRV_KMODS      += usbsacm
470 DRV_KMODS      += usbsksp
471 DRV_KMODS      += usbspri
472 DRV_KMODS      += usb_ac
473 DRV_KMODS      += usb_as
474 DRV_KMODS      += usbskel
475 DRV_KMODS      += usbvc
476 DRV_KMODS      += usbftdi
477 DRV_KMODS      += wusb_df
478 DRV_KMODS      += wusb_ca
479 DRV_KMODS      += usbecm

481 #
482 #      1394 modules
483 #
484 MISC_KMODS     += s1394 sbp2
485 DRV_KMODS      += hcil394 scsa1394
486 DRV_KMODS      += av1394
487 DRV_KMODS      += dcam1394

489 #
490 #      InfiniBand pseudo drivers
491 #
492 DRV_KMODS      += ib ibp eibnx eoib rdsib sdp iser daplt hermon tavor sol_ucma
493 DRV_KMODS      += sol_umad

495 #
496 #      LVM modules
497 #
498 DRV_KMODS      += md
499 MISC_KMODS     += md_stripe md_hotspares md_mirror md_raid md_trans md_notify
500 MISC_KMODS     += md_sp

502 #
503 #      Brand modules
504 #
505 BRAND_KMODS    += snl_brand s10_brand

507 #
508 #      Exec Class Modules (/kernel/exec):
509 #
510 EXEC_KMODS     += elfexec intpexec shbinexec javaexec

512 #
513 #      Scheduling Class Modules (/kernel/sched):
514 #
515 SCHED_KMODS    += IA RT TS RT_DPTBL TS_DPTBL FSS FX FX_DPTBL SDC

517 #
518 #      File System Modules (/kernel/fs):
519 #
520 FS_KMODS       += autofs cachefs ctfs dcfs dev devfs fdfs fifofs hsfds lofs
521 FS_KMODS       += mntfs namefs nfs objfs zfs zut

```

```

522 FS_KMODS      += pcfs procfs sockfs specfs tmpfs udfs ufs sharefs
523 FS_KMODS      += smbfs

525 #
526 #           Streams Modules (/kernel/strmod):
527 #
528 STRMOD_KMODS   += bufmod connld dedump ldterm pckt pfmod pipemod
529 STRMOD_KMODS   += ptem redirmod rpcmod rlmmod telmod timod
530 STRMOD_KMODS   += sppsasyn sppscomp
531 STRMOD_KMODS   += tirdwr ttcompat
532 STRMOD_KMODS   += usbkbm
533 STRMOD_KMODS   += usbms
534 STRMOD_KMODS   += usbwcm
535 STRMOD_KMODS   += usb_ah
536 STRMOD_KMODS   += drcompat
537 STRMOD_KMODS   += cryptmod
538 STRMOD_KMODS   += vuid2ps2
539 STRMOD_KMODS   += vuid3ps2
540 STRMOD_KMODS   += vuidm3p
541 STRMOD_KMODS   += vuidm4p
542 STRMOD_KMODS   += vuidm5p

544 #
545 #           'System' Modules (/kernel/sys):
546 #
547 SYS_KMODS      += c2audit
548 SYS_KMODS      += doorfs
549 SYS_KMODS      += exacctsys
550 SYS_KMODS      += inst_sync
551 SYS_KMODS      += kaio
552 SYS_KMODS      += msgsys
553 SYS_KMODS      += pipe
554 SYS_KMODS      += portfs
555 SYS_KMODS      += pset
556 SYS_KMODS      += semsys
557 SYS_KMODS      += shmsys
558 SYS_KMODS      += sysacct
559 SYS_KMODS      += acctctl

561 #
562 #           'Misc' Modules (/kernel/misc)
563 #           MISC_KMODS are built both 32-bit and 64-bit
564 #           MISC_KMODS_32 are built only 32-bit
565 #           MISC_KMODS_64 are built only 64-bit
566 #
567 MISC_KMODS     += ac97
568 MISC_KMODS     += acpica
569 MISC_KMODS     += agpmaster
570 MISC_KMODS     += bignum
571 MISC_KMODS     += bootdev
572 MISC_KMODS     += busra
573 MISC_KMODS     += cmlb
574 MISC_KMODS     += consconfig
575 MISC_KMODS     += ctf
576 MISC_KMODS     += dadk
577 MISC_KMODS     += dcopy
578 MISC_KMODS     += dls
579 MISC_KMODS     += drm
580 MISC_KMODS     += fssnap_if
581 MISC_KMODS     += gda
582 MISC_KMODS     += gld
583 MISC_KMODS     += hidparser
584 MISC_KMODS     += hook
585 MISC_KMODS     += hpcsvc
586 MISC_KMODS     += ibcm
587 MISC_KMODS     += ibdm

```

