```
**********************************************************
   37055 Sun Jul 28 20:29:06 2013
new/usr/src/Makefile.master
Arrange things Gordon's way, and be sure to pass -xs
3735 should include an empty make variable in the default CFLAGS/CCFLAGS
3844 the build should make source-level debugging easier
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Garrett D'Amore <garrett@damore.org>
**********************************************************
```

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#

#
# Copyright (c) 1989, 2010, Oracle and/or its affiliates. All rights reserved.
# Copyright (c) 2012 by Delphix. All rights reserved.
#

#
# Makefile.master, global definitions for system source
#
ROOT=           /proto

#
# RELEASE_BUILD should be cleared for final release builds.
# NOT_RELEASE_BUILD is exactly what the name implies.
#
# INTERNAL_RELEASE_BUILD is a subset of RELEASE_BUILD. It mostly controls
# identification strings. Enabling RELEASE_BUILD automatically enables
# INTERNAL_RELEASE_BUILD.
#
# EXPORT_RELEASE_BUILD controls whether binaries are built in a form that
# can be released for export under a binary license.  It is orthogonal to
# the other *RELEASE_BUILD settings.  ("#" means do an export release
# build, "" means do a normal build.)
#
# CLOSED_BUILD controls whether we try to build files under
# usr/closed.  ("" means to build closed code, "#" means don't try to
# build it.)  Skipping the closed code implies doing an export release
# build.
#
# STRIP_COMMENTS toggles comment section striping. Generally the same setting
# as INTERNAL_RELEASE_BUILD.
#
# __GNUC toggles the building of ON components using gcc and related tools.
# Normally set to '#', set it to '' to do gcc build.
#
# The declaration POUND_SIGN is always '#'. This is needed to get around the
# make feature that '#' is always a comment delimiter, even when escaped or
```



```
# quoted. We use this macro expansion method to get POUND_SIGN rather than
# always breaking out a shell because the general case can cause a noticable
# slowdown in build times when so many Makefiles include Makefile.master.
#
# While the majority of users are expected to override the setting below
# with an env file (via nightly or bldenv), if you aren't building that way
# (ie, you're using "ws" or some other bootstrapping method) then you need
# this definition in order to avoid the subshell invocation mentioned above.
#

PRE_POUND=                              pre\#
POUND_SIGN=                             $(PRE_POUND:pre\%=%)

NOT_RELEASE_BUILD=
INTERNAL_RELEASE_BUILD=                 $(POUND_SIGN)
RELEASE_BUILD=                          $(POUND_SIGN)
$(RELEASE_BUILD)NOT_RELEASE_BUILD=      $(POUND_SIGN)
$(RELEASE_BUILD)INTERNAL_RELEASE_BUILD=
PATCH_BUILD=                            $(POUND_SIGN)

# If CLOSED_IS_PRESENT is not set, assume the closed tree is present.
CLOSED_BUILD_1= $(CLOSED_IS_PRESENT:yes=)
CLOSED_BUILD=   $(CLOSED_BUILD_1:no=$(POUND_SIGN))

EXPORT_RELEASE_BUILD=                   $(POUND_SIGN)
$(CLOSED_BUILD)EXPORT_RELEASE_BUILD=

# SPARC_BLD is '#' for an Intel build.
# INTEL_BLD is '#' for a Sparc build.
SPARC_BLD_1=    $(MACH:i386=$(POUND_SIGN))
SPARC_BLD=      $(SPARC_BLD_1:sparc=)
INTEL_BLD_1=    $(MACH:sparc=$(POUND_SIGN))
INTEL_BLD=      $(INTEL_BLD_1:i386=)

STRIP_COMMENTS= $(INTERNAL_RELEASE_BUILD)

# Are we building tonic closedbins? Unless you have used the
# -O flag to nightly or bldenv, leave the definition of TONICBUILD
# as $(POUND_SIGN).
#
# IF YOU CHANGE CLOSEDROOT, you MUST change install.bin
# to match the new definition.
TONICBUILD=     $(POUND_SIGN)
$(TONICBUILD)CLOSEDROOT= $(ROOT)-closed


# The variables below control the compilers used during the build.
# There are a number of permutations.
#
# __GNUC and __SUNC control (and indicate) the primary compiler.  Whichever
# one is not POUND_SIGN is the primary, with the other as the shadow.  They
# may also be used to control entirely compiler-specific Makefile assignments.
# __SUNC and Sun Studio are the default.
#
# __GNUC64 indicates that the 64bit build should use the GNU C compiler.
# There is no Sun C analogue.
#
# The following version-specific options are operative regardless of which
# compiler is primary, and control the versions of the given compilers to be
# used.  They also allow compiler-version specific Makefile fragments.
#

__GNUC=                 $(POUND_SIGN)
$(__GNUC)__SUNC=        $(POUND_SIGN)
__GNUC64=               $(__GNUC)
```

```
124 # CLOSED is the root of the tree that contains source which isn't released
125 # as open source
126 CLOSED=         $(SRC)/../closed

128 # BUILD_TOOLS is the root of all tools including compilers.
129 # ONBLD_TOOLS is the root of all the tools that are part of SUNWonbld.

131 BUILD_TOOLS=            /ws/onnv-tools
132 ONBLD_TOOLS=            $(BUILD_TOOLS)/onbld

134 JAVA_ROOT=      /usr/java

136 SFW_ROOT=       /usr/sfw
137 SFWINCDIR=      $(SFW_ROOT)/include
138 SFWLIBDIR=      $(SFW_ROOT)/lib
139 SFWLIBDIR64=    $(SFW_ROOT)/lib/$(MACH64)

141 GCC_ROOT=       /opt/gcc/4.4.4
142 GCCLIBDIR=      $(GCC_ROOT)/lib
143 GCCLIBDIR64=    $(GCC_ROOT)/lib/$(MACH64)

145 DOCBOOK_XSL_ROOT=       /usr/share/sgml/docbook/xsl-stylesheets

147 RPCGEN=         /usr/bin/rpcgen
148 STABS=          $(ONBLD_TOOLS)/bin/$(MACH)/stabs
149 ELFEXTRACT=     $(ONBLD_TOOLS)/bin/$(MACH)/elfextract
150 MBH_PATCH=      $(ONBLD_TOOLS)/bin/$(MACH)/mbh_patch
151 ECHO=           echo
152 INS=            install
153 TRUE=           true
154 SYMLINK=        /usr/bin/ln -s
155 LN=             /usr/bin/ln
156 CHMOD=          /usr/bin/chmod
157 MV=             /usr/bin/mv -f
158 RM=             /usr/bin/rm -f
159 CUT=            /usr/bin/cut
160 NM=             /usr/ccs/bin/nm
161 DIFF=           /usr/bin/diff
162 GREP=           /usr/bin/grep
163 EGREP=          /usr/bin/egrep
164 ELFWRAP=        /usr/bin/elfwrap
165 KSH93=          /usr/bin/ksh93
166 SED=            /usr/bin/sed
167 NAWK=           /usr/bin/nawk
168 CP=             /usr/bin/cp -f
169 MCS=            /usr/ccs/bin/mcs
170 CAT=            /usr/bin/cat
171 ELFDUMP=        /usr/ccs/bin/elfdump
172 M4=             /usr/ccs/bin/m4
173 STRIP=          /usr/ccs/bin/strip
174 LEX=            /usr/ccs/bin/lex
175 FLEX=           $(SFW_ROOT)/bin/flex
176 YACC=           /usr/ccs/bin/yacc
177 CPP=            /usr/lib/cpp
178 JAVAC=          $(JAVA_ROOT)/bin/javac
179 JAVAH=          $(JAVA_ROOT)/bin/javah
180 JAVADOC=        $(JAVA_ROOT)/bin/javadoc
181 RMIC=           $(JAVA_ROOT)/bin/rmic
182 JAR=            $(JAVA_ROOT)/bin/jar
183 CTFCONVERT=     $(ONBLD_TOOLS)/bin/$(MACH)/ctfconvert
184 CTFMERGE=       $(ONBLD_TOOLS)/bin/$(MACH)/ctfmerge
185 CTFSTABS=       $(ONBLD_TOOLS)/bin/$(MACH)/ctfstabs
186 CTFSTRIP=       $(ONBLD_TOOLS)/bin/$(MACH)/ctfstrip
187 NDRGEN=         $(ONBLD_TOOLS)/bin/$(MACH)/ndrgen
188 GENOFFSETS=     $(ONBLD_TOOLS)/bin/genoffsets
189 CTFCVTPTBL=     $(ONBLD_TOOLS)/bin/ctfcvtptbl
```



```
190 CTFFINDMOD=     $(ONBLD_TOOLS)/bin/ctffindmod
191 XREF=           $(ONBLD_TOOLS)/bin/xref
192 FIND=           /usr/bin/find
193 PERL=           /usr/bin/perl
194 PYTHON_26=      /usr/bin/python2.6
195 PYTHON=         $(PYTHON_26)
196 SORT=           /usr/bin/sort
197 TOUCH=          /usr/bin/touch
198 WC=             /usr/bin/wc
199 XARGS=          /usr/bin/xargs
200 ELFEDIT=        /usr/bin/elfedit
201 ELFSIGN=        /usr/bin/elfsign
202 DTRACE=         /usr/sbin/dtrace -xnolibs
203 UNIQ=           /usr/bin/uniq
204 TAR=            /usr/bin/tar

206 FILEMODE=       644
207 DIRMODE=        755

209 #
210 # The version of the patch makeup table optimized for build-time use.  Used
211 # during patch builds only.
212 $(PATCH_BUILD)PMTMO_FILE=$(SRC)/patch_makeup_table.mo

214 # Declare that nothing should be built in parallel.
215 # Individual Makefiles can use the .PARALLEL target to declare otherwise.
216 .NO_PARALLEL:

218 # For stylistic checks
219 #
220 # Note that the X and C checks are not used at this time and may need
221 # modification when they are actually used.
222 #
223 CSTYLE=         $(ONBLD_TOOLS)/bin/cstyle
224 CSTYLE_TAIL=
225 HDRCHK=         $(ONBLD_TOOLS)/bin/hdrchk
226 HDRCHK_TAIL=
227 JSTYLE=         $(ONBLD_TOOLS)/bin/jstyle

229 DOT_H_CHECK=    \
230         @$(ECHO) "checking $<"; $(CSTYLE) $< $(CSTYLE_TAIL); \
231         $(HDRCHK) $< $(HDRCHK_TAIL)

233 DOT_X_CHECK=    \
234         @$(ECHO) "checking $<"; $(RPCGEN) -C -h $< | $(CSTYLE) $(CSTYLE_TAIL); \
235         $(RPCGEN) -C -h $< | $(HDRCHK) $< $(HDRCHK_TAIL)

237 DOT_C_CHECK=    \
238         @$(ECHO) "checking $<"; $(CSTYLE) $< $(CSTYLE_TAIL)

240 MANIFEST_CHECK= \
241         @$(ECHO) "checking $<"; \
242         SVCCFG_DTD=$(SRC)/cmd/svc/dtd/service_bundle.dtd.1 \
243         SVCCFG_REPOSITORY=$(SRC)/cmd/svc/seed/global.db \
244         SVCCFG_CONFIGD_PATH=$(SRC)/cmd/svc/configd/svc.configd-native \
245         $(SRC)/cmd/svc/svccfg/svccfg-native validate $<

247 #
248 # IMPORTANT:: If you change any of INS.file, INS.dir, INS.rename,
249 # INS.link or INS.symlink here, then you must also change the
250 # corresponding override definitions in $CLOSED/Makefile.tonic.
251 # If you do not do this, then the closedbins build for the OpenSolaris
252 # community will break. PS, the gatekeepers will be upset too.
253 INS.file=       $(RM) $@; $(INS) -s -m $(FILEMODE) -f $(@D) $<
254 INS.dir=        $(INS) -s -d -m $(DIRMODE) $@
255 # installs and renames at once
```

```
256 #
257 INS.rename=     $(INS.file); $(MV) $(@D)/$(<F) $@

259 # install a link
260 INSLINKTARGET=  $<
261 INS.link=       $(RM) $@; $(LN) $(INSLINKTARGET) $@
262 INS.symlink=    $(RM) $@; $(SYMLINK) $(INSLINKTARGET) $@

264 #
265 # Python bakes the mtime of the .py file into the compiled .pyc and
266 # rebuilds if the baked-in mtime != the mtime of the source file
267 # (rather than only if it's less than), thus when installing python
268 # files we must make certain to not adjust the mtime of the source
269 # (.py) file.
270 #
271 INS.pyfile=     $(INS.file); $(TOUCH) -r $< $@

273 # MACH must be set in the shell environment per uname -p on the build host
274 # More specific architecture variables should be set in lower makefiles.
275 #
276 # MACH64 is derived from MACH, and BUILD64 is set to '#' for
277 # architectures on which we do not build 64-bit versions.
278 # (There are no such architectures at the moment.)
279 #
280 # Set BUILD64=# in the environment to disable 64-bit amd64
281 # builds on i386 machines.

283 MACH64_1=       $(MACH:sparc=sparcv9)
284 MACH64=         $(MACH64_1:i386=amd64)

286 MACH32_1=       $(MACH:sparc=sparcv7)
287 MACH32=         $(MACH32_1:i386=i86)

289 sparc_BUILD64=
290 i386_BUILD64=
291 BUILD64=        $($(MACH)_BUILD64)

293 #
294 # C compiler mode. Future compilers may change the default on us,
295 # so force extended ANSI mode globally. Lower level makefiles can
296 # override this by setting CCMODE.
297 #
298 CCMODE=                 -Xa
299 CCMODE64=               -Xa

301 #
302 # C compiler verbose mode. This is so we can enable it globally,
303 # but turn it off in the lower level makefiles of things we cannot
304 # (or aren't going to) fix.
305 #
306 CCVERBOSE=              -v

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

312 # set this to the secret flag "-Wc,-Qiselect-regsym=0" to disable register
313 # symbols (used to detect conflicts between objects that use global registers)
314 # we disable this now for safety, and because genunix doesn't link with
315 # this feature (the v9 default) enabled.
316 #
317 # REGSYM is separate since the C++ driver syntax is different.
318 CCREGSYM=               -Wc,-Qiselect-regsym=0
319 CCCREGSYM=              -Qoption cg -Qiselect-regsym=0

321 # Prevent the removal of static symbols by the SPARC code generator (cg).
```



```
322 # The x86 code generator (ube) does not remove such symbols and as such
323 # using this workaround is not applicable for x86.
324 #
325 CCSTATICSYM=            -Wc,-Qassembler-ounrefsym=0
326 #
327 # generate 32-bit addresses in the v9 kernel. Saves memory.
328 CCABS32=                -Wc,-xcode=abs32
329 #
330 # generate v9 code which tolerates callers using the v7 ABI, for the sake of
331 # system calls.
332 CC32BITCALLERS=         -_gcc=-massume-32bit-callers

334 # GCC, especially, is increasingly beginning to auto-inline functions and
335 # sadly does so separately not under the general -fno-inline-functions
336 # Additionally, we wish to prevent optimisations which cause GCC to clone
337 # functions -- in particular, these may cause unhelpful symbols to be
338 # emitted instead of function names
339 CCNOAUTOINLINE= -_gcc=-fno-inline-small-functions \
340         -_gcc=-fno-inline-functions-called-once \
341         -_gcc=-fno-ipa-cp

343 # One optimization the compiler might perform is to turn this:
344 #       #pragma weak foo
345 #       extern int foo;
346 #       if (&foo)
347 #               foo = 5;
348 # into
349 #       foo = 5;
350 # Since we do some of this (foo might be referenced in common kernel code
351 # but provided only for some cpu modules or platforms), we disable this
352 # optimization.
353 #
354 sparc_CCUNBOUND = -Wd,-xsafe=unboundsym
355 i386_CCUNBOUND  =
356 CCUNBOUND       = $($(MACH)_CCUNBOUND)

358 #
359 # compiler '-xarch' flag. This is here to centralize it and make it
360 # overridable for testing.
361 sparc_XARCH=    -m32
362 sparcv9_XARCH=  -m64
363 i386_XARCH=
364 amd64_XARCH=    -m64 -Ui386 -U__i386

366 # assembler '-xarch' flag.  Different from compiler '-xarch' flag.
367 sparc_AS_XARCH=         -xarch=v8plus
368 sparcv9_AS_XARCH=       -xarch=v9
369 i386_AS_XARCH=
370 amd64_AS_XARCH=         -xarch=amd64 -P -Ui386 -U__i386

372 #
373 # These flags define what we need to be 'standalone' i.e. -not- part
374 # of the rather more cosy userland environment.  This basically means
375 # the kernel.
376 #
377 # XX64  future versions of gcc will make -mcmodel=kernel imply -mno-red-zone
378 #
379 sparc_STAND_FLAGS=      -_gcc=-ffreestanding
380 sparcv9_STAND_FLAGS=    -_gcc=-ffreestanding
381 # Disabling MMX also disables 3DNow, disabling SSE also disables all later
382 # additions to SSE (SSE2, AVX ,etc.)
383 NO_SIMD=                -_gcc=-mno-mmx -_gcc=-mno-sse
384 i386_STAND_FLAGS=       -_gcc=-ffreestanding $(NO_SIMD)
385 amd64_STAND_FLAGS=      -xmodel=kernel $(NO_SIMD)

387 SAVEARGS=               -Wu,-save_args
```

```
388 amd64_STAND_FLAGS       += $(SAVEARGS)

390 STAND_FLAGS_32 = $($(MACH)_STAND_FLAGS)
391 STAND_FLAGS_64 = $($(MACH64)_STAND_FLAGS)

393 #
394 # disable the incremental linker
395 ILDOFF=                 -xildoff
396 #
397 XDEPEND=                -xdepend
398 XFFLAG=                 -xF=%all
399 XESS=                   -xs
400 XSTRCONST=              -xstrconst

402 #
403 # turn warnings into errors (C)
404 CERRWARN = -errtags=yes -errwarn=%all
405 CERRWARN += -erroff=E_EMPTY_TRANSLATION_UNIT
406 CERRWARN += -erroff=E_STATEMENT_NOT_REACHED

408 CERRWARN += -_gcc=-Wno-missing-braces
409 CERRWARN += -_gcc=-Wno-sign-compare
410 CERRWARN += -_gcc=-Wno-unknown-pragmas
411 CERRWARN += -_gcc=-Wno-unused-parameter
412 CERRWARN += -_gcc=-Wno-missing-field-initializers

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

417 # DEBUG v. -nd make for frequent unused variables, empty conditions, etc. in
418 # -nd builds
419 $(RELEASE_BUILD)CERRWARN += -_gcc=-Wno-unused
420 $(RELEASE_BUILD)CERRWARN += -_gcc=-Wno-empty-body