```

588 MISC_KMODS     += ibdma
589 MISC_KMODS     += ibmf
590 MISC_KMODS     += ibt1
591 MISC_KMODS     += idm
592 MISC_KMODS     += idmap
593 MISC_KMODS     += iomulib
594 MISC_KMODS     += ipc
595 MISC_KMODS     += kbtrans
596 MISC_KMODS     += kcf
597 MISC_KMODS     += kgssapi
598 MISC_KMODS     += kmecch_dummy
599 MISC_KMODS     += kmecch_krb5
600 MISC_KMODS     += ksocket
601 MISC_KMODS     += mac
602 MISC_KMODS     += mii
603 MISC_KMODS     += mwlfw
604 MISC_KMODS     += net80211
605 MISC_KMODS     += nfs_dlboot
606 MISC_KMODS     += nfssrv
607 MISC_KMODS     += neti
608 MISC_KMODS     += pci_autoconfig
609 MISC_KMODS     += pcicfg
610 MISC_KMODS     += pcihp
611 MISC_KMODS     += pcmcia
612 MISC_KMODS     += rpcsec
613 MISC_KMODS     += rpcsec_gss
614 MISC_KMODS     += rsmops
615 MISC_KMODS     += sata
616 MISC_KMODS     += scsi
617 MISC_KMODS     += sda
618 MISC_KMODS     += sol_ofs
619 MISC_KMODS     += spuni
620 MISC_KMODS     += strategy
621 MISC_KMODS     += strplumb
622 MISC_KMODS     += tem
623 MISC_KMODS     += tlimod
624 MISC_KMODS     += usba usba10 usbs49_fw
625 MISC_KMODS     += scsi_vhci_f_sym_hds
626 MISC_KMODS     += scsi_vhci_f_sym
627 MISC_KMODS     += scsi_vhci_f_tpqs
628 MISC_KMODS     += scsi_vhci_f_asym_sun
629 MISC_KMODS     += scsi_vhci_f_tape
630 MISC_KMODS     += scsi_vhci_f_tpqs_tape
631 MISC_KMODS     += fctl
632 MISC_KMODS     += emlxs_fw
633 MISC_KMODS     += qlc_fw_2200
634 MISC_KMODS     += qlc_fw_2300
635 MISC_KMODS     += qlc_fw_2400
636 MISC_KMODS     += qlc_fw_2500
637 MISC_KMODS     += qlc_fw_6322
638 MISC_KMODS     += qlc_fw_8100
639 MISC_KMODS     += hwa1480_fw
640 MISC_KMODS     += uathfw
641 MISC_KMODS     += uwba

643 MISC_KMODS     += klmmod klmops

645 #
646 #           Software Cryptographic Providers (/kernel/crypto):
647 #
648 CRYPTO_KMODS   += aes
649 CRYPTO_KMODS   += arcfour
650 CRYPTO_KMODS   += blowfish
651 CRYPTO_KMODS   += des
652 CRYPTO_KMODS   += ecc
653 CRYPTO_KMODS   += md4

```



```

654 CRYPTO_KMODS      += md5
655 CRYPTO_KMODS      += rsa
656 CRYPTO_KMODS      += sha1
657 CRYPTO_KMODS      += sha2
658 CRYPTO_KMODS      += swrand

660 #
661 #       IP Policy Modules (/kernel/ipp)
662 #
663 IPP_KMODS           += dlcosmk
664 IPP_KMODS           += flowacct
665 IPP_KMODS           += ipgpc
666 IPP_KMODS           += dscpmk
667 IPP_KMODS           += tokenmt
668 IPP_KMODS           += tswtclmt

670 #
671 #       generic-unix module (/kernel/genunix):
672 #
673 GENUNIX_KMODS      += genunix

675 #
676 #       Modules eXcluded from the product:
677 #

679 #
680 #       'Dacf' Modules (/kernel/dacf):
681 #

683 #
684 #       Performance Counter BackEnd modules (/usr/kernel/pcbe)
685 #
686 PCBE_KMODS         += p123_pcbe p4_pcbe opteron_pcbe core_pcbe

688 #
689 #       MAC-Type Plugin Modules (/kernel/mac)
690 #
691 MAC_KMODS           += mac_6to4
692 MAC_KMODS           += mac_ether
693 MAC_KMODS           += mac_ipv4
694 MAC_KMODS           += mac_ipv6
695 MAC_KMODS           += mac_wifi
696 MAC_KMODS           += mac_ib

698 #
699 #       socketmod (kernel/socketmod)
700 #
701 SOCKET_KMODS        += sockpfp
702 SOCKET_KMODS        += socksctp
703 SOCKET_KMODS        += socksdp
704 SOCKET_KMODS        += sockrds
705 SOCKET_KMODS        += ksslif

707 #
708 #       kiconv modules (/kernel/kiconv):
709 #
710 KICONV_KMODS        += kiconv_emea kiconv_ja kiconv_ko kiconv_sc kiconv_tc

712 #
713 #       'Dacf' Modules (/kernel/dacf):
714 #
715 DACF_KMODS          += net_dacf

717 #
718 #       Ensure that the variable member of the cpu_t (cpu_m) is defined
719 #       for the lint builds so as not to cause lint errors during the

```

```

720 # global cross check.
721 #
722 LINTFLAGS            += -D_MACHDEP -I$(UTSBASE)/i86pc

```