422 #
423 # turn warnings into errors (C++)
424 CCERRWARN=              -xwe

426 # C99 mode
427 C99_ENABLE=     -xc99=%all
428 C99_DISABLE=    -xc99=%none
429 C99MODE=        $(C99_DISABLE)
430 C99LMODE=       $(C99MODE:-xc99%=-Xc99%)

432 # In most places, assignments to these macros should be appended with +=
433 # (CPPFLAGS.master allows values to be prepended to CPPFLAGS).
434 sparc_CFLAGS=   $(sparc_XARCH) $(CCSTATICSYM)
435 sparcv9_CFLAGS= $(sparcv9_XARCH) -dalign $(CCVERBOSE) $(V9ABIWARN) $(CCREGSYM) \
436                 $(CCSTATICSYM)
437 i386_CFLAGS=    $(i386_XARCH)
438 amd64_CFLAGS=   $(amd64_XARCH)

440 sparc_ASFLAGS=  $(sparc_AS_XARCH)
441 sparcv9_ASFLAGS=$(sparcv9_AS_XARCH)
442 i386_ASFLAGS=   $(i386_AS_XARCH)
443 amd64_ASFLAGS=  $(amd64_AS_XARCH)

445 #
446 sparc_COPTFLAG=         -xO3
447 sparcv9_COPTFLAG=       -xO3
448 i386_COPTFLAG=          -O
449 amd64_COPTFLAG=         -xO3

451 COPTFLAG= $($(MACH)_COPTFLAG)
452 COPTFLAG64= $($(MACH64)_COPTFLAG)
```



```
454 # When -g is used, the compiler globalizes static objects
455 # (gives them a unique prefix). Disable that.
456 CNOGLOBAL= -W0,-noglobal

458 # Direct the Sun Studio compiler to use a static globalization prefix based on t
459 # name of the module rather than something unique. Otherwise, objects
460 # will not build deterministically, as subsequent compilations of identical
461 # source will yeild objects that always look different.
462 #
463 # In the same spirit, this will also remove the date from the N_OPT stab.
464 CGLOBALSTATIC= -W0,-xglobalstatic

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

470 #
471 # Default debug format for Sun Studio 11 is dwarf, so force it to
472 # generate stabs.
473 #
474 DEBUGFORMAT=    -xdebugformat=stabs

476 #
477 # Flags used to build in debug mode for ctf generation.  Bugs in the Devpro
478 # compilers currently prevent us from building with cc-emitted DWARF.
479 #
480 CTF_FLAGS_sparc = -g -Wc,-Qiselect-T1 $(C99MODE) $(CNOGLOBAL) $(CDWARFSTR)
481 CTF_FLAGS_i386  = -g $(C99MODE) $(CNOGLOBAL) $(CDWARFSTR)

483 CTF_FLAGS_sparcv9       = $(CTF_FLAGS_sparc)
484 CTF_FLAGS_amd64         = $(CTF_FLAGS_i386)

486 # Sun Studio produces broken userland code when saving arguments.
487 $(__GNUC)CTF_FLAGS_amd64 += $(SAVEARGS)

489 CTF_FLAGS_32    = $(CTF_FLAGS_$(MACH)) $(DEBUGFORMAT)
490 CTF_FLAGS_64    = $(CTF_FLAGS_$(MACH64)) $(DEBUGFORMAT)
491 CTF_FLAGS       = $(CTF_FLAGS_32)

493 #
494 # Flags used with genoffsets
495 #
496 GOFLAGS = -_noecho \
497         $(CALLSYMS) \
498         $(CDWARFSTR)

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

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

506 #
507 # tradeoff time for space (smaller is better)
508 #
509 sparc_SPACEFLAG         = -xspace -W0,-Lt
510 sparcv9_SPACEFLAG       = -xspace -W0,-Lt
511 i386_SPACEFLAG          = -xspace
512 amd64_SPACEFLAG         =

514 SPACEFLAG               = $($(MACH)_SPACEFLAG)
515 SPACEFLAG64             = $($(MACH64)_SPACEFLAG)

517 #
518 # The Sun Studio 11 compiler has changed the behaviour of integer
519 # wrap arounds and so a flag is needed to use the legacy behaviour
```

```
# (without this flag panics/hangs could be exposed within the source).
#
sparc_IROPTFLAG         = -W2,-xwrap_int
sparcv9_IROPTFLAG       = -W2,-xwrap_int
i386_IROPTFLAG          =
amd64_IROPTFLAG         =

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

sparc_XREGSFLAG         = -xregs=no%appl
sparcv9_XREGSFLAG       = -xregs=no%appl
i386_XREGSFLAG          =
amd64_XREGSFLAG         =

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

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

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

CSOURCEDEBUGFLAGS       =
CCSOURCEDEBUGFLAGS      =
$(SRCDBGBLD)CSOURCEDEBUGFLAGS   = -g -xs
$(SRCDBGBLD)CCSOURCEDEBUGFLAGS  = -g -xs

#endif /* ! codereview */
CFLAGS=         $(COPTFLAG) $($(MACH)_CFLAGS) $(SPACEFLAG) $(CCMODE) \
                $(ILDOFF) $(CERRWARN) $(C99MODE) $(CCUNBOUND) $(IROPTFLAG) \
                $(CGLOBALSTATIC) $(CCNOAUTOINLINE) $(CSOURCEDEBUGFLAGS) \
                $(CUSERFLAGS)
                $(CGLOBALSTATIC) $(CCNOAUTOINLINE)
CFLAGS64=       $(COPTFLAG64) $($(MACH64)_CFLAGS) $(SPACEFLAG64) $(CCMODE64) \
                $(ILDOFF) $(CERRWARN) $(C99MODE) $(CCUNBOUND) $(IROPTFLAG64) \
                $(CGLOBALSTATIC) $(CCNOAUTOINLINE) $(CSOURCEDEBUGFLAGS) \
                $(CUSERFLAGS64)
                $(CGLOBALSTATIC) $(CCNOAUTOINLINE)
#
# Flags that are used to build parts of the code that are subsequently
# run on the build machine (also known as the NATIVE_BUILD).
#
NATIVE_CFLAGS=  $(COPTFLAG) $($(NATIVE_MACH)_CFLAGS) $(CCMODE) \
                $(ILDOFF) $(CERRWARN) $(C99MODE) $($(NATIVE_MACH)_CCUNBOUND) \
                $(IROPTFLAG) $(CGLOBALSTATIC) $(CCNOAUTOINLINE) \
                $(CSOURCEDEBUGFLAGS) $(CUSERFLAGS)
                $(IROPTFLAG) $(CGLOBALSTATIC) $(CCNOAUTOINLINE)

DTEXTDOM=-DTEXT_DOMAIN=\"$(TEXT_DOMAIN)\"       # For messaging.
DTS_ERRNO=-D_TS_ERRNO
CPPFLAGS.master=$(DTEXTDOM) $(DTS_ERRNO) \
```



```
        $(ENVCPPFLAGS1) $(ENVCPPFLAGS2) $(ENVCPPFLAGS3) $(ENVCPPFLAGS4)
CPPFLAGS.native=$(ENVCPPFLAGS1) $(ENVCPPFLAGS2) $(ENVCPPFLAGS3) $(ENVCPPFLAGS4)
CPPFLAGS=       $(CPPFLAGS.master)
AS_CPPFLAGS=    $(CPPFLAGS.master)
JAVAFLAGS=      -deprecation

#
# For source message catalogue
#
.SUFFIXES: $(SUFFIXES) .i .po
MSGROOT= $(ROOT)/catalog
MSGDOMAIN= $(MSGROOT)/$(TEXT_DOMAIN)
MSGDOMAINPOFILE = $(MSGDOMAIN)/$(POFILE)
DCMSGDOMAIN= $(MSGROOT)/LC_TIME/$(TEXT_DOMAIN)
DCMSGDOMAINPOFILE = $(DCMSGDOMAIN)/$(DCFILE:.dc=.po)

CLOBBERFILES += $(POFILE) $(POFILES)
COMPILE.cpp= $(CC) -E -C $(CFLAGS) $(CPPFLAGS)
XGETTEXT= /usr/bin/xgettext
XGETFLAGS= -c TRANSLATION_NOTE
GNUXGETTEXT= /usr/gnu/bin/xgettext
GNUXGETFLAGS= --add-comments=TRANSLATION_NOTE --keyword=_ \
        --strict --no-location --omit-header
BUILD.po= $(XGETTEXT) $(XGETFLAGS) -d $(<F) $<.i ;\
        $(RM)   $@ ;\
        $(SED) "/^domain/d" < $(<F).po > $@ ;\
        $(RM) $(<F).po $<.i

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

sparc_CCFLAGS=          -cg92 -compat=4 \
                        -Qoption ccfe -messages=no%anachronism \
                        $(CCERRWARN)
sparcv9_CCFLAGS=        $(sparcv9_XARCH) -dalign -compat=5 \
                        -Qoption ccfe -messages=no%anachronism \
                        -Qoption ccfe -features=no%conststrings \
                        $(CCCREGSYM) \
                        $(CCERRWARN)
i386_CCFLAGS=           -compat=4 \
                        -Qoption ccfe -messages=no%anachronism \
                        -Qoption ccfe -features=no%conststrings \
                        $(CCERRWARN)
amd64_CCFLAGS=          $(amd64_XARCH) -compat=5 \
                        -Qoption ccfe -messages=no%anachronism \
                        -Qoption ccfe -features=no%conststrings \
                        $(CCERRWARN)

sparc_CCOPTFLAG=        -O
sparcv9_CCOPTFLAG=      -O
i386_CCOPTFLAG=         -O
amd64_CCOPTFLAG=        -O

CCOPTFLAG=      $($(MACH)_CCOPTFLAG)
CCOPTFLAG64=    $($(MACH64)_CCOPTFLAG)
CCFLAGS=        $(CCOPTFLAG) $($(MACH)_CCFLAGS) $(CCSOURCEDEBUGFLAGS) \
                $(CCUSERFLAGS)
CCFLAGS64=      $(CCOPTFLAG64) $($(MACH64)_CCFLAGS) $(CCSOURCEDEBUGFLAGS) \
                $(CCUSERFLAGS64)
CCFLAGS=        $(CCOPTFLAG) $($(MACH)_CCFLAGS)
CCFLAGS64=      $(CCOPTFLAG64) $($(MACH64)_CCFLAGS)

#
#
```

```
#
ELFWRAP_FLAGS   =
ELFWRAP_FLAGS64 =       -64

#
# Various mapfiles that are used throughout the build, and delivered to
# /usr/lib/ld.
#
MAPFILE.NED_i386 =      $(SRC)/common/mapfiles/common/map.noexdata
MAPFILE.NED_sparc =
MAPFILE.NED =           $(MAPFILE.NED_$(MACH))
MAPFILE.PGA =           $(SRC)/common/mapfiles/common/map.pagealign
MAPFILE.NES =           $(SRC)/common/mapfiles/common/map.noexstk
MAPFILE.FLT =           $(SRC)/common/mapfiles/common/map.filter
MAPFILE.LEX =           $(SRC)/common/mapfiles/common/map.lex.yy

#
# Generated mapfiles that are compiler specific, and used throughout the
# build.  These mapfiles are not delivered in /usr/lib/ld.
#
MAPFILE.NGB_sparc=      $(SRC)/common/mapfiles/gen/sparc_cc_map.noexeglobs
$(__GNUC64)MAPFILE.NGB_sparc= \
                        $(SRC)/common/mapfiles/gen/sparc_gcc_map.noexeglobs
MAPFILE.NGB_sparcv9=    $(SRC)/common/mapfiles/gen/sparcv9_cc_map.noexeglobs
$(__GNUC64)MAPFILE.NGB_sparcv9= \
                        $(SRC)/common/mapfiles/gen/sparcv9_gcc_map.noexeglobs
MAPFILE.NGB_i386=       $(SRC)/common/mapfiles/gen/i386_cc_map.noexeglobs
$(__GNUC64)MAPFILE.NGB_i386= \
                        $(SRC)/common/mapfiles/gen/i386_gcc_map.noexeglobs
MAPFILE.NGB_amd64=      $(SRC)/common/mapfiles/gen/amd64_cc_map.noexeglobs
$(__GNUC64)MAPFILE.NGB_amd64= \
                        $(SRC)/common/mapfiles/gen/amd64_gcc_map.noexeglobs
MAPFILE.NGB =           $(MAPFILE.NGB_$(MACH))

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

#
# LDLIBS32 can be set in the environment to override the following assignment.
# LDLIBS64 can be set to override the assignment made in Makefile.master.64.
# These environment settings make sure that no libraries are searched outside
# of the local workspace proto area:
#       LDLIBS32=-YP,$ROOT/lib:$ROOT/usr/lib
#       LDLIBS64=-YP,$ROOT/lib/$MACH64:$ROOT/usr/lib/$MACH64
#
LDLIBS32 =      $(ENVLDLIBS1) $(ENVLDLIBS2) $(ENVLDLIBS3)
LDLIBS.cmd =    $(LDLIBS32)
LDLIBS.lib =    $(LDLIBS32)
#
# Define compilation macros.
#
COMPILE.c=      $(CC) $(CFLAGS) $(CPPFLAGS) -c
COMPILE64.c=    $(CC) $(CFLAGS64) $(CPPFLAGS) -c
COMPILE.cc=     $(CCC) $(CCFLAGS) $(CPPFLAGS) -c
COMPILE64.cc=   $(CCC) $(CCFLAGS64) $(CPPFLAGS) -c
COMPILE.s=      $(AS) $(ASFLAGS) $(AS_CPPFLAGS)
COMPILE64.s=    $(AS) $(ASFLAGS) $($(MACH64)_AS_XARCH) $(AS_CPPFLAGS)
COMPILE.d=      $(DTRACE) -G -32
COMPILE64.d=    $(DTRACE) -G -64
COMPILE.b=      $(ELFWRAP) $(ELFWRAP_FLAGS$(CLASS))
COMPILE64.b=    $(ELFWRAP) $(ELFWRAP_FLAGS$(CLASS))

CLASSPATH=      .
```



```
COMPILE.java=   $(JAVAC) $(JAVAFLAGS) -classpath $(CLASSPATH)

#
# Link time macros
#
CCNEEDED                = -lC
CCEXTNEEDED             = -lCrun -lCstd
$(__GNUC)CCNEEDED       = -L$(GCCLIBDIR) -R$(GCCLIBDIR) -lstdc++ -lgcc_s
$(__GNUC)CCEXTNEEDED    = $(CCNEEDED)

LINK.c=         $(CC) $(CFLAGS) $(CPPFLAGS) $(LDFLAGS)
LINK64.c=       $(CC) $(CFLAGS64) $(CPPFLAGS) $(LDFLAGS)
NORUNPATH=      -norunpath -nolib
LINK.cc=        $(CCC) $(CCFLAGS) $(CPPFLAGS) $(NORUNPATH) \
                $(LDFLAGS) $(CCNEEDED)
LINK64.cc=      $(CCC) $(CCFLAGS64) $(CPPFLAGS) $(NORUNPATH) \
                $(LDFLAGS) $(CCNEEDED)

#
# lint macros
#
# Note that the undefine of __PRAGMA_REDEFINE_EXTNAME can be removed once
# ON is built with a version of lint that has the fix for 4484186.
#
ALWAYS_LINT_DEFS =      -errtags=yes -s
ALWAYS_LINT_DEFS +=     -erroff=E_PTRDIFF_OVERFLOW
ALWAYS_LINT_DEFS +=     -erroff=E_ASSIGN_NARROW_CONV
ALWAYS_LINT_DEFS +=     -U__PRAGMA_REDEFINE_EXTNAME
ALWAYS_LINT_DEFS +=     $(C99LMODE)
ALWAYS_LINT_DEFS +=     -errsecurity=$(SECLEVEL)
ALWAYS_LINT_DEFS +=     -erroff=E_SEC_CREAT_WITHOUT_EXCL
ALWAYS_LINT_DEFS +=     -erroff=E_SEC_FORBIDDEN_WARN_CREAT
# XX64 -- really only needed for amd64 lint
ALWAYS_LINT_DEFS +=     -erroff=E_ASSIGN_INT_TO_SMALL_INT
ALWAYS_LINT_DEFS +=     -erroff=E_CAST_INT_CONST_TO_SMALL_INT
ALWAYS_LINT_DEFS +=     -erroff=E_CAST_INT_TO_SMALL_INT
ALWAYS_LINT_DEFS +=     -erroff=E_CAST_TO_PTR_FROM_INT
ALWAYS_LINT_DEFS +=     -erroff=E_COMP_INT_WITH_LARGE_INT
ALWAYS_LINT_DEFS +=     -erroff=E_INTEGRAL_CONST_EXP_EXPECTED
ALWAYS_LINT_DEFS +=     -erroff=E_PASS_INT_TO_SMALL_INT
ALWAYS_LINT_DEFS +=     -erroff=E_PTR_CONV_LOSES_BITS

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

SECLEVEL=       core
LINT.c=         $(LINT) $(ONLY_LINT_DEFS) $(LINTFLAGS) $(CPPFLAGS) \
                $(ALWAYS_LINT_DEFS)
LINT64.c=       $(LINT) $(ONLY_LINT_DEFS) $(LINTFLAGS64) $(CPPFLAGS) \
                $(ALWAYS_LINT_DEFS)
LINT.s=         $(LINT.c)

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

# Define native compilation macros
#

# Base directory where compilers are loaded.
# Defined here so it can be overridden by developer.
#
```

```
779 SPRO_ROOT=              $(BUILD_TOOLS)/SUNWspro
780 SPRO_VROOT=             $(SPRO_ROOT)/SS12
781 GNU_ROOT=               $(SFW_ROOT)

783 # Till SS12u1 formally becomes the NV CBE, LINT is hard
784 # coded to be picked up from the $SPRO_ROOT/sunstudio12.1/
785 # location. Impacted variables are sparc_LINT, sparcv9_LINT,
786 # i386_LINT, amd64_LINT.
787 # Reset them when SS12u1 is rolled out.
788 #

790 # Specify platform compiler versions for languages
791 # that we use (currently only c and c++).
792 #
793 sparc_CC=               $(ONBLD_TOOLS)/bin/$(MACH)/cw -_cc
794 $(__GNUC)sparc_CC=      $(ONBLD_TOOLS)/bin/$(MACH)/cw -_gcc
795 sparc_CCC=              $(ONBLD_TOOLS)/bin/$(MACH)/cw -_CC
796 $(__GNUC)sparc_CCC=     $(ONBLD_TOOLS)/bin/$(MACH)/cw -_g++
797 sparc_CPP=              /usr/ccs/lib/cpp
798 sparc_AS=               /usr/ccs/bin/as -xregsym=no
799 sparc_LD=               /usr/ccs/bin/ld
800 sparc_LINT=             $(SPRO_ROOT)/sunstudio12.1/bin/lint

802 sparcv9_CC=            $(ONBLD_TOOLS)/bin/$(MACH)/cw -_cc
803 $(__GNUC64)sparcv9_CC=  $(ONBLD_TOOLS)/bin/$(MACH)/cw -_gcc
804 sparcv9_CCC=           $(ONBLD_TOOLS)/bin/$(MACH)/cw -_CC
805 $(__GNUC64)sparcv9_CCC= $(ONBLD_TOOLS)/bin/$(MACH)/cw -_g++
806 sparcv9_CPP=           /usr/ccs/lib/cpp
807 sparcv9_AS=            /usr/ccs/bin/as -xregsym=no
808 sparcv9_LD=            /usr/ccs/bin/ld
809 sparcv9_LINT=          $(SPRO_ROOT)/sunstudio12.1/bin/lint

811 i386_CC=              $(ONBLD_TOOLS)/bin/$(MACH)/cw -_cc
812 $(__GNUC)i386_CC=      $(ONBLD_TOOLS)/bin/$(MACH)/cw -_gcc
813 i386_CCC=             $(ONBLD_TOOLS)/bin/$(MACH)/cw -_CC
814 $(__GNUC)i386_CCC=     $(ONBLD_TOOLS)/bin/$(MACH)/cw -_g++
815 i386_CPP=             /usr/ccs/lib/cpp
816 i386_AS=              /usr/ccs/bin/as
817 $(__GNUC)i386_AS=      $(ONBLD_TOOLS)/bin/$(MACH)/aw
818 i386_LD=              /usr/ccs/bin/ld
819 i386_LINT=            $(SPRO_ROOT)/sunstudio12.1/bin/lint

821 amd64_CC=             $(ONBLD_TOOLS)/bin/$(MACH)/cw -_cc
822 $(__GNUC64)amd64_CC=   $(ONBLD_TOOLS)/bin/$(MACH)/cw -_gcc
823 amd64_CCC=            $(ONBLD_TOOLS)/bin/$(MACH)/cw -_CC
824 $(__GNUC64)amd64_CCC=  $(ONBLD_TOOLS)/bin/$(MACH)/cw -_g++
825 amd64_CPP=            /usr/ccs/lib/cpp
826 amd64_AS=             $(ONBLD_TOOLS)/bin/$(MACH)/aw
827 amd64_LD=             /usr/ccs/bin/ld
828 amd64_LINT=           $(SPRO_ROOT)/sunstudio12.1/bin/lint

830 NATIVECC=             $($(NATIVE_MACH)_CC)
831 NATIVECCC=            $($(NATIVE_MACH)_CCC)
832 NATIVECPP=            $($(NATIVE_MACH)_CPP)
833 NATIVEAS=             $($(NATIVE_MACH)_AS)
834 NATIVELD=             $($(NATIVE_MACH)_LD)
835 NATIVELINT=           $($(NATIVE_MACH)_LINT)

837 #
838 # Makefile.master.64 overrides these settings
839 #
840 CC=                   $(NATIVECC)
841 CCC=                  $(NATIVECCC)
842 CPP=                  $(NATIVECPP)
843 AS=                   $(NATIVEAS)
844 LD=                   $(NATIVELD)
```



```
845 LINT=                 $(NATIVELINT)

847 # The real compilers used for this build
848 CW_CC_CMD=             $(CC) -_compiler
849 CW_CCC_CMD=            $(CCC) -_compiler
850 REAL_CC=               $(CW_CC_CMD:sh)
851 REAL_CCC=              $(CW_CCC_CMD:sh)

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

856 BDIRECT=        -Bdirect
857 BDYNAMIC=       -Bdynamic
858 BLOCAL=         -Blocal
859 BNODIRECT=      -Bnodirect
860 BREDUCE=        -Breduce
861 BSTATIC=        -Bstatic

863 ZDEFS=          -zdefs
864 ZDIRECT=        -zdirect
865 ZIGNORE=        -zignore
866 ZINITFIRST=     -zinitfirst
867 ZINTERPOSE=     -zinterpose
868 ZLAZYLOAD=      -zlazyload
869 ZLOADFLTR=      -zloadfltr
870 ZMULDEFS=       -zmuldefs
871 ZNODEFAULTLIB=  -znodefaultlib
872 ZNODEFS=        -znodefs
873 ZNODELETE=      -znodelete
874 ZNODLOPEN=      -znodlopen
875 ZNODUMP=        -znodump
876 ZNOLAZYLOAD=    -znolazyload
877 ZNOLDYNSYM=     -znoldynsym
878 ZNORELOC=       -znoreloc
879 ZNOVERSION=     -znoversion
880 ZRECORD=        -zrecord
881 ZREDLOCSYM=     -zredlocsym
882 ZTEXT=          -ztext
883 ZVERBOSE=       -zverbose

885 GSHARED=        -G
886 CCMT=           -mt

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

891 sparc_C_PICFLAGS =      -K pic
892 sparcv9_C_PICFLAGS =    -K pic
893 i386_C_PICFLAGS =       -K pic
894 amd64_C_PICFLAGS =      -K pic
895 C_PICFLAGS =            $($(MACH)_C_PICFLAGS)
896 C_PICFLAGS64 =          $($(MACH64)_C_PICFLAGS)

898 sparc_C_BIGPICFLAGS =   -K PIC
899 sparcv9_C_BIGPICFLAGS = -K PIC
900 i386_C_BIGPICFLAGS =    -K PIC
901 amd64_C_BIGPICFLAGS =   -K PIC
902 C_BIGPICFLAGS =         $($(MACH)_C_BIGPICFLAGS)
903 C_BIGPICFLAGS64 =       $($(MACH64)_C_BIGPICFLAGS)

905 # CC requires there to be no space between '-K' and 'pic' or 'PIC'.
906 sparc_CC_PICFLAGS =     -Kpic
907 sparcv9_CC_PICFLAGS =   -KPIC
908 i386_CC_PICFLAGS =      -Kpic
909 amd64_CC_PICFLAGS =     -Kpic
910 CC_PICFLAGS =           $($(MACH)_CC_PICFLAGS)
```

```
911 CC_PICFLAGS64 =         $($(MACH64)_CC_PICFLAGS)

913 AS_PICFLAGS=            $(C_PICFLAGS)
914 AS_BIGPICFLAGS=         $(C_BIGPICFLAGS)

916 #
917 # Default label for CTF sections
918 #
919 CTFCVTFLAGS=            -i -L VERSION
920 $(SRCDBGBLD)CTFCVTFLAGS         += -g
921 #endif /* ! codereview */

923 #
924 # Override to pass module-specific flags to ctfmerge.  Currently used only by
925 # krtld to turn on fuzzy matching, and source-level debugging to inhibit
926 # stripping.
888 # Override to pass module-specific flags to ctfmerge.  Currently used
889 # only by krtld to turn on fuzzy matching.
927 #
928 CTFMRGFLAGS=
929 $(SRCDBGBLD)CTFMRGFLAGS         += -g

931 #endif /* ! codereview */

933 CTFCONVERT_O            = $(CTFCONVERT) $(CTFCVTFLAGS) $@

935 ELFSIGN_O=      $(TRUE)
936 ELFSIGN_CRYPTO= $(ELFSIGN_O)
937 ELFSIGN_OBJECT= $(ELFSIGN_O)
938 $(EXPORT_RELEASE_BUILD)ELFSIGN_O =      $(ELFSIGN)
939 $(EXPORT_RELEASE_BUILD)ELFSIGN_CFNAME = SUNWosnetCF
940 $(EXPORT_RELEASE_BUILD)ELFSIGN_KEY =    \
941                         $(CLOSED)/cmd/cmd-crypto/etc/keys/$(ELFSIGN_CFNAME)
942 $(EXPORT_RELEASE_BUILD)ELFSIGN_CERT=    \
943                         $(CLOSED)/cmd/cmd-crypto/etc/certs/$(ELFSIGN_CFNAME)
944 $(EXPORT_RELEASE_BUILD)ELFSIGN_SENAME = SUNWosnetSE
945 $(EXPORT_RELEASE_BUILD)ELFSIGN_SEKEY =  \
946                         $(CLOSED)/cmd/cmd-crypto/etc/keys/$(ELFSIGN_SENAME)
947 $(EXPORT_RELEASE_BUILD)ELFSIGN_SECERT=  \
948                         $(CLOSED)/cmd/cmd-crypto/etc/certs/$(ELFSIGN_SENAME)
949 $(EXPORT_RELEASE_BUILD)ELFSIGN_CRYPTO=  $(ELFSIGN_O) sign \
950                         $(ELFSIGN_FORMAT_OPTION) \
951                         -k $(ELFSIGN_KEY) -c $(ELFSIGN_CERT) -e $@
952 $(EXPORT_RELEASE_BUILD)ELFSIGN_OBJECT=  $(ELFSIGN_O) sign \
953                         $(ELFSIGN_FORMAT_OPTION) \
954                         -k $(ELFSIGN_SEKEY) -c $(ELFSIGN_SECERT) -e $@

956 # Rules (normally from make.rules) and macros which are used for post
957 # processing files. Normally, these do stripping of the comment section
958 # automatically.
959 #    RELEASE_CM:        Should be editted to reflect the release.
960 #    POST_PROCESS_O:    Post-processing for '.o' files.
961 #    POST_PROCESS_A:    Post-processing for '.a' files (currently null).
962 #    POST_PROCESS_SO:   Post-processing for '.so' files.
963 #    POST_PROCESS:      Post-processing for executable files (no suffix).
964 # Note that these macros are not completely generalized as they are to be
965 # used with the file name to be processed following.
966 #
967 # It is left as an exercise to Release Engineering to embellish the generation
968 # of the release comment string.
969 #
970 #       If this is a standard development build:
971 #               compress the comment section (mcs -c)
972 #               add the standard comment (mcs -a $(RELEASE_CM))
973 #               add the development specific comment (mcs -a $(DEV_CM))
974 #
```



```
975 #       If this is an installation build:
976 #               delete the comment section (mcs -d)
977 #               add the standard comment (mcs -a $(RELEASE_CM))
978 #               add the development specific comment (mcs -a $(DEV_CM))
979 #
980 #       If this is an release build:
981 #               delete the comment section (mcs -d)
982 #               add the standard comment (mcs -a $(RELEASE_CM))
983 #
984 # The following list of macros are used in the definition of RELEASE_CM
985 # which is used to label all binaries in the build:
986 #
987 #       RELEASE         Specific release of the build, eg: 5.2
988 #       RELEASE_MAJOR   Major version number part of $(RELEASE)
989 #       RELEASE_MINOR   Minor version number part of $(RELEASE)
990 #       VERSION         Version of the build (alpha, beta, Generic)
991 #       PATCHID         If this is a patch this value should contain
992 #                       the patchid value (eg: "Generic 100832-01"), otherwise
993 #                       it will be set to $(VERSION)
994 #       RELEASE_DATE    Date of the Release Build
995 #       PATCH_DATE      Date the patch was created, if this is blank it
996 #                       will default to the RELEASE_DATE
997 #
998 RELEASE_MAJOR=  5
999 RELEASE_MINOR=  11
1000 RELEASE=        $(RELEASE_MAJOR).$(RELEASE_MINOR)
1001 VERSION=        SunOS Development
1002 PATCHID=        $(VERSION)
1003 RELEASE_DATE=   release date not set
1004 PATCH_DATE=     $(RELEASE_DATE)
1005 RELEASE_CM=     "@($(POUND_SIGN))SunOS $(RELEASE) $(PATCHID) $(PATCH_DATE)"
1006 DEV_CM=         "@($(POUND_SIGN))SunOS Internal Development: non-nightly build"

1008 PROCESS_COMMENT=                   @?${MCS} -c -a $(RELEASE_CM) -a $(DEV_CM)
1009 $(STRIP_COMMENTS)PROCESS_COMMENT=  @?${MCS} -d -a $(RELEASE_CM) -a $(DEV_CM)
1010 $(RELEASE_BUILD)PROCESS_COMMENT=   @?${MCS} -d -a $(RELEASE_CM)

1012 STRIP_STABS=                       :
1013 $(RELEASE_BUILD)STRIP_STABS=       $(STRIP) -x $@
1014 $(SRCDBGBLD)STRIP_STABS=           :
1015 #endif /* ! codereview */

1017 POST_PROCESS_O=         $(PROCESS_COMMENT) $@
1018 POST_PROCESS_A=
1019 POST_PROCESS_SO=        $(PROCESS_COMMENT) $@ ; $(STRIP_STABS) ; \
1020                         $(ELFSIGN_OBJECT)
1021 POST_PROCESS=           $(PROCESS_COMMENT) $@ ; $(STRIP_STABS) ; \
1022                         $(ELFSIGN_OBJECT)

1024 #
1025 # chk4ubin is a tool that inspects a module for a symbol table
1026 # ELF section size which can trigger an OBP bug on older platforms.
1027 # This problem affects only specific sun4u bootable modules.
1028 #
1029 CHK4UBIN=       $(ONBLD_TOOLS)/bin/$(MACH)/chk4ubin
1030 CHK4UBINFLAGS=
1031 CHK4UBINARY=    $(CHK4UBIN) $(CHK4UBINFLAGS) $@

1033 #
1034 # PKGARCHIVE specifies the default location where packages should be
1035 # placed if built.
1036 #
1037 $(RELEASE_BUILD)PKGARCHIVESUFFIX=       -nd
1038 PKGARCHIVE=$(SRC)/../../packages/$(MACH)/nightly$(PKGARCHIVESUFFIX)

1040 #
```

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

#       Default build rules which perform comment section post-processing.
#
.c:
        $(LINK.c) -o $@ $< $(LDLIBS)
        $(POST_PROCESS)
.c.o:
        $(COMPILE.c) $(OUTPUT_OPTION) $< $(CTFCONVERT_HOOK)
        $(POST_PROCESS_O)
.c.a:
        $(COMPILE.c) -o $% $<
        $(PROCESS_COMMENT) $%
        $(AR) $(ARFLAGS) $@ $%
        $(RM) $%
.s.o:
        $(COMPILE.s) -o $@ $<
        $(POST_PROCESS_O)
.s.a:
        $(COMPILE.s) -o $% $<
        $(PROCESS_COMMENT) $%
        $(AR) $(ARFLAGS) $@ $%
        $(RM) $%
.cc:
        $(LINK.cc) -o $@ $< $(LDLIBS)
        $(POST_PROCESS)
.cc.o:
        $(COMPILE.cc) $(OUTPUT_OPTION) $<
        $(POST_PROCESS_O)
.cc.a:
        $(COMPILE.cc) -o $% $<
        $(AR) $(ARFLAGS) $@ $%
        $(PROCESS_COMMENT) $%
        $(RM) $%
.y:
        $(YACC.y) $<
        $(LINK.c) -o $@ y.tab.c $(LDLIBS)
        $(POST_PROCESS)
        $(RM) y.tab.c
.y.o:
        $(YACC.y) $<
        $(COMPILE.c) -o $@ y.tab.c $(CTFCONVERT_HOOK)
        $(POST_PROCESS_O)
        $(RM) y.tab.c
.l:
        $(RM) $*.c
        $(LEX.l) $< > $*.c
        $(LINK.c) -o $@ $*.c -ll $(LDLIBS)
        $(POST_PROCESS)
        $(RM) $*.c
.l.o:
        $(RM) $*.c
        $(LEX.l) $< > $*.c
        $(COMPILE.c) -o $@ $*.c $(CTFCONVERT_HOOK)
        $(POST_PROCESS_O)
        $(RM) $*.c

.bin.o:
        $(COMPILE.b) -o $@ $<
        $(POST_PROCESS_O)
```



```
.java.class:
        $(COMPILE.java) $<

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

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

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

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

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

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

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

.pl.po .pm.po:
        $(XGETTEXT) $(XGETFLAGS) -d $(<F) $< ;
        $(RM)   $@ ;
        $(SED) "/^domain/d" < $(<F).po > $@ ;
        $(RM) $(<F).po

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

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

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

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

.l.i:
        $(LEX) $<
        $(CPPFORPO) lex.yy.c  > $@
```

```
1173         $(RM) lex.yy.c

1175 .c.po:
1176         $(CPPFORPO) $< > $<.i
1177         $(BUILD.po)

1179 .y.po:
1180         $(YACC) -d $<
1181         $(CPPFORPO) y.tab.c  > $<.i
1182         $(BUILD.po)
1183         $(RM) y.tab.c

1185 .l.po:
1186         $(LEX) $<
1187         $(CPPFORPO) lex.yy.c  > $<.i
1188         $(BUILD.po)
1189         $(RM) lex.yy.c

1191 #
1192 # Rules to perform stylistic checks
1193 #
1194 .SUFFIXES: .x .xml .check .xmlchk

1196 .h.check:
1197         $(DOT_H_CHECK)

1199 .x.check:
1200         $(DOT_X_CHECK)

1202 .xml.xmlchk:
1203         $(MANIFEST_CHECK)

1205 #
1206 # Rules to process ONC+ Source partial files
1207 #
1208 %_onc_plus:     %
1209         @$(ECHO) "extracting code from $< ... "
1210         sed -n -e '/ONC_PLUS EXTRACT START/,/ONC_PLUS EXTRACT END/p' $<  > $@

1212 #
1213 # Include rules to render automated sccs get rules "safe".
1214 #
1215 include $(SRC)/Makefile.noget
```

****
1086 Sun Jul 28 20:29:07 2013
new/usr/src/cmd/Makefile.ctf
3735 should include an empty make variable in the default CFLAGS/CCFLAGS
3844 the build should make source-level debugging easier
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Garrett D'Amore <garrett@damore.org>
****

```
   1 #
   2 # CDDL HEADER START
   3 #
   4 # The contents of this file are subject to the terms of the
   5 # Common Development and Distribution License (the "License").
   6 # You may not use this file except in compliance with the License.
   7 #
   8 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
   9 # or http://www.opensolaris.org/os/licensing.
  10 # See the License for the specific language governing permissions
  11 # and limitations under the License.
  12 #
  13 # When distributing Covered Code, include this CDDL HEADER in each
  14 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  15 # If applicable, add the following below this CDDL HEADER, with the
  16 # fields enclosed by brackets "[]" replaced with your own identifying
  17 # information: Portions Copyright [yyyy] [name of copyright owner]
  18 #
  19 # CDDL HEADER END
  20 #
  21 #
  22 # Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
  23 # Use is subject to license terms.
  24 #

  26 POST_PROCESS += ; $(CTFMERGE) $(CTFMRGFLAGS) -L VERSION -o $@ $(OBJS)
  26 POST_PROCESS += ; $(CTFMERGE) -L VERSION -o $@ $(OBJS)
  27 POST_PROCESS_O += ; $(CTFCONVERT_O)

  29 CFLAGS += $(CTF_FLAGS)
  30 CFLAGS64 += $(CTF_FLAGS)
  31 NATIVE_CFLAGS += $(CTF_FLAGS)
```

```
**********************************************************
    8882 Sun Jul 28 20:29:07 2013
new/usr/src/lib/Makefile.lib
3735 should include an empty make variable in the default CFLAGS/CCFLAGS
3844 the build should make source-level debugging easier
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Garrett D'Amore <garrett@damore.org>
**********************************************************
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
# Copyright (c) 1989, 2010, Oracle and/or its affiliates. All rights reserved.
#
#
# Definitions common to libraries.
#
# include global definitions; SRC should be defined in the shell.
# SRC is needed until RFE 1026993 is implemented.

include         $(SRC)/Makefile.master

LORDER=         lorder
TSORT=          tsort
AWK=            awk

#
# By default, we define the source directory for libraries to be
# one level up from the ISA-specific directory, where the code is
# actually built.  Many libraries define a 'common' directory to
# contain the source.  These libraries must redefine SRCDIR as:
#       SRCDIR = ../common
# Other variations are possible (../port, ../src, etc).
#
SRCDIR =        ..

#
# We define MAPFILES here for the benefit of most libraries, those that
# follow the convention of having source files and other common files
# in the $(SRCDIR) directory.  Libraries that do not follow this
# convention must define MAPFILES, or MAPFILEDIR for themselves.
# Libraries that do follow this convention but that need supplemental
# ISA-specific mapfiles can augment MAPFILES like this:
#       MAPFILES += mapfile-vers
#
MAPFILEDIR =    $(SRCDIR)
MAPFILES =      $(MAPFILEDIR)/mapfile-vers

#
# If HDRDIR is left unset, then it's possible for the $(ROOTHDRDIR)/%
```



```
# install rule in lib/Makefile.targ to generate false matches if there
# are any common directory names between / and /usr/include ('xfn' is
# one common example).  To prevent this, we set HDRDIR to a directory
# name that will almost surely not exist on the build machine.
#
HDRDIR=         /__nonexistent_directory__

#
# We don't build archive (*.a) libraries by default anymore.
# If a component of the build needs to build an archive library
# for its own internal purposes, it can define LIBS for itself
# after including Makefile.lib, like this:
#       LIBS = $(LIBRARY)
# or:
#       LIBS = $(LIBRARYCCC)
# Archive libraries must not be installed in the proto area.
#
LIBS=
MACHLIBS=       $(LIBS:%=$(MACH)/%)
MACHLIBS64=     $(LIBS:%=$(MACH64)/%)
DYNLIB=         $(LIBRARY:.a=.so$(VERS))
DYNLIBPSR=      $(LIBRARY:.a=_psr.so$(VERS))
DYNLIBCCC=      $(LIBRARYCCC:.a=.so$(VERS))
LIBLINKS=       $(LIBRARY:.a=.so)
LIBLINKSCCC=    $(LIBRARYCCC:.a=.so)
LIBNAME=        $(LIBRARY:lib%.a=%)
LIBLINKPATH=
LIBNULL=        null.a
ROOTHDRDIR=     $(ROOT)/usr/include
ROOTLIBDIR=     $(ROOT)/usr/lib
ROOTLIBDIR64=   $(ROOT)/usr/lib/$(MACH64)
ROOTFS_LIBDIR=  $(ROOT)/lib
ROOTFS_LIBDIR64=        $(ROOT)/lib/$(MACH64)
ROOTLINTDIR=    $(ROOTLIBDIR)
ROOTFS_LINTDIR= $(ROOTFS_LIBDIR)
ROOTFS_LINTDIR64=       $(ROOTFS_LIBDIR64)
ROOTHDRS=       $(HDRS:%=$(ROOTHDRDIR)/%)
HDRSRCS=        $(HDRS:%=$(HDRDIR)/%)
CHECKHDRS=      $(HDRSRCS:%.h=%.check)
ROOTLIBS=       $(LIBS:%=$(ROOTLIBDIR)/%)
ROOTLIBS64=     $(LIBS:%=$(ROOTLIBDIR64)/%)
ROOTFS_LIBS=    $(DYNLIB:%=$(ROOTFS_LIBDIR)/%)
ROOTFS_LIBS64=  $(DYNLIB:%=$(ROOTFS_LIBDIR64)/%)
ROOTLINKS=      $(ROOTLIBDIR)/$(LIBLINKS)
ROOTLINKS64=    $(ROOTLIBDIR64)/$(LIBLINKS)
ROOTFS_LINKS=   $(ROOTFS_LIBDIR)/$(LIBLINKS)
ROOTFS_LINKS64= $(ROOTFS_LIBDIR64)/$(LIBLINKS)
ROOTLINKSCCC=   $(ROOTLIBDIR)/$(LIBLINKSCCC)
ROOTLINKSCCC64= $(ROOTLIBDIR64)/$(LIBLINKSCCC)
ROOTFS_LINKSCCC=        $(ROOTFS_LIBDIR)/$(LIBLINKSCCC)
ROOTFS_LINKSCCC64=      $(ROOTFS_LIBDIR64)/$(LIBLINKSCCC)
ROOTLINT=       $(LINTSRC:%=$(ROOTLINTDIR)/%)
ROOTFS_LINT=    $(LINTSRC:%=$(ROOTFS_LINTDIR)/%)
ROOTFS_LINT64=  $(LINTSRC:%=$(ROOTFS_LINTDIR64)/%)
ROOTMAN3=     $(ROOT)/usr/share/man/man3
ROOTMAN3FILES=        $(MAN3FILES:%=$(ROOTMAN3)/%)
$(ROOTMAN3FILES) := FILEMODE= 444

# Demo rules
DEMOFILES=
DEMOFILESRCDIR=         common
ROOTDEMODIRBASE=        __nonexistent_directory__
ROOTDEMODIRS=
ROOTDEMOFILES=  $(DEMOFILES:%=$(ROOTDEMODIRBASE)/%)
$(ROOTDEMODIRS) :=      DIRMODE =       755
```

```
125 LINTLIB=        llib-l$(LIBNAME).ln
126 LINTFLAGS=      -uaxm
127 LINTFLAGS64=    -uaxm -m64
128 LINTSRC=        $(LINTLIB:%.ln=%)
129 LINTOUT=        lint.out
130 ARFLAGS=        r
131 SONAME=         $(DYNLIB)
132 # For most libraries, we should be able to resolve all symbols at link time,
133 # either within the library or as dependencies, all text should be pure, and
134 # combining relocations into one relocation table reduces startup costs.
135 # All options are tunable to allow overload/omission from lower makefiles.


138 HSONAME=        -h$(SONAME)
139 DYNFLAGS=       $(HSONAME) $(ZTEXT) $(ZDEFS) $(BDIRECT) \
140                 $(MAPFILES:%=-M%) $(MAPFILE.PGA:%=-M%) $(MAPFILE.NED:%=-M%)

142 LDLIBS=         $(LDLIBS.lib)

144 OBJS=           $(OBJECTS:%=objs/%)
145 PICS=           $(OBJECTS:%=pics/%)

147 # Declare that all library .o's can all be made in parallel.
148 # The DUMMY target is for those instances where OBJS and PICS
149 # are empty (to avoid an unconditional .PARALLEL declaration).
150 .PARALLEL:      $(OBJS) $(PICS) DUMMY

152 # default value for "portable" source
153 SRCS=           $(OBJECTS:%.o=$(SRCDIR)/%.c)

155 # default build of an archive and a shared object,
156 # overridden locally when extra processing is needed
157 BUILD.AR=       $(AR) $(ARFLAGS) $@ $(AROBJS)
158 BUILD.SO=       $(CC) -o $@ $(GSHARED) $(DYNFLAGS) $(PICS) $(EXTPICS) $(LDLIBS)
159 BUILDCCC.SO=    $(CCC) -o $@ $(GSHARED) $(DYNFLAGS) $(PICS) $(EXTPICS) $(LDLIBS)

161 # default dynamic library symlink
162 # IMPORTANT:: If you change INS.liblink OR INS.liblink64 here, then you
163 # MUST also change the corresponding override definitions in
164 # $CLOSED/Makefile.tonic.
165 #
166 # If you do not do this, then the closedbins build for the OpenSolaris
167 # community will break. PS, the gatekeepers will be upset too.
168 #
169 INS.liblink=    -$(RM) $@; $(SYMLINK) $(LIBLINKPATH)$(LIBLINKS)$(VERS) $@
170 INS.liblinkccc= -$(RM) $@; $(SYMLINK) $(LIBLINKPATH)$(LIBLINKSCCC)$(VERS) $@

172 # default 64-bit dynamic library symlink
173 INS.liblink64=  -$(RM) $@; $(SYMLINK) $(LIBLINKPATH)$(LIBLINKS)$(VERS) $@
174 INS.liblinkccc64= -$(RM) $@; $(SYMLINK) $(LIBLINKPATH)$(LIBLINKSCCC)$(VERS) $@

176 #
177 # If appropriate, augment POST_PROCESS_O and POST_PROCESS_SO to do CTF
178 # processing.  We'd like to just conditionally append to POST_PROCESS_O and
179 # POST_PROCESS_SO, but ParallelMake has a bug which causes the same value to
180 # sometimes get appended more than once, which will cause ctfconvert to fail.
181 # So, instead we introduce CTFCONVERT_POST and CTFMERGE_POST, which are always
182 # appended to POST_PROCESS_O and POST_PROCESS_SO but are no-ops unless CTF
183 # processing should be done.
184 #
185 CTFCONVERT_POST = :
186 CTFMERGE_POST   = :
187 POST_PROCESS_O += ; $(CTFCONVERT_POST)
188 POST_PROCESS_SO += ; $(CTFMERGE_POST)

190 CTFMERGE_LIB    = $(CTFMERGE) $(CTFMRGFLAGS) -t -f -L VERSION -o $@ $(PICS)
```



```
190 CTFMERGE_LIB    = $(CTFMERGE) -t -f -L VERSION -o $@ $(PICS)

192 # conditional assignments

194 $(OBJS)  :=     sparc_CFLAGS += -xregs=no%appl

196 $(PICS)  :=     sparc_CFLAGS += -xregs=no%appl $(sparc_C_PICFLAGS)
197 $(PICS)  :=     sparcv9_CFLAGS += -xregs=no%appl $(sparcv9_C_PICFLAGS)
198 $(PICS)  :=     i386_CFLAGS += $(i386_C_PICFLAGS)
199 $(PICS)  :=     amd64_CFLAGS += $(amd64_C_PICFLAGS)
200 $(PICS)  :=     CCFLAGS += $(CC_PICFLAGS)
201 $(PICS)  :=     CPPFLAGS += -DPIC -D_REENTRANT
202 $(PICS)  :=     sparcv9_CCFLAGS += -xregs=no%appl $(sparcv9_CC_PICFLAGS)
203 $(PICS)  :=     amd64_CCFLAGS += $(amd64_CC_PICFLAGS)
204 $(PICS)  :=     CFLAGS += $(CTF_FLAGS)
205 $(PICS)  :=     CFLAGS64 += $(CTF_FLAGS)
206 $(PICS)  :=     CTFCONVERT_POST = $(CTFCONVERT_O)
207 $(DYNLIB) :=    CTFMERGE_POST = $(CTFMERGE_LIB)

209 $(LINTLIB):=    LOG = -DLOGGING
210 $(LIBRARY):=    AROBJS = $(OBJS)
211 $(LIBRARY):=    DIR = objs
212 $(DYNLIB):=     DIR = pics
213 $(DYNLIBCCC):=  DIR = pics

215 SONAMECCC=      $(DYNLIBCCC)
216 HSONAMECCC=     -h $(SONAMECCC)
217 #
218 # Keep in sync with the standard DYNFLAGS
219 #
220 $(DYNLIBCCC):=  DYNFLAGS = $(HSONAMECCC) $(ZTEXT) $(ZDEFS) \
221                 $(MAPFILES:%=-M%) $(MAPFILE.PGA:%=-M%) $(MAPFILE.NED:%=-M%) \
222                 $(BDIRECT) $(NORUNPATH)


225 # build rule for "portable" source
226 objs/%.o pics/%.o: %.c
227         $(COMPILE.c) -o $@ $<
228         $(POST_PROCESS_O)

230 objs/%.o pics/%.o: %.cc
231         $(COMPILE.cc) -o $@ $<
232         $(POST_PROCESS_O)

234 .PRECIOUS: $(LIBS)

236 # Define the majority text domain in this directory.
237 TEXT_DOMAIN= SUNW_OST_OSLIB

239 $(ROOTMAN3)/%: %.sunman
240         $(INS.rename)

242 #
243 # For library source code, we expect that some symbols may not be used or
244 # may *appear* to be able to rescoped to static; shut lint up.  Never add
245 # a flag here unless you're *sure* that all libraries need to be linted
246 # with it.
247 #
248 LINTCHECKFLAGS = -m -erroff=E_NAME_DEF_NOT_USED2
249 LINTCHECKFLAGS += -erroff=E_NAME_DECL_NOT_USED_DEF2

251 #
252 # Target Architecture
253 #
254 TARGETMACH=     $(MACH)
```

```
256 #
257 # Allow people to define their own clobber rules.  Normal makefiles
258 # shouldn't override this - they should override $(CLOBBERFILES) instead.
259 #
260 CLOBBERTARGFILES= $(LIBS) $(DYNLIB) $(CLOBBERFILES)
```

```
**********************************************************
   23258 Sun Jul 28 20:29:08 2013
new/usr/src/uts/Makefile.uts
3735 should include an empty make variable in the default CFLAGS/CCFLAGS
3844 the build should make source-level debugging easier
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Garrett D'Amore <garrett@damore.org>
**********************************************************
```

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#

#
# Copyright (c) 1991, 2010, Oracle and/or its affiliates. All rights reserved.
# Copyright (c) 2011 Bayard G. Bell. All rights reserved.
# Copyright (c) 2011 by Delphix. All rights reserved.
#

#
#       This Makefile contains the common targets and definitions for
#       all kernels. It is to be included in the Makefiles for specific
#       implementation architectures and processor architecture dependent
#       modules: i.e.: all driving kernel Makefiles.
#
#       Include global definitions:
#
include $(SRC)/Makefile.master

#
#       No text domain in the kernel.
#
DTEXTDOM =

#
#       Keep references to $(SRC)/common relative.
COMMONBASE=     $(UTSBASE)/../common

#
#       Setup build-specific vars
#       To add a build type:
#               add name to ALL_BUILDS32 & ALL_BUILDS64
#               set CLASS_name and OBJ_DIR_name
#               add targets to Makefile.targ
#


#
#       DEF_BUILDS is for def, lint, sischeck, and install
#       ALL_BUILDS is for everything else (all, clean, ...)
#
```



```
# The NOT_RELEASE_BUILD noise is to maintain compatibility with the
# gatekeeper's nightly build script.
#
DEF_BUILDS32                            = obj32
DEF_BUILDS64                            = obj64
DEF_BUILDSONLY64                        = obj64
$(NOT_RELEASE_BUILD)DEF_BUILDS32        = debug32
$(NOT_RELEASE_BUILD)DEF_BUILDS64        = debug64
$(NOT_RELEASE_BUILD)DEF_BUILDSONLY64    = debug64
ALL_BUILDS32                            = obj32 debug32
ALL_BUILDS64                            = obj64 debug64
ALL_BUILDSONLY64                        = obj64 debug64

#
#       For modules in 64b dirs that aren't built 64b
#       or modules in 64b dirs that aren't built 32b we
#       need to create empty modlintlib files so global lint works
#
LINT32_BUILDS   = debug32
LINT64_BUILDS   = debug64

#
#       Build class (32b or 64b)
#
CLASS_OBJ32     = 32
CLASS_DBG32     = 32
CLASS_OBJ64     = 64
CLASS_DBG64     = 64
CLASS           = $(CLASS_$(BUILD_TYPE))

#
#       Build subdirectory
#
OBJS_DIR_OBJ32  = obj32
OBJS_DIR_DBG32  = debug32
OBJS_DIR_OBJ64  = obj64
OBJS_DIR_DBG64  = debug64
OBJS_DIR        = $(OBJS_DIR_$(BUILD_TYPE))

#
#       Create defaults so empty rules don't
#       confuse make
#
CLASS_          = 32
OBJS_DIR_       = debug32

#
#       Build tools
#
CC_sparc_32     = $(sparc_CC)
CC_sparc_64     = $(sparcv9_CC)

CC_i386_32      = $(i386_CC)
CC_i386_64      = $(amd64_CC)
CC_amd64_64     = $(amd64_CC)

CC              = $(CC_$(MACH)_$(CLASS))

AS_sparc_32     = $(sparc_AS)
AS_sparc_64     = $(sparcv9_AS)

AS_i386_32      = $(i386_AS)
AS_i386_64      = $(amd64_AS)
AS_amd64_64     = $(amd64_AS)

AS              = $(AS_$(MACH)_$(CLASS))
```

```
126 LD_sparc_32     = $(sparc_LD)
127 LD_sparc_64     = $(sparcv9_LD)

129 LD_i386_32      = $(i386_LD)
130 LD_i386_64      = $(amd64_LD)
131 LD_amd64_64     = $(amd64_LD)

133 LD              = $(LD_$(MACH)_$(CLASS))

135 LINT_sparc_32   = $(sparc_LINT)
136 LINT_sparc_64   = $(sparcv9_LINT)

138 LINT_i386_32    = $(i386_LINT)
139 LINT_i386_64    = $(amd64_LINT)
140 LINT_amd64_64   = $(amd64_LINT)

142 LINT            = $(LINT_$(MACH)_$(CLASS))

144 MODEL_32        = ilp32
145 MODEL_64        = lp64
146 MODEL           = $(MODEL_$(CLASS))

148 #
149 #       Build rules for linting the kernel.
150 #
151 LHEAD = $(ECHO) "\n$@";

153 # Note: egrep returns "failure" if there are no matches, which is
154 # exactly the opposite of what we need.
155 LGREP.2 =       if egrep -v ' (_init|_fini|_info) '; then false; else true; fi

157 LTAIL =

159 LINT.c =        $(LINT) -c -dirout=$(LINTS_DIR) $(LINTFLAGS) $(LINT_DEFS) $(CPPF

161 # Please do not add new erroff directives here.  If you need to disable
162 # lint warnings in your module for things that cannot be fixed in any
163 # reasonable manner, please augment LINTTAGS in your module Makefile
164 # instead.
165 LINTTAGS        = -erroff=E_INCONS_ARG_DECL2
166 LINTTAGS        += -erroff=E_INCONS_VAL_TYPE_DECL2

168 LINTFLAGS_sparc_32      = $(LINTCCMODE) -nsxmuF -errtags=yes
169 LINTFLAGS_sparc_64      = $(LINTFLAGS_sparc_32) -m64
170 LINTFLAGS_i386_32       = $(LINTCCMODE) -nsxmuF -errtags=yes
171 LINTFLAGS_i386_64       = $(LINTFLAGS_i386_32) -m64

173 LINTFLAGS       = $(LINTFLAGS_$(MACH)_$(CLASS)) $(LINTTAGS)
174 LINTFLAGS       += $(C99LMODE)

176 #
177 #       Override this variable to modify the name of the lint target.
178 #
179 LINT_MODULE=    $(MODULE)

181 #
182 #       Build the compile/assemble lines:
183 #
184 EXTRA_OPTIONS           =
185 AS_DEFS                 = -D_ASM -D__STDC__=0

187 ALWAYS_DEFS_32          = -D_KERNEL -D_SYSCALL32 -D_DDI_STRICT
188 ALWAYS_DEFS_64          = -D_KERNEL -D_SYSCALL32 -D_SYSCALL32_IMPL -D_ELF64 \
189                         -D_DDI_STRICT
190 #
```



```
191 # XX64  This should be defined by the compiler!
192 #
193 ALWAYS_DEFS_64          += -Dsun -D__sun -D__SVR4
194 ALWAYS_DEFS             = $(ALWAYS_DEFS_$(CLASS))

196 #
197 #       CPPFLAGS is deliberatly set with a "=" and not a "+=".  For the kernel
198 #       the header include path should not look for header files outside of
199 #       the kernel code.  This "=" removes the search path built in
200 #       Makefile.master inside CPPFLAGS.  Ditto for AS_CPPFLAGS.
201 #
202 CPPFLAGS        = $(ALWAYS_DEFS) $(ALL_DEFS) $(CONFIG_DEFS) \
203                     $(INCLUDE_PATH) $(EXTRA_OPTIONS)
204 ASFLAGS         += -P
205 AS_CPPFLAGS     = $(ALWAYS_DEFS) $(ALL_DEFS) $(CONFIG_DEFS) $(AS_DEFS) \
206                     $(AS_INC_PATH) $(EXTRA_OPTIONS)

208 #
209 #       Make it (relatively) easy to share compilation options between
210 #       all kernel implementations.
211 #

213 # Override the default, the kernel is squeaky clean
214 CERRWARN = -errtags=yes -errwarn=%all

216 CERRWARN += -_gcc=-Wno-missing-braces
217 CERRWARN += -_gcc=-Wno-sign-compare
218 CERRWARN += -_gcc=-Wno-unknown-pragmas
219 CERRWARN += -_gcc=-Wno-unused-parameter
220 CERRWARN += -_gcc=-Wno-missing-field-initializers

222 # DEBUG v. -nd make for frequent unused variables, empty conditions, etc. in
223 # -nd builds
224 $(RELEASE_BUILD)CERRWARN += -_gcc=-Wno-unused
225 $(RELEASE_BUILD)CERRWARN += -_gcc=-Wno-empty-body

227 C99MODE = $(C99_ENABLE)

229 CFLAGS_uts              =
230 CFLAGS_uts              += $(STAND_FLAGS_$(CLASS))
231 CFLAGS_uts              += $(CCVERBOSE)
232 CFLAGS_uts              += $(ILDOFF)
233 CFLAGS_uts              += $(XAOPT)
234 CFLAGS_uts              += $(CTF_FLAGS_$(CLASS))
235 CFLAGS_uts              += $(CERRWARN)
236 CFLAGS_uts              += $(CCNOAUTOINLINE)
237 CFLAGS_uts              += $(CGLOBALSTATIC)
238 CFLAGS_uts              += $(EXTRA_CFLAGS)
239 CFLAGS_uts              += $(CSOURCEDEBUGFLAGS)
240 CFLAGS_uts              += $(CUSERFLAGS)
241 #endif /* ! codereview */

243 #
244 #       Declare that $(OBJECTS) and $(LINTS) can be compiled in parallel.
245 #       The DUMMY target is for those instances where OBJECTS and LINTS
246 #       are empty (to avoid an unconditional .PARALLEL).
247 .PARALLEL:      $(OBJECTS) $(LINTS) DUMMY

249 #
250 #       Expanded dependencies
251 #
252 DEF_DEPS        = $(DEF_BUILDS:%=def.%)
253 ALL_DEPS        = $(ALL_BUILDS:%=all.%)
254 CLEAN_DEPS      = $(ALL_BUILDS:%=clean.%)
255 CLOBBER_DEPS    = $(ALL_BUILDS:%=clobber.%)
256 LINT_DEPS       = $(DEF_BUILDS:%=lint.%)
```

```
MODLINTLIB_DEPS = $(DEF_BUILDS:%=modlintlib.%)
MODLIST_DEPS    = $(DEF_BUILDS:%=modlist.%)
CLEAN_LINT_DEPS = $(ALL_BUILDS:%=clean.lint.%)
INSTALL_DEPS    = $(DEF_BUILDS:%=install.%)
SYM_DEPS        = $(SYM_BUILDS:%=symcheck.%)
SISCHECK_DEPS   = $(DEF_BUILDS:%=sischeck.%)
SISCLEAN_DEPS   = $(ALL_BUILDS:%=sisclean.%)

#
#       Default module name
#
BINARY          = $(OBJS_DIR)/$(MODULE)

#
#       Default cleanup definitions
#
CLEANLINTFILES  = $(LINTS) $(MOD_LINT_LIB)
CLEANFILES      = $(OBJECTS) $(CLEANLINTFILES)
CLOBBERFILES    = $(BINARY) $(CLEANFILES)

#
#       Installation constants:
#
#               FILEMODE is the mode given to the kernel modules
#               CFILEMODE is the mode given to the '.conf' files
#
FILEMODE         = 755
DIRMODE          = 755
CFILEMODE        = 644

#
#       Special Installation Macros for the installation of '.conf' files.
#
#       These are unique because they are not installed from the current
#       working directory.
#
# Sigh.  Apparently at some time in the past there was a confusion on
# whether the name is SRC_CONFFILE or SRC_CONFILE.  Consistency with the
# other names would indicate SRC_CONFFILE, but the voting is >180 Makefiles
# with SRC_CONFILE and about 11 with SRC_CONFFILE.  Software development
# isn't a popularity contest, though, and so my inclination is to define
# both names for now and incrementally convert to SRC_CONFFILE to be consistent
# with the other names.
#
CONFFILE                = $(MODULE).conf
SRC_CONFFILE            = $(CONF_SRCDIR)/$(CONFFILE)
SRC_CONFILE             = $(SRC_CONFFILE)
ROOT_CONFFILE_32        = $(ROOTMODULE).conf
ROOT_CONFFILE_64        = $(ROOTMODULE:%/$(SUBDIR64)/$(MODULE)=%/$(MODULE)).conf
ROOT_CONFFILE           = $(ROOT_CONFFILE_$(CLASS))


INS.conffile= \
        $(RM) $@; $(INS) -s -m $(CFILEMODE) -f $(@D) $(SRC_CONFFILE)

#
# The CTF merge of child kernel modules is performed against one of the genunix
# modules.  For Intel builds, all modules will be used with a single genunix:
# the one built in intel/genunix.  For SPARC builds, a given
# module may be
# used with one of a number of genunix files, depending on what platform the
# module is deployed on.  We merge against the sun4u genunix to optimize for
# the common case.  We also merge against the ip driver since networking is
# typically loaded and types defined therein are shared between many modules.
#
CTFMERGE_GUDIR_sparc    = sun4u
```



```
CTFMERGE_GUDIR_i386     = intel
CTFMERGE_GUDIR          = $(CTFMERGE_GUDIR_$(MACH))

CTFMERGE_GENUNIX        = \
        $(UTSBASE)/$(CTFMERGE_GUDIR)/genunix/$(OBJS_DIR)/genunix

#
# Used to uniquify a non-genunix module against genunix.  If used in patch
# mode (PATCH_BUILD != "#"), the patch ID corresponding to the module being
# built will be used as the label.  If no ID is available, or if patch mode
# is not being used, the value of $VERSION will be used.
#
# For the ease of developers dropping modules onto possibly unrelated systems,
# you can set NO_GENUNIX_UNIQUIFY= in the environment to skip uniquifying
# against genunix.
#
NO_GENUNIX_UNIQUIFY=$(POUND_SIGN)
SKIP_GENUNIX_UNIQUIFY=no
$(NO_GENUNIX_UNIQUIFY)SKIP_GENUNIX_UNIQUIFY=yes

CTFMERGE_UNIQUIFY_AGAINST_GENUNIX       = \
        @label="-L VERSION" ; \
        uniq= ; \
        if [ -z "$(PATCH_BUILD)" ] ; then \
                uniq="-D BASE" ; \
                set -- `$(CTFFINDMOD) -n -r -t $(PMTMO_FILE) $@` ; \
                if [ "X$$1" != "X-" ] ; then \
                        label="-l $$1" ; \
                        if [ "$$2" != "fcs" ] ; then \
                                uniq="-D $$2" ; \
                        fi ; \
                fi ; \
        fi ; \
        if [ "$(SKIP_GENUNIX_UNIQUIFY)" = "yes" ]; then \
                uniq= ; \
        else \
                uniq="-d $(CTFMERGE_GENUNIX) $$uniq" ; \
        fi ; \
        cmd="$(CTFMERGE) $(CTFMRGFLAGS) $$label $$uniq" ; \
        cmd="$$cmd -o $@ $(OBJECTS) $(CTFEXTRAOBJS)" ; \
        echo $$cmd ; \
        $$cmd

#
# Used to merge the genunix module.  genunix has special requirements in
# patch mode.  In particular, it needs to be able to find the genunix used
# in the previous version of the KU patch (or the FCS version of genunix in
# the case of KU 1).
#
CTFMERGE_GENUNIX_MERGE          = \
        @if [ -z "$(PATCH_BUILD)" ] ; then \
                set -- `$(CTFFINDMOD) -b $(OBJS_DIR) -o patch,lastgu -n -r \
                    -t $(PMTMO_FILE) $(GENUNIX) || true` '' ; \
                msg= ; \
                if [ $$$(POUND_SIGN) -eq 1 ] ; \
                    then msg="Error in $(CTFFINDMOD)" ; \
                elif [ "X$$1" = "X-" ] ; then msg="Did not get label" ; \
                elif [ "X$$2" = "X-" ] ; then msg="Did not get withfile" ; \
                fi ; \
                if [ -n "$$msg" ] ; then \
                        echo "make ctf: $$msg - removing $(GENUNIX)" ; \
                        $(RM) $(GENUNIX) ; \
                        exit 1 ; \
                fi ; \
                label="-l $$1" ; \
                with="-w $$2" ; \
```

```
389         else \
390                 label="-L VERSION" ; \
391         fi ; \
392         cmd="$(CTFMERGE) $(CTFMRGFLAGS) $$label $$with -o $@" ; \
393         echo $$cmd "$(OBJECTS) $(CTFEXTRAOBJS) $(IPCTF_TARGET)"; \
394         $$cmd $(OBJECTS) $(CTFEXTRAOBJS) $(IPCTF_TARGET)

396 #
397 # We ctfmerge the ip objects into genunix to maximize the number of common types
398 # found there, thus maximizing the effectiveness of uniquification.  We don't
399 # want the genunix build to have to know about the individual ip objects, so we
400 # put them in an archive.  The genunix ctfmerge then includes this archive.
401 #
402 IPCTF           = $(IPDRV_DIR)/$(OBJS_DIR)/ipctf.a

404 #
405 # Rule for building fake shared libraries used for symbol resolution
406 # when building other modules.  -znoreloc is needed here to avoid
407 # tripping over code that isn't really suitable for shared libraries.
408 #
409 BUILD.SO                = \
410         $(LD) -o $@ $(GSHARED) $(ZNORELOC) -h $(SONAME)

412 #
413 # SONAME defaults for common fake shared libraries.
414 #
415 $(LIBGEN)               := SONAME = $(MODULE)
416 $(PLATLIB)              := SONAME = misc/platmod
417 $(CPULIB)               := SONAME = 'cpu/$$CPU'
418 $(DTRACESTUBS)          := SONAME = dtracestubs

420 #
421 #       Installation directories
422 #

424 #
425 #       For now, 64b modules install into a subdirectory
426 #       of their 32b brethren.
427 #
428 SUBDIR64_sparc          = sparcv9
429 SUBDIR64_i386           = amd64
430 SUBDIR64                = $(SUBDIR64_$(MACH))

432 ROOT_MOD_DIR            = $(ROOT)/kernel

434 ROOT_KERN_DIR_32        = $(ROOT_MOD_DIR)
435 ROOT_BRAND_DIR_32       = $(ROOT_MOD_DIR)/brand
436 ROOT_DRV_DIR_32         = $(ROOT_MOD_DIR)/drv
437 ROOT_DTRACE_DIR_32      = $(ROOT_MOD_DIR)/dtrace
438 ROOT_EXEC_DIR_32        = $(ROOT_MOD_DIR)/exec
439 ROOT_FS_DIR_32          = $(ROOT_MOD_DIR)/fs
440 ROOT_SCHED_DIR_32       = $(ROOT_MOD_DIR)/sched
441 ROOT_SOCK_DIR_32        = $(ROOT_MOD_DIR)/socketmod
442 ROOT_STRMOD_DIR_32      = $(ROOT_MOD_DIR)/strmod
443 ROOT_IPP_DIR_32         = $(ROOT_MOD_DIR)/ipp
444 ROOT_SYS_DIR_32         = $(ROOT_MOD_DIR)/sys
445 ROOT_MISC_DIR_32        = $(ROOT_MOD_DIR)/misc
446 ROOT_KGSS_DIR_32        = $(ROOT_MOD_DIR)/misc/kgss
447 ROOT_SCSI_VHCI_DIR_32   = $(ROOT_MOD_DIR)/misc/scsi_vhci
448 ROOT_PMCS_FW_DIR_32     = $(ROOT_MOD_DIR)/misc/pmcs
449 ROOT_QLC_FW_DIR_32      = $(ROOT_MOD_DIR)/misc/qlc
450 ROOT_EMLXS_FW_DIR_32    = $(ROOT_MOD_DIR)/misc/emlxs
451 ROOT_NLMISC_DIR_32      = $(ROOT_MOD_DIR)/misc
452 ROOT_MACH_DIR_32        = $(ROOT_MOD_DIR)/mach
453 ROOT_CPU_DIR_32         = $(ROOT_MOD_DIR)/cpu
454 ROOT_TOD_DIR_32         = $(ROOT_MOD_DIR)/tod
```



```
455 ROOT_FONT_DIR_32        = $(ROOT_MOD_DIR)/fonts
456 ROOT_DACF_DIR_32        = $(ROOT_MOD_DIR)/dacf
457 ROOT_CRYPTO_DIR_32      = $(ROOT_MOD_DIR)/crypto
458 ROOT_MAC_DIR_32         = $(ROOT_MOD_DIR)/mac
459 ROOT_KICONV_DIR_32      = $(ROOT_MOD_DIR)/kiconv

461 ROOT_KERN_DIR_64        = $(ROOT_MOD_DIR)/$(SUBDIR64)
462 ROOT_BRAND_DIR_64       = $(ROOT_MOD_DIR)/brand/$(SUBDIR64)
463 ROOT_DRV_DIR_64         = $(ROOT_MOD_DIR)/drv/$(SUBDIR64)
464 ROOT_DTRACE_DIR_64      = $(ROOT_MOD_DIR)/dtrace/$(SUBDIR64)
465 ROOT_EXEC_DIR_64        = $(ROOT_MOD_DIR)/exec/$(SUBDIR64)
466 ROOT_FS_DIR_64          = $(ROOT_MOD_DIR)/fs/$(SUBDIR64)
467 ROOT_SCHED_DIR_64       = $(ROOT_MOD_DIR)/sched/$(SUBDIR64)
468 ROOT_SOCK_DIR_64        = $(ROOT_MOD_DIR)/socketmod/$(SUBDIR64)
469 ROOT_STRMOD_DIR_64      = $(ROOT_MOD_DIR)/strmod/$(SUBDIR64)
470 ROOT_IPP_DIR_64         = $(ROOT_MOD_DIR)/ipp/$(SUBDIR64)
471 ROOT_SYS_DIR_64         = $(ROOT_MOD_DIR)/sys/$(SUBDIR64)
472 ROOT_MISC_DIR_64        = $(ROOT_MOD_DIR)/misc/$(SUBDIR64)
473 ROOT_KGSS_DIR_64        = $(ROOT_MOD_DIR)/misc/kgss/$(SUBDIR64)
474 ROOT_SCSI_VHCI_DIR_64   = $(ROOT_MOD_DIR)/misc/scsi_vhci/$(SUBDIR64)
475 ROOT_PMCS_FW_DIR_64     = $(ROOT_MOD_DIR)/misc/pmcs/$(SUBDIR64)
476 ROOT_QLC_FW_DIR_64      = $(ROOT_MOD_DIR)/misc/qlc/$(SUBDIR64)
477 ROOT_EMLXS_FW_DIR_64    = $(ROOT_MOD_DIR)/misc/emlxs/$(SUBDIR64)
478 ROOT_NLMISC_DIR_64      = $(ROOT_MOD_DIR)/misc/$(SUBDIR64)
479 ROOT_MACH_DIR_64        = $(ROOT_MOD_DIR)/mach/$(SUBDIR64)
480 ROOT_CPU_DIR_64         = $(ROOT_MOD_DIR)/cpu/$(SUBDIR64)
481 ROOT_TOD_DIR_64         = $(ROOT_MOD_DIR)/tod/$(SUBDIR64)
482 ROOT_FONT_DIR_64        = $(ROOT_MOD_DIR)/fonts/$(SUBDIR64)
483 ROOT_DACF_DIR_64        = $(ROOT_MOD_DIR)/dacf/$(SUBDIR64)
484 ROOT_CRYPTO_DIR_64      = $(ROOT_MOD_DIR)/crypto/$(SUBDIR64)
485 ROOT_MAC_DIR_64         = $(ROOT_MOD_DIR)/mac/$(SUBDIR64)
486 ROOT_KICONV_DIR_64      = $(ROOT_MOD_DIR)/kiconv/$(SUBDIR64)

488 ROOT_KERN_DIR           = $(ROOT_KERN_DIR_$(CLASS))
489 ROOT_BRAND_DIR          = $(ROOT_BRAND_DIR_$(CLASS))
490 ROOT_DRV_DIR            = $(ROOT_DRV_DIR_$(CLASS))
491 ROOT_DTRACE_DIR         = $(ROOT_DTRACE_DIR_$(CLASS))
492 ROOT_EXEC_DIR           = $(ROOT_EXEC_DIR_$(CLASS))
493 ROOT_FS_DIR             = $(ROOT_FS_DIR_$(CLASS))
494 ROOT_SCHED_DIR          = $(ROOT_SCHED_DIR_$(CLASS))
495 ROOT_SOCK_DIR           = $(ROOT_SOCK_DIR_$(CLASS))
496 ROOT_STRMOD_DIR         = $(ROOT_STRMOD_DIR_$(CLASS))
497 ROOT_IPP_DIR            = $(ROOT_IPP_DIR_$(CLASS))
498 ROOT_SYS_DIR            = $(ROOT_SYS_DIR_$(CLASS))
499 ROOT_MISC_DIR           = $(ROOT_MISC_DIR_$(CLASS))
500 ROOT_KGSS_DIR           = $(ROOT_KGSS_DIR_$(CLASS))
501 ROOT_SCSI_VHCI_DIR      = $(ROOT_SCSI_VHCI_DIR_$(CLASS))
502 ROOT_PMCS_FW_DIR        = $(ROOT_PMCS_FW_DIR_$(CLASS))
503 ROOT_QLC_FW_DIR         = $(ROOT_QLC_FW_DIR_$(CLASS))
504 ROOT_EMLXS_FW_DIR       = $(ROOT_EMLXS_FW_DIR_$(CLASS))
505 ROOT_NLMISC_DIR         = $(ROOT_NLMISC_DIR_$(CLASS))
506 ROOT_MACH_DIR           = $(ROOT_MACH_DIR_$(CLASS))
507 ROOT_CPU_DIR            = $(ROOT_CPU_DIR_$(CLASS))
508 ROOT_TOD_DIR            = $(ROOT_TOD_DIR_$(CLASS))
509 ROOT_FONT_DIR           = $(ROOT_FONT_DIR_$(CLASS))
510 ROOT_DACF_DIR           = $(ROOT_DACF_DIR_$(CLASS))
511 ROOT_CRYPTO_DIR         = $(ROOT_CRYPTO_DIR_$(CLASS))
512 ROOT_MAC_DIR            = $(ROOT_MAC_DIR_$(CLASS))
513 ROOT_KICONV_DIR         = $(ROOT_KICONV_DIR_$(CLASS))

515 ROOT_MOD_DIRS_32        = $(ROOT_BRAND_DIR_32) $(ROOT_DRV_DIR_32)
516 ROOT_MOD_DIRS_32        = $(ROOT_BRAND_DIR_32) $(ROOT_DRV_DIR_32)
517 ROOT_MOD_DIRS_32        += $(ROOT_EXEC_DIR_32) $(ROOT_DTRACE_DIR_32)
518 ROOT_MOD_DIRS_32        += $(ROOT_FS_DIR_32) $(ROOT_SCHED_DIR_32)
519 ROOT_MOD_DIRS_32        += $(ROOT_STRMOD_DIR_32) $(ROOT_SYS_DIR_32)
520 ROOT_MOD_DIRS_32        += $(ROOT_IPP_DIR_32) $(ROOT_SOCK_DIR_32)
```

```
ROOT_MOD_DIRS_32        += $(ROOT_MISC_DIR_32) $(ROOT_MACH_DIR_32)
ROOT_MOD_DIRS_32        += $(ROOT_KGSS_DIR_32)
ROOT_MOD_DIRS_32        += $(ROOT_SCSI_VHCI_DIR_32)
ROOT_MOD_DIRS_32        += $(ROOT_PMCS_FW_DIR_32)
ROOT_MOD_DIRS_32        += $(ROOT_QLC_FW_DIR_32)
ROOT_MOD_DIRS_32        += $(ROOT_EMLXS_FW_DIR_32)
ROOT_MOD_DIRS_32        += $(ROOT_CPU_DIR_32) $(ROOT_FONT_DIR_32)
ROOT_MOD_DIRS_32        += $(ROOT_TOD_DIR_32) $(ROOT_DACF_DIR_32)
ROOT_MOD_DIRS_32        += $(ROOT_CRYPTO_DIR_32) $(ROOT_MAC_DIR_32)
ROOT_MOD_DIRS_32        += $(ROOT_KICONV_DIR_32)

USR_MOD_DIR             = $(ROOT)/usr/kernel

USR_DRV_DIR_32          = $(USR_MOD_DIR)/drv
USR_EXEC_DIR_32         = $(USR_MOD_DIR)/exec
USR_FS_DIR_32           = $(USR_MOD_DIR)/fs
USR_SCHED_DIR_32        = $(USR_MOD_DIR)/sched
USR_SOCK_DIR_32         = $(USR_MOD_DIR)/socketmod
USR_STRMOD_DIR_32       = $(USR_MOD_DIR)/strmod
USR_SYS_DIR_32          = $(USR_MOD_DIR)/sys
USR_MISC_DIR_32         = $(USR_MOD_DIR)/misc
USR_DACF_DIR_32         = $(USR_MOD_DIR)/dacf
USR_PCBE_DIR_32         = $(USR_MOD_DIR)/pcbe
USR_DTRACE_DIR_32       = $(USR_MOD_DIR)/dtrace
USR_BRAND_DIR_32        = $(USR_MOD_DIR)/brand

USR_DRV_DIR_64          = $(USR_MOD_DIR)/drv/$(SUBDIR64)
USR_EXEC_DIR_64         = $(USR_MOD_DIR)/exec/$(SUBDIR64)
USR_FS_DIR_64           = $(USR_MOD_DIR)/fs/$(SUBDIR64)
USR_SCHED_DIR_64        = $(USR_MOD_DIR)/sched/$(SUBDIR64)
USR_SOCK_DIR_64         = $(USR_MOD_DIR)/socketmod/$(SUBDIR64)
USR_STRMOD_DIR_64       = $(USR_MOD_DIR)/strmod/$(SUBDIR64)
USR_SYS_DIR_64          = $(USR_MOD_DIR)/sys/$(SUBDIR64)
USR_MISC_DIR_64         = $(USR_MOD_DIR)/misc/$(SUBDIR64)
USR_DACF_DIR_64         = $(USR_MOD_DIR)/dacf/$(SUBDIR64)
USR_PCBE_DIR_64         = $(USR_MOD_DIR)/pcbe/$(SUBDIR64)
USR_DTRACE_DIR_64       = $(USR_MOD_DIR)/dtrace/$(SUBDIR64)
USR_BRAND_DIR_64        = $(USR_MOD_DIR)/brand/$(SUBDIR64)

USR_DRV_DIR             = $(USR_DRV_DIR_$(CLASS))
USR_EXEC_DIR            = $(USR_EXEC_DIR_$(CLASS))
USR_FS_DIR              = $(USR_FS_DIR_$(CLASS))
USR_SCHED_DIR           = $(USR_SCHED_DIR_$(CLASS))
USR_SOCK_DIR            = $(USR_SOCK_DIR_$(CLASS))
USR_STRMOD_DIR          = $(USR_STRMOD_DIR_$(CLASS))
USR_SYS_DIR             = $(USR_SYS_DIR_$(CLASS))
USR_MISC_DIR            = $(USR_MISC_DIR_$(CLASS))
USR_DACF_DIR            = $(USR_DACF_DIR_$(CLASS))
USR_PCBE_DIR            = $(USR_PCBE_DIR_$(CLASS))
USR_DTRACE_DIR          = $(USR_DTRACE_DIR_$(CLASS))
USR_BRAND_DIR           = $(USR_BRAND_DIR_$(CLASS))

USR_MOD_DIRS_32         = $(USR_DRV_DIR_32) $(USR_EXEC_DIR_32)
USR_MOD_DIRS_32         += $(USR_FS_DIR_32) $(USR_SCHED_DIR_32)
USR_MOD_DIRS_32         += $(USR_STRMOD_DIR_32) $(USR_SYS_DIR_32)
USR_MOD_DIRS_32         += $(USR_MISC_DIR_32) $(USR_DACF_DIR_32)
USR_MOD_DIRS_32         += $(USR_PCBE_DIR_32)
USR_MOD_DIRS_32         += $(USR_DTRACE_DIR_32) $(USR_BRAND_DIR_32)
USR_MOD_DIRS_32         += $(USR_SOCK_DIR_32)

#
#
#
include $(SRC)/Makefile.psm

#
```



```
#       The "-r" on the remove may be considered temporary, but is required
#       while the replacement of the SUNW,SPARCstation-10,SX directory by
#       a symbolic link is being propagated.
#
# IMPORTANT:: if you change any of these INS.mumble rules, then you MUST also
# change the corresponding override definitions in $CLOSED/Makefile.tonic.
# If you do not do this, then the closedbins build for the OpenSolaris
# community will break. PS, the gatekeepers will be upset too.
#
INS.slink1= $(RM) -r $@; $(SYMLINK) $(PLATFORM) $@
INS.slink2= $(RM) -r $@; $(SYMLINK) ../$(PLATFORM)/$(@F) $@
INS.slink3= $(RM) -r $@; $(SYMLINK) $(IMPLEMENTED_PLATFORM) $@
INS.slink4= $(RM) -r $@; $(SYMLINK) ../$(PLATFORM)/include $@
INS.slink5= $(RM) -r $@; $(SYMLINK) ../$(PLATFORM)/sbin $@
INS.slink6= $(RM) -r $@; $(SYMLINK) ../../$(PLATFORM)/lib/$(MODULE) $@
INS.slink7= $(RM) -r $@; $(SYMLINK) ../../$(PLATFORM)/sbin/$(@F) $@

ROOT_PLAT_LINKS         = $(PLAT_LINKS:%=$(ROOT_PLAT_DIR)/%)
ROOT_PLAT_LINKS_2       = $(PLAT_LINKS_2:%=$(ROOT_PLAT_DIR)/%)
USR_PLAT_LINKS          = $(PLAT_LINKS:%=$(USR_PLAT_DIR)/%)
USR_PLAT_LINKS_2        = $(PLAT_LINKS_2:%=$(USR_PLAT_DIR)/%)

#
# Collection of all relevant, delivered kernel modules.
#
# Note that we insist on building genunix first, because everything else
# uniquifies against it.  When doing a 'make' from usr/src/uts/, we'll enter
# the platform directories first.  These will cd into the corresponding genunix
# directory and build it.  So genunix /shouldn't/ get rebuilt when we get to
# building all the kernel modules.  However, due to an as-yet-unexplained
# problem with dependencies, sometimes it does get rebuilt, which then messes
# up the other modules.  So we always force the issue here rather than try to
# build genunix in parallel with everything else.
#
PARALLEL_KMODS = $(DRV_KMODS) $(EXEC_KMODS) $(FS_KMODS) $(SCHED_KMODS) \
                 $(TOD_KMODS) $(STRMOD_KMODS) $(SYS_KMODS) $(MISC_KMODS) \
                 $(NLMISC_KMODS) $(MACH_KMODS) $(CPU_KMODS) $(GSS_KMODS) \
                 $(MMU_KMODS) $(DACF_KMODS) $(EXPORT_KMODS) $(IPP_KMODS) \
                 $(CRYPTO_KMODS) $(PCBE_KMODS) \
                 $(DRV_KMODS_$(CLASS)) $(MISC_KMODS_$(CLASS)) $(MAC_KMODS) \
                 $(BRAND_KMODS) $(KICONV_KMODS) \
                 $(SOCKET_KMODS)

KMODS = $(GENUNIX_KMODS) $(PARALLEL_KMODS)

$(PARALLEL_KMODS): $(GENUNIX_KMODS)

$(CLOSED_BUILD)CLOSED_KMODS = $(CLOSED_DRV_KMODS) $(CLOSED_TOD_KMODS) \
        $(CLOSED_MISC_KMODS) $(CLOSED_CPU_KMODS) \
        $(CLOSED_NLMISC_KMODS) $(CLOSED_DRV_KMODS_$(CLASS))

LINT_KMODS = $(DRV_KMODS) $(EXEC_KMODS) $(FS_KMODS) $(SCHED_KMODS) \
          $(TOD_KMODS) $(STRMOD_KMODS) $(SYS_KMODS) $(MISC_KMODS) \
          $(MACH_KMODS) $(GSS_KMODS) $(DACF_KMODS) $(IPP_KMODS) \
          $(CRYPTO_KMODS) $(PCBE_KMODS) \
          $(DRV_KMODS_$(CLASS)) $(MISC_KMODS_$(CLASS)) $(MAC_KMODS) \
          $(BRAND_KMODS) $(KICONV_KMODS) $(SOCKET_KMODS)

$(CLOSED_BUILD)CLOSED_LINT_KMODS = $(CLOSED_DRV_KMODS) $(CLOSED_TOD_KMODS) \
          $(CLOSED_MISC_KMODS) $(CLOSED_DRV_KMODS_$(CLASS))

THIS_YEAR:sh=   /bin/date +%Y
$(OBJS_DIR)/logsubr.o   := CPPFLAGS += -DTHIS_YEAR=$(THIS_YEAR)
$(OBJS_DIR)/logsubr.ln  := CPPFLAGS += -DTHIS_YEAR=$(THIS_YEAR)

#
```

```
653 #       Files to be compiled with -xa, to generate basic block execution
654 #       count data.
655 #
656 #       There are several ways to compile parts of the kernel for kcov:
657 #               1)  Add targets to BB_FILES here or in other Makefiles
658 #                       (they must in the form of $(OBJS_DIR)/target.o)
659 #               2)  setenv BB_FILES '$(XXX_OBJS:%=$(OBJS_DIR)/%)'
660 #               3)  setenv BB_FILES '$(OBJECTS)'
661 #
662 #       Do NOT setenv CFLAGS -xa, as that will cause infinite recursion
663 #       in unix_bb.o
664 #
665 BB_FILES =
666 $(BB_FILES)     := XAOPT = -xa

668 #
669 #       The idea here is for unix_bb.o to be in all kernels except the
670 #       kernel which actually gets shipped to customers.  In practice,
671 #       $(RELEASE_BUILD) is on for a number of the late beta and fcs builds.
672 #
673 CODE_COVERAGE=
674 $(RELEASE_BUILD)CODE_COVERAGE:sh=       echo \\043
675 $(CODE_COVERAGE)$(OBJS_DIR)/unix_bb.o   := CPPFLAGS     += -DKCOV
676 $(CODE_COVERAGE)$(OBJS_DIR)/unix_bb.ln  := CPPFLAGS     += -DKCOV

678 #
679 #       Do not let unix_bb.o get compiled with -xa!
680 #
681 $(OBJS_DIR)/unix_bb.o   := XAOPT =

683 #
684 # Privilege files
685 #
686 PRIVS_AWK = $(SRC)/uts/common/os/privs.awk
687 PRIVS_DEF = $(SRC)/uts/common/os/priv_defs

689 #
690 # USB device data
691 #
692 USBDEVS_AWK =   $(SRC)/uts/common/io/usb/usbdevs2h.awk
693 USBDEVS_DATA =  $(SRC)/uts/common/io/usb/usbdevs
```

```
**********************************************************
   13730 Sun Jul 28 20:29:08 2013
new/usr/src/uts/sparc/Makefile.sparc.shared
3735 should include an empty make variable in the default CFLAGS/CCFLAGS
3844 the build should make source-level debugging easier
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Garrett D'Amore <garrett@damore.org>
**********************************************************
```

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#

# Copyright (c) 2005, 2010, Oracle and/or its affiliates. All rights reserved.


#
#       This makefile contains the common definitions for all sparc
#       implementation architecture independent modules.
#


#
#       Define supported builds
#
DEF_BUILDS      = $(DEF_BUILDS64)
ALL_BUILDS      = $(ALL_BUILDS64)

#
#       Everybody needs to know how to build modstubs.o and to locate unix.o.
#       Note that unix.o must currently be selected from among the possible
#       "implementation architectures". Note further, that unix.o is only
#       used as an optional error check for undefines so (theoretically)
#       any "implementation architectures" could be used. We choose sun4u
#       because it is the reference port.
#
UNIX_DIR         = $(UTSBASE)/sun4u/unix
GENLIB_DIR       = $(UTSBASE)/sun4u/genunix
IPDRV_DIR        = $(UTSBASE)/sparc/ip
MODSTUBS_DIR     = $(UNIX_DIR)
DSF_DIR          = $(UNIX_DIR)
LINTS_DIR        = $(OBJS_DIR)
LINT_LIB_DIR     = $(UTSBASE)/sparc/lint-libs/$(OBJS_DIR)

UNIX_O           = $(UNIX_DIR)/$(OBJS_DIR)/unix.o
MODSTUBS_O       = $(MODSTUBS_DIR)/$(OBJS_DIR)/modstubs.o
GENLIB           = $(UTSBASE)/sun4u/lint-libs/$(OBJS_DIR)/libgenunix.so

LINT_LIB_32      = $(UTSBASE)/sun4u/lint-libs/$(OBJS_DIR)/llib-lunix.ln
GEN_LINT_LIB_32  = $(UTSBASE)/sun4u/lint-libs/$(OBJS_DIR)/llib-lgenunix.ln
```



```
LINT_LIB_64      = $(UTSBASE)/sun4u/lint-libs/$(OBJS_DIR)/llib-lunix.ln
GEN_LINT_LIB_64  = $(UTSBASE)/sun4u/lint-libs/$(OBJS_DIR)/llib-lgenunix.ln

LINT_LIB         = $(LINT_LIB_$(CLASS))
GEN_LINT_LIB     = $(GEN_LINT_LIB_$(CLASS))

LINT32_DIRS      = $(LINT32_BUILDS:%=$(UTSBASE)/sparc/lint-libs/%)
LINT32_FILES     = $(LINT32_DIRS:%=%/llib-l$(MODULE).ln)

LINT64_DIRS      = $(LINT64_BUILDS:%=$(UTSBASE)/sparc/lint-libs/%)
LINT64_FILES     = $(LINT64_DIRS:%=%/llib-l$(MODULE).ln)

#
#       Include the makefiles which define build rule templates, the
#       collection of files per module, and a few specific flags. Note
#       that order is significant, just as with an include path. The
#       first build rule template which matches the files name will be
#       used. By including these in order from most machine dependent
#       to most machine independent, we allow a machine dependent file
#       to be used in preference over a machine independent version
#       (Such as a machine specific optimization, which preserves the
#       interfaces.)
#
include $(UTSBASE)/sparc/Makefile.files
include $(UTSBASE)/sparc/v9/Makefile.files
include $(UTSTREE)/sun/Makefile.files
include $(UTSTREE)/common/Makefile.files

#
# ----- TRANSITIONAL SECTION --------------------------------------------------
#

#
#       Not everything which *should* be a module is a module yet. The
#       following is a list of such objects which are currently part of
#       genunix but which might someday become kmods.   This must be
#       defined before we include Makefile.uts, or else genunix's build
#       won't be as parallel as we might like.
#
NOT_YET_KMODS    = $(OLDPTY_OBJS) $(PTY_OBJS) $(VCONS_CONF_OBJS) $(MOD_OBJS)

#
# ----- END OF TRANSITIONAL SECTION -------------------------------------------
#
#       Include machine independent rules. Note that this does not imply
#       that the resulting module from rules in Makefile.uts is machine
#       independent. Only that the build rules are machine independent.
#
include $(UTSBASE)/Makefile.uts

#
#       machine specific optimization, override default in Makefile.master
#
XARCH_32         = -xarch=v8
XARCH_64         = -m64
XARCH            = $(XARCH_$(CLASS))

COPTIMIZE_32     = -xO3
COPTIMIZE_64     = -xO3
COPTIMIZE        = $(COPTIMIZE_$(CLASS))

CCMODE           = -Xa

CFLAGS_32        = -xcg92
CFLAGS_64        = -xchip=ultra $(CCABS32) $(CCREGSYM)
CFLAGS           = $(CFLAGS_$(CLASS))
```

```
126 CFLAGS          += $(XARCH)
127 CFLAGS          += $(COPTIMIZE)
128 CFLAGS          += $(EXTRA_CFLAGS)
129 CFLAGS          += $(XAOPT)
130 CFLAGS          += $(INLINES) -D_ASM_INLINES
131 CFLAGS          += $(CCMODE)
132 CFLAGS          += $(SPACEFLAG)
133 CFLAGS          += $(CERRWARN)
134 CFLAGS          += $(CTF_FLAGS_$(CLASS))
135 CFLAGS          += $(C99MODE)
136 CFLAGS          += $(CCUNBOUND)
137 CFLAGS          += $(CCSTATICSYM)
138 CFLAGS          += $(CC32BITCALLERS)
139 CFLAGS          += $(CCNOAUTOINLINE)
140 CFLAGS          += $(IROPTFLAG)
141 CFLAGS          += $(CGLOBALSTATIC)
142 CFLAGS          += -xregs=no%float
143 CFLAGS          += -xstrconst
144 CFLAGS          += $(CSOURCEDEBUGFLAGS)
145 CFLAGS          += $(CUSERFLAGS)
146 #endif /* ! codereview */

148 ASFLAGS         += $(XARCH)

150 LINT_DEFS_32    =
151 LINT_DEFS_64    = -m64
152 LINT_DEFS       += $(LINT_DEFS_$(CLASS))

154 #
155 #       The following must be defined for all implementations:
156 #
157 #       MODSTUBS:       Module stubs source file.
158 #
159 MODSTUBS         = $(UTSBASE)/sparc/ml/modstubs.s

161 #
162 #       Define the actual specific platforms - obviously none.
163 #
164 MACHINE_DEFS     =

166 #
167 #       Debugging level
168 #
169 #       Special knowledge of which special debugging options effect which
170 #       file is used to optimize the build if these flags are changed.
171 #
172 #       XXX: The above could possibly be done for more flags and files, but
173 #            is left as an experiment to the interested reader. Be forewarned,
174 #            that excessive use could lead to maintenance difficulties.
175 #
176 DEBUG_DEFS_OBJ32        =
177 DEBUG_DEFS_DBG32        = -DDEBUG
178 DEBUG_DEFS_OBJ64        =
179 DEBUG_DEFS_DBG64        = -DDEBUG
180 DEBUG_DEFS              = $(DEBUG_DEFS_$(BUILD_TYPE))

182 DEBUG_COND_OBJ32        :sh = echo \\043
183 DEBUG_COND_DBG32        =
184 DEBUG_COND_OBJ64        :sh = echo \\043
185 DEBUG_COND_DBG64        =
186 IF_DEBUG_OBJ            = $(DEBUG_COND_$(BUILD_TYPE))$(OBJS_DIR)/

188 $(IF_DEBUG_OBJ)syscall.o        :=      DEBUG_DEFS      += -DSYSCALLTRACE
189 $(IF_DEBUG_OBJ)clock.o          :=      DEBUG_DEFS      += -DKSLICE=1
```



```
191 # Comment these out if you don't want dispatcher lock statistics.

193 # $(IF_DEBUG_OBJ)disp_lock.o    := DEBUG_DEFS   += -DDISP_LOCK_STATS

195 #
196 #       Collect the preprocessor definitions to be associated with *all*
197 #       files.
198 #
199 ALL_DEFS         = $(MACHINE_DEFS) $(DEBUG_DEFS) $(OPTION_DEFS)
200 #
201 #
202 #       The kernels modules which are "implementation architecture"
203 #       specific for this machine are enumerated below. Note that most
204 #       of these modules must exist (in one form or another) for each
205 #       architecture.
206 #
207 #       Common Drivers (usually pseudo drivers) (/kernel/drv):
208 #
209 DRV_KMODS       += aggr arp audio bl blkdev bofi clone cn conskbd consms cpuid
210 DRV_KMODS       += crypto cryptoadm devinfo dump
211 DRV_KMODS       += dtrace fasttrap fbt lockstat profile sdt systrace dcpc
212 DRV_KMODS       += fssnap icmp icmp6 ip ip6 ipnet ipsecah
213 DRV_KMODS       += ipsecesp iptun iwscn keysock kmdb kstat ksyms llc1
214 DRV_KMODS       += lofi
215 DRV_KMODS       += log logindmux kssl mm nca physmem pm poll pool
216 DRV_KMODS       += pseudo ptc ptm pts ptsl ramdisk random rsm rts sad
217 DRV_KMODS       += simnet softmac sppp sppptun sy sysevent sysmsg
218 DRV_KMODS       += spdsock
219 DRV_KMODS       += tcp tcp6 tl tnf ttymux udp udp6 wc winlock zcons
220 DRV_KMODS       += ippctl
221 DRV_KMODS       += dld
222 DRV_KMODS       += ipf
223 DRV_KMODS       += rpcib
224 DRV_KMODS       += dlpistub
225 DRV_KMODS       += vnic
226 DRV_KMODS       += xge
227 DRV_KMODS       += rds
228 DRV_KMODS       += rdsv3
229 DRV_KMODS       += chxge
230 DRV_KMODS       += smbsrv
231 DRV_KMODS       += vscan
232 DRV_KMODS       += nsmb
233 DRV_KMODS       += fm
234 DRV_KMODS       += nulldriver
235 DRV_KMODS       += bridge trill
236 DRV_KMODS       += bpf
237 DRV_KMODS       += dca

239 $(CLOSED_BUILD)CLOSED_DRV_KMODS += glm
240 $(CLOSED_BUILD)CLOSED_DRV_KMODS += isp
241 $(CLOSED_BUILD)CLOSED_DRV_KMODS += mpt
242 $(CLOSED_BUILD)CLOSED_DRV_KMODS += qus
243 $(CLOSED_BUILD)CLOSED_DRV_KMODS += se

245 #
246 #       Hardware Drivers in common space
247 #

249 DRV_KMODS       += afe
250 DRV_KMODS       += audio1575
251 DRV_KMODS       += audioens
252 DRV_KMODS       += audiols
253 DRV_KMODS       += audiop16x
254 DRV_KMODS       += audiopci
255 DRV_KMODS       += audiots
256 DRV_KMODS       += e1000g
```

```
DRV_KMODS       += efe
DRV_KMODS       += hxge
DRV_KMODS       += mxfe
DRV_KMODS       += pcan
DRV_KMODS       += pcwl
DRV_KMODS       += rge
DRV_KMODS       += rtls
DRV_KMODS       += sfe
DRV_KMODS       += aac
DRV_KMODS       += igb
DRV_KMODS       += ixgbe
DRV_KMODS       += vr
DRV_KMODS       += mr_sas
$(CLOSED_BUILD)CLOSED_DRV_KMODS += ixgb
DRV_KMODS       += yge

#
#       Machine Specific Driver Modules (/kernel/drv):
#
DRV_KMODS       += audiocs
DRV_KMODS       += bge dmfe eri fas hme qfe
DRV_KMODS       += openeepr options sd ses st
DRV_KMODS       += ssd
DRV_KMODS       += ecpp
DRV_KMODS       += hid hubd ehci ohci uhci usb_mid usb_ia scsa2usb usbprn ugen
DRV_KMODS       += usbser usbsacm usbsksp usbsprl
DRV_KMODS       += usb_as usb_ac
DRV_KMODS       += usbskel
DRV_KMODS       += usbvc
DRV_KMODS       += usbftdi
DRV_KMODS       += wusb_df hwahc hwarc wusb_ca
DRV_KMODS       += usbecm
DRV_KMODS       += hci1394 av1394 scsa1394 dcam1394
DRV_KMODS       += sbp2
DRV_KMODS       += ib ibp eibnx eoib rdsib sdp iser daplt hermon tavor sol_ucma
DRV_KMODS       += sol_umad
DRV_KMODS       += pci_pci pcieb pcieb_bcm
DRV_KMODS       += i8042 kb8042 mouse8042
DRV_KMODS       += fcode
DRV_KMODS       += mpt_sas
DRV_KMODS       += socal
DRV_KMODS       += sgen
DRV_KMODS       += myri10ge
DRV_KMODS       += smp
DRV_KMODS       += dad
DRV_KMODS       += scsi_vhci
DRV_KMODS       += fcp
DRV_KMODS       += fcip
DRV_KMODS       += fcsm
DRV_KMODS       += fp
DRV_KMODS       += qlc
DRV_KMODS       += qlge
DRV_KMODS       += stmf
DRV_KMODS       += stmf_sbd
DRV_KMODS       += fct
DRV_KMODS       += fcoe
DRV_KMODS       += fcoet
DRV_KMODS       += fcoei
DRV_KMODS       += qlt
DRV_KMODS       += iscsit
DRV_KMODS       += pppt
DRV_KMODS       += ncall nsctl sdbc nskern sv
DRV_KMODS       += ii rdc rdcsrv rdcstub
DRV_KMODS       += iscsi
DRV_KMODS       += emlxs
DRV_KMODS       += oce
```



```
DRV_KMODS       += srpt
DRV_KMODS       += pmcs
DRV_KMODS       += pmcs8001fw

$(CLOSED_BUILD)CLOSED_DRV_KMODS += ifp
$(CLOSED_BUILD)CLOSED_DRV_KMODS += uata
$(CLOSED_BUILD)CLOSED_DRV_KMODS += usbser_edge

#
#       I/O framework test drivers
#
DRV_KMODS       += pshot
DRV_KMODS       += gen_drv
DRV_KMODS       += tvhci tphci tclient
DRV_KMODS       += emul64

#
# PCMCIA specific module(s)
#
DRV_KMODS       += pcs
MISC_KMODS      += busra cardbus dada pcmcia
DRV_KMODS       += pcata
DRV_KMODS       += pcic

$(CLOSED_BUILD)CLOSED_DRV_KMODS += pcser

# Add lvm
#
DRV_KMODS       += md
MISC_KMODS      += md_mirror md_stripe md_hotspares md_raid md_trans md_notify
MISC_KMODS      += md_sp

#
#       Exec Class Modules (/kernel/exec):
#
EXEC_KMODS      += aoutexec elfexec intpexec shbinexec javaexec

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

#
#       File System Modules (/kernel/fs):
#
FS_KMODS        += dev devfs fdfs fifofs hsfs lofs namefs nfs pcfs tmpfs zfs
FS_KMODS        += zut specfs udfs ufs autofs cachefs procfs sockfs mntfs
FS_KMODS        += ctfs objfs sharefs dcfs smbfs

#
#       Streams Modules (/kernel/strmod):
#
STRMOD_KMODS    += bufmod connld dedump ldterm ms pckt pfmod
STRMOD_KMODS    += pipemod ptem redirmod rpcmod rlmod telmod timod
STRMOD_KMODS    += spppasyn spppcomp
STRMOD_KMODS    += tirdwr ttcompat
STRMOD_KMODS    += usbkbm usbms usbwcm usb_ah
STRMOD_KMODS    += drcompat
STRMOD_KMODS    += cryptmod
STRMOD_KMODS    += vuid3ps2

#
#       'System' Modules (/kernel/sys):
#
SYS_KMODS       += c2audit
SYS_KMODS       += exacctsys
```

```
389 SYS_KMODS       += inst_sync kaio msgsys semsys shmsys sysacct pipe
390 SYS_KMODS       += doorfs pset acctctl portfs

392 #
393 #       'User' Modules (/kernel/misc):
394 #
395 MISC_KMODS      += ac97
396 MISC_KMODS      += bignum
397 MISC_KMODS      += consconfig gld ipc nfs_dlboot nfssrv scsi
398 MISC_KMODS      += strplumb swapgeneric tlimod
399 MISC_KMODS      += rpcsec rpcsec_gss kgssapi kmech_dummy
400 MISC_KMODS      += kmech_krb5
401 MISC_KMODS      += fssnap_if
402 MISC_KMODS      += hidparser kbtrans usba usba10 usbs49_fw
403 MISC_KMODS      += s1394
404 MISC_KMODS      += hpcsvc pcihp
405 MISC_KMODS      += rsmops
406 MISC_KMODS      += kcf
407 MISC_KMODS      += ksocket
408 MISC_KMODS      += ibcm
409 MISC_KMODS      += ibdm
410 MISC_KMODS      += ibdma
411 MISC_KMODS      += ibmf
412 MISC_KMODS      += ibtl
413 MISC_KMODS      += sol_ofs
414 MISC_KMODS      += idm
415 MISC_KMODS      += idmap
416 MISC_KMODS      += hook
417 MISC_KMODS      += neti
418 MISC_KMODS      += ctf
419 MISC_KMODS      += mac dls
420 MISC_KMODS      += cmlb
421 MISC_KMODS      += tem
422 MISC_KMODS      += pcicfg fcodem fcpci
423 MISC_KMODS      += scsi_vhci_f_sym scsi_vhci_f_tpgs scsi_vhci_f_asym_sun
424 MISC_KMODS      += scsi_vhci_f_sym_hds
425 MISC_KMODS      += scsi_vhci_f_tape scsi_vhci_f_tpgs_tape
426 MISC_KMODS      += fctl
427 MISC_KMODS      += emlxs_fw
428 MISC_KMODS      += qlc_fw_2200
429 MISC_KMODS      += qlc_fw_2300
430 MISC_KMODS      += qlc_fw_2400
431 MISC_KMODS      += qlc_fw_2500
432 MISC_KMODS      += qlc_fw_6322
433 MISC_KMODS      += qlc_fw_8100
434 MISC_KMODS      += spuni
435 MISC_KMODS      += hwa1480_fw uwba
436 MISC_KMODS      += mii

438 $(CLOSED_BUILD)CLOSED_MISC_KMODS       += klmmod klmops
439 $(CLOSED_BUILD)CLOSED_MISC_KMODS       += scsi_vhci_f_asym_lsi
440 $(CLOSED_BUILD)CLOSED_MISC_KMODS       += scsi_vhci_f_asym_emc
441 $(CLOSED_BUILD)CLOSED_MISC_KMODS       += scsi_vhci_f_sym_emc

443 #
444 #       Software Cryptographic Providers (/kernel/crypto):
445 #
446 CRYPTO_KMODS    += aes
447 CRYPTO_KMODS    += arcfour
448 CRYPTO_KMODS    += blowfish
449 CRYPTO_KMODS    += des
450 CRYPTO_KMODS    += md4
451 CRYPTO_KMODS    += md5
452 CRYPTO_KMODS    += ecc
453 CRYPTO_KMODS    += rsa
454 CRYPTO_KMODS    += sha1
```



```
455 CRYPTO_KMODS    += sha2
456 CRYPTO_KMODS    += swrand

458 #
459 # IP Policy Modules (/kernel/ipp):
460 #
461 IPP_KMODS       += dlcosmk
462 IPP_KMODS       += flowacct
463 IPP_KMODS       += ipgpc
464 IPP_KMODS       += dscpmk
465 IPP_KMODS       += tokenmt
466 IPP_KMODS       += tswtclmt

468 #
469 # 'Dacf' modules (/kernel/dacf)
470 DACF_KMODS      += consconfig_dacf

472 #
473 #       SVVS Testing Modules (/kernel/strmod):
474 #
475 #       These are streams and driver modules which are not to be
476 #       delivered with a released system. However, during development
477 #       it is convenient to build and install the SVVS kernel modules.
478 #
479 SVVS_KMODS      += lmodb lmode lmodr lmodt svvslo tidg tivc tmux

481 $(CLOSED_BUILD)SVVS             += svvs

483 #
484 #       Modules eXcluded from the product:
485 #
486 XMODS           +=
487 $(CLOSED_BUILD)CLOSED_XMODS =           \
488         sdpib           \
489         wsdrv

491 #
492 #       'Dacf' Modules (/kernel/dacf):
493 #
494 DACF_KMODS      += net_dacf

496 #
497 #       MAC-Type Plugin Modules (/kernel/mac)
498 #
499 MAC_KMODS       += mac_6to4
500 MAC_KMODS       += mac_ether
501 MAC_KMODS       += mac_ipv4
502 MAC_KMODS       += mac_ipv6
503 MAC_KMODS       += mac_wifi
504 MAC_KMODS       += mac_ib

506 #
507 # socketmod (kernel/socketmod)
508 #
509 SOCKET_KMODS    += sockpfp
510 SOCKET_KMODS    += socksctp
511 SOCKET_KMODS    += socksdp
512 SOCKET_KMODS    += sockrds
513 SOCKET_KMODS    += ksslf

515 #
516 #       kiconv modules (/kernel/kiconv):
517 #
518 KICONV_KMODS    += kiconv_emea kiconv_ja kiconv_ko kiconv_sc kiconv_tc
```

```
**********************************************************
   13770 Sun Jul 28 20:29:09 2013
new/usr/src/uts/sun4u/Makefile.sun4u.shared
3735 should include an empty make variable in the default CFLAGS/CCFLAGS
3844 the build should make source-level debugging easier
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Garrett D'Amore <garrett@damore.org>
**********************************************************
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#

#
# Copyright 2010 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#
#       This makefile contains the common definitions for the sun4u unix
#       and all sun4u implementation architecture dependent modules.
#


#
#       Machine type (implementation architecture):
#
PLATFORM         = sun4u
PROMIF           = ieee1275
PSMBASE          = $(UTSBASE)/../psm

#
#       uname -m value
#
UNAME_M          = $(PLATFORM)

#
# Definitions for the platform-specific /platform directories.
#
# PLATFORMS designates those sun4u machines which have no platform
# specific code.
#
# IMPLEMENTATIONS is used to designate sun4u machines which do have
# platform specific modules (perhaps including their own unix).  All
# code specific to a given implementation resides in the appropriately
# named subdirectory.   This requires these platforms to have their
# own Makefiles to define ROOT_PLAT_DIRS, USR_PLAT_DIRS, etc.
#
# So if we had an implementation named 'foo', we would need the following
# Makefiles in the foo subdirectory:
#
#       sun4u/foo/Makefile
#       sun4u/foo/Makefile.foo
```



```
#       sun4u/foo/Makefile.targ
#


#
# /usr/platform/$(IMPLEMENTED_PLATFORM) is created as a directory that
# all the $(LINKED_PLATFORMS) link to.
#
IMPLEMENTED_PLATFORM    = SUNW,Ultra-2

LINKED_PLATFORMS        += SUNW,Ultra-30
LINKED_PLATFORMS        += SUNW,Ultra-60

#
# all PLATFORMS that do not belong in the $(IMPLEMENTATIONS) list
# ie. all desktop platforms
#
PLATFORMS                = $(IMPLEMENTED_PLATFORM)
PLATFORMS               += $(LINKED_PLATFORMS)

ROOT_PLAT_DIRS          = $(PLATFORMS:%=$(ROOT_PLAT_DIR)/%)
USR_PLAT_DIRS           = $(PLATFORMS:%=$(USR_PLAT_DIR)/%)

USR_DESKTOP_DIR         = $(USR_PLAT_DIR)/$(IMPLEMENTED_PLATFORM)
USR_DESKTOP_INC_DIR     = $(USR_DESKTOP_DIR)/include
USR_DESKTOP_SBIN_DIR    = $(USR_DESKTOP_DIR)/sbin
USR_DESKTOP_LIB_DIR     = $(USR_DESKTOP_DIR)/lib

#
# Welcome to SPARC V9.
#


#
#       Define supported builds
#
DEF_BUILDS      = $(DEF_BUILDS64)
ALL_BUILDS      = $(ALL_BUILDS64)

#
#       Everybody needs to know how to build modstubs.o and to locate unix.o
#
UNIX_DIR         = $(UTSBASE)/$(PLATFORM)/unix
GENLIB_DIR       = $(UTSBASE)/$(PLATFORM)/genunix
MODSTUBS_DIR     = $(UNIX_DIR)
DSF_DIR          = $(UTSBASE)/$(PLATFORM)/genassym
LINTS_DIR        = $(OBJS_DIR)
LINT_LIB_DIR     = $(UTSBASE)/$(PLATFORM)/lint-libs/$(OBJS_DIR)

DTRACESTUBS_O    = $(OBJS_DIR)/dtracestubs.o
DTRACESTUBS      = $(OBJS_DIR)/libdtracestubs.so

UNIX_O           = $(UNIX_DIR)/$(OBJS_DIR)/unix.o
MODSTUBS_O       = $(MODSTUBS_DIR)/$(OBJS_DIR)/modstubs.o
GENLIB           = $(GENLIB_DIR)/$(OBJS_DIR)/libgenunix.so

LINT_LIB         = $(LINT_LIB_DIR)/llib-lunix.ln
GEN_LINT_LIB     = $(LINT_LIB_DIR)/llib-lgenunix.ln

LINT64_DIRS      = $(LINT64_BUILDS:%=$(UTSBASE)/$(PLATFORM)/lint-libs/%)
LINT64_FILES     = $(LINT64_DIRS:%=%/llib-l$(MODULE).ln)

#
#       cpu and platform modules need to know how to build their own symcheck mo
#
PLATMOD          = platmod
PLATLIB          = $(PLAT_DIR)/$(OBJS_DIR)/libplatmod.so
```

```
125 CPUNAME          = cpu
126 CPULIB           = $(CPU_DIR)/$(OBJS_DIR)/libcpu.so

128 SYM_MOD          = $(OBJS_DIR)/unix.sym

130 #
131 #       Include the makefiles which define build rule templates, the
132 #       collection of files per module, and a few specific flags. Note
133 #       that order is significant, just as with an include path. The
134 #       first build rule template which matches the files name will be
135 #       used. By including these in order from most machine dependent
136 #       to most machine independent, we allow a machine dependent file
137 #       to be used in preference over a machine independent version
138 #       (Such as a machine specific optimization, which preserves the
139 #       interfaces.)
140 #
141 include $(UTSBASE)/sun4/Makefile.files
142 include $(UTSTREE)/$(PLATFORM)/Makefile.files
143 include $(UTSBASE)/sfmmu/Makefile.files
144 include $(UTSBASE)/sparc/v9/Makefile.files
145 include $(UTSBASE)/sparc/Makefile.files
146 include $(UTSTREE)/sun/Makefile.files
147 include $(SRC)/psm/promif/$(PROMIF)/common/Makefile.files
148 include $(SRC)/psm/promif/$(PROMIF)/$(PLATFORM)/Makefile.files
149 include $(UTSTREE)/common/Makefile.files

151 #
152 #       Include machine independent rules. Note that this does not imply
153 #       that the resulting module from rules in Makefile.uts is machine
154 #       independent. Only that the build rules are machine independent.
155 #
156 include $(UTSBASE)/Makefile.uts

158 # These come after Makefile.uts (for CLOSED_BUILD).
159 IMPLEMENTATIONS          = tazmo
160 IMPLEMENTATIONS         += starfire
161 IMPLEMENTATIONS         += javelin
162 IMPLEMENTATIONS         += darwin
163 IMPLEMENTATIONS         += quasar
164 IMPLEMENTATIONS         += grover
165 IMPLEMENTATIONS         += enchilada
166 IMPLEMENTATIONS         += taco
167 IMPLEMENTATIONS         += mpxu
168 IMPLEMENTATIONS         += excalibur
169 IMPLEMENTATIONS         += montecarlo
170 IMPLEMENTATIONS         += serengeti
171 IMPLEMENTATIONS         += littleneck
172 IMPLEMENTATIONS         += starcat
173 IMPLEMENTATIONS         += daktari
174 IMPLEMENTATIONS         += cherrystone
175 IMPLEMENTATIONS         += fjlite
176 IMPLEMENTATIONS         += snowbird
177 IMPLEMENTATIONS         += schumacher
178 IMPLEMENTATIONS         += blade
179 IMPLEMENTATIONS         += boston
180 IMPLEMENTATIONS         += seattle
181 IMPLEMENTATIONS         += chicago
182 IMPLEMENTATIONS         += sunfire
183 IMPLEMENTATIONS         += lw8
184 IMPLEMENTATIONS         += makaha
185 IMPLEMENTATIONS         += opl
186 IMPLEMENTATIONS         += lw2plus

188 $(CLOSED_BUILD)CLOSED_IMPLEMENTATIONS   = chalupa
189 $(CLOSED_BUILD)CLOSED_IMPLEMENTATIONS   += ents
```



```
191 #
192 #       machine specific optimization, override default in Makefile.master
193 #
194 CC_XARCH        = -m64 -xarch=sparcvis
195 AS_XARCH        = -xarch=v9a
196 COPTIMIZE       = -xO3
197 CCMODE          = -Xa

199 CFLAGS          = -xchip=ultra $(CCABS32) $(CCREGSYM)
200 CFLAGS          += $(CC_XARCH)
201 CFLAGS          += $(COPTIMIZE)
202 CFLAGS          += $(EXTRA_CFLAGS)
203 CFLAGS          += $(XAOPT)
204 CFLAGS          += $(INLINES) -D_ASM_INLINES
205 CFLAGS          += $(CCMODE)
206 CFLAGS          += $(SPACEFLAG)
207 CFLAGS          += $(CERRWARN)
208 CFLAGS          += $(CTF_FLAGS_$(CLASS))
209 CFLAGS          += $(C99MODE)
210 CFLAGS          += $(CCUNBOUND)
211 CFLAGS          += $(CCNOAUTOINLINE)
212 CFLAGS          += $(CCSTATICSYM)
213 CFLAGS          += $(CC32BITCALLERS)
214 CFLAGS          += $(IROPTFLAG)
215 CFLAGS          += $(CGLOBALSTATIC)
216 CFLAGS          += -xregs=no%float
217 CFLAGS          += -xstrconst
218 CFLAGS          += $(CSOURCEDEBUGFLAGS)
219 CFLAGS          += $(CUSERFLAGS)
220 #endif /* ! codereview */

222 ASFLAGS         += $(AS_XARCH)

224 AS_INC_PATH     += -I$(DSF_DIR)/$(OBJS_DIR)

226 LINT_KMODS      += $(GENUNIX_KMODS)

228 LINT_DEFS = -m64

230 #
231 #       The following must be defined for all implementations:
232 #
233 #       MAPFILE:                ld mapfile for the build of kernel/unix.
234 #       MODSTUBS:               Module stubs source file.
235 #       GENCONST_SRC:           genconst.c
236 #       OFFSETS:                offsets.in
237 #       PLATFORM_OFFSETS:       Platform specific mach_offsets.in
238 #       FDOFFSETS:              fd_offsets.in
239 #
240 MAPFILE          = $(UTSBASE)/sun4/conf/Mapfile
241 MODSTUBS         = $(UTSBASE)/sparc/ml/modstubs.s
242 GENCONST_SRC     = $(UTSBASE)/sun4/ml/genconst.c
243 OFFSETS          = $(UTSBASE)/sun4/ml/offsets.in
244 PLATFORM_OFFSETS = $(UTSBASE)/sun4u/ml/mach_offsets.in
245 FDOFFSETS        = $(UTSBASE)/sun/io/fd_offsets.in

247 #
248 #       Define the actual specific platforms
249 #

251 MACHINE_DEFS    = -D$(PLATFORM) -D_MACHDEP -DSFMMU

253 #
254 #       Software workarounds for hardware "features"
255 #
```

```
include $(UTSBASE)/$(PLATFORM)/Makefile.workarounds

#
#       Debugging level
#
#       Special knowledge of which special debugging options effect which
#       file is used to optimize the build if these flags are changed.
#
#       XXX: The above could possibly be done for more flags and files, but
#            is left as an experiment to the interested reader. Be forewarned,
#            that excessive use could lead to maintenance difficulties.
#
#       Note: kslice can be enabled for the sun4u, but is disabled by default
#             in all cases.
#

DEBUG_DEFS_OBJ64        =
DEBUG_DEFS_DBG64        = -DDEBUG
DEBUG_DEFS              = $(DEBUG_DEFS_$(BUILD_TYPE))

DEBUG_COND_OBJ64        :sh = echo \\043
DEBUG_COND_DBG64        =
IF_DEBUG_OBJ            = $(DEBUG_COND_$(BUILD_TYPE))$(OBJS_DIR)/

$(IF_DEBUG_OBJ)trap.o           :=      DEBUG_DEFS      += -DTRAPDEBUG
$(IF_DEBUG_OBJ)mach_trap.o      :=      DEBUG_DEFS      += -DTRAPDEBUG
$(IF_DEBUG_OBJ)syscall_trap.o   :=      DEBUG_DEFS      += -DSYSCALLTRACE
$(IF_DEBUG_OBJ)clock.o          :=      DEBUG_DEFS      += -DKSLICE=0

IF_TRAPTRACE_OBJ = $(IF_DEBUG_OBJ)
# comment this out for a non-debug kernel with TRAPTRACE
#IF_TRAPTRACE_OBJ = $(OBJS_DIR)/

$(IF_TRAPTRACE_OBJ)mach_locore.o        :=      DEBUG_DEFS      += -DTRAPTRACE
$(IF_TRAPTRACE_OBJ)mlsetup.o            :=      DEBUG_DEFS      += -DTRAPTRACE
$(IF_TRAPTRACE_OBJ)syscall_trap.o       :=      DEBUG_DEFS      += -DTRAPTRACE
$(IF_TRAPTRACE_OBJ)startup.o            :=      DEBUG_DEFS      += -DTRAPTRACE
$(IF_TRAPTRACE_OBJ)mach_startup.o       :=      DEBUG_DEFS      += -DTRAPTRACE
$(IF_TRAPTRACE_OBJ)mp_startup.o         :=      DEBUG_DEFS      += -DTRAPTRACE
$(IF_TRAPTRACE_OBJ)cpu_states.o         :=      DEBUG_DEFS      += -DTRAPTRACE
$(IF_TRAPTRACE_OBJ)mach_cpu_states.o    :=      DEBUG_DEFS      += -DTRAPTRACE
$(IF_TRAPTRACE_OBJ)interrupt.o          :=      DEBUG_DEFS      += -DTRAPTRACE
$(IF_TRAPTRACE_OBJ)mach_interrupt.o     :=      DEBUG_DEFS      += -DTRAPTRACE
$(IF_TRAPTRACE_OBJ)sfmmu_asm.o          :=      DEBUG_DEFS      += -DTRAPTRACE
$(IF_TRAPTRACE_OBJ)trap_table.o         :=      DEBUG_DEFS      += -DTRAPTRACE
$(IF_TRAPTRACE_OBJ)xc.o                 :=      DEBUG_DEFS      += -DTRAPTRACE
$(IF_TRAPTRACE_OBJ)mach_xc.o            :=      DEBUG_DEFS      += -DTRAPTRACE
$(IF_TRAPTRACE_OBJ)wbuf.o               :=      DEBUG_DEFS      += -DTRAPTRACE
$(IF_TRAPTRACE_OBJ)trap.o               :=      DEBUG_DEFS      += -DTRAPTRACE
$(IF_TRAPTRACE_OBJ)mach_trap.o          :=      DEBUG_DEFS      += -DTRAPTRACE
$(IF_TRAPTRACE_OBJ)x_call.o             :=      DEBUG_DEFS      += -DTRAPTRACE
$(IF_TRAPTRACE_OBJ)spitfire_asm.o       :=      DEBUG_DEFS      += -DTRAPTRACE
$(IF_TRAPTRACE_OBJ)us3_common_asm.o     :=      DEBUG_DEFS      += -DTRAPTRACE
$(IF_TRAPTRACE_OBJ)us3_cheetah_asm.o    :=      DEBUG_DEFS      += -DTRAPTRACE
$(IF_TRAPTRACE_OBJ)us3_cheetahplus_asm.o :=     DEBUG_DEFS      += -DTRAPTRACE
$(IF_TRAPTRACE_OBJ)us3_jalapeno_asm.o   :=      DEBUG_DEFS      += -DTRAPTRACE
$(IF_TRAPTRACE_OBJ)opl_olympus_asm.o    :=      DEBUG_DEFS      += -DTRAPTRACE

# Comment these out if you don't want dispatcher lock statistics.

#$(IF_DEBUG_OBJ)lock_prim.o     := DEBUG_DEFS   += -DDISP_LOCK_STATS
#$(IF_DEBUG_OBJ)disp.o          := DEBUG_DEFS   += -DDISP_LOCK_STATS

# Comment these out if you don't want dispatcher debugging

#$(IF_DEBUG_OBJ)lock_prim.o     := DEBUG_DEFS   += -DDISP_DEBUG
```



```
#
#       Collect the preprocessor definitions to be associated with *all*
#       files.
#
ALL_DEFS         = $(MACHINE_DEFS) $(WORKAROUND_DEFS) $(DEBUG_DEFS) \
                   $(OPTION_DEFS)
GENCONST_DEFS    = $(MACHINE_DEFS) $(OPTION_DEFS)

#
# ----- TRANSITIONAL SECTION --------------------------------------------------
#

#
#       Not everything which *should* be a module is a module yet. The
#       following is a list of such objects which are currently part of
#       the base kernel but should soon become kmods.
#
MACH_NOT_YET_KMODS       = $(AUTOCONF_OBJS)

#
# ----- END OF TRANSITIONAL SECTION -------------------------------------------
#

#
#       The kernels modules which are "implementation architecture"
#       specific for this machine are enumerated below. Note that most
#       of these modules must exist (in one form or another) for each
#       architecture.
#
#       Common Drivers (usually pseudo drivers) (/kernel/drv):
#

#
#       Machine Specific Driver Modules (/kernel/drv):
#
#       XXX: How many of these are really machine specific?
#
DRV_KMODS       += bbc_beep
DRV_KMODS       += cpc
DRV_KMODS       += fd
DRV_KMODS       += rootnex sbusmem upa64s zs zsh
DRV_KMODS       += sbus
DRV_KMODS       += pcisch pcipsy simba
DRV_KMODS       += px
DRV_KMODS       += ebus
DRV_KMODS       += su
DRV_KMODS       += tod
DRV_KMODS       += power
DRV_KMODS       += epic
DRV_KMODS       += grbeep
DRV_KMODS       += pcf8584 max1617 seeprom tda8444 pca9556
DRV_KMODS       += ics951601 adm1031
DRV_KMODS       += lm75 ltc1427 pcf8591 pcf8574 ssc050 ssc100
DRV_KMODS       += pic16f819
DRV_KMODS       += pic16f747
DRV_KMODS       += adm1026
DRV_KMODS       += us
DRV_KMODS       += ppm schppm jbusppm
DRV_KMODS       += mc-us3
DRV_KMODS       += mc-us3i
DRV_KMODS       += smbus
DRV_KMODS       += db21554
DRV_KMODS       += gpio_87317
DRV_KMODS       += isadma
DRV_KMODS       += sbbc
```

```
389 DRV_KMODS       += pmubus
390 DRV_KMODS       += pmugpio
391 DRV_KMODS       += pmc
392 DRV_KMODS       += trapstat
393 DRV_KMODS       += rmc_comm
394 DRV_KMODS       += rmcadm
395 DRV_KMODS       += rmclomv
396 DRV_KMODS       += sf
397 DRV_KMODS       += nxge
398 DRV_KMODS       += i2bsc
399 DRV_KMODS       += mem_cache

401 $(CLOSED_BUILD)CLOSED_DRV_KMODS += ctsmc
402 $(CLOSED_BUILD)CLOSED_DRV_KMODS += m1535ppm
403 $(CLOSED_BUILD)CLOSED_DRV_KMODS += memtest
404 $(CLOSED_BUILD)CLOSED_DRV_KMODS += mi2cv
405 $(CLOSED_BUILD)CLOSED_DRV_KMODS += smbus_ara

407 #
408 #       Exec Class Modules (/kernel/exec):
409 #
410 EXEC_KMODS      +=

412 #
413 #       Scheduling Class Modules (/kernel/sched):
414 #
415 SCHED_KMODS     +=

417 #
418 #       File System Modules (/kernel/fs):
419 #
420 FS_KMODS        +=

422 #
423 #       Streams Modules (/kernel/strmod):
424 #
425 STRMOD_KMODS    += kb

427 #
428 #       'System' Modules (/kernel/sys):
429 #
430 SYS_KMODS       +=

432 #
433 #       'User' Modules (/kernel/misc):
434 #
435 MISC_KMODS      += bignum
436 MISC_KMODS      += obpsym bootdev vis cpr platmod md5 sha1 i2c_svc
437 MISC_KMODS      += sbd

439 MISC_KMODS      += opl_cfg
440 MISC_KMODS      += zuluvm
441 MISC_KMODS      += gptwo_cpu gptwocfg
442 MISC_KMODS      += pcie

444 #
445 #       Brand modules
446 #
447 BRAND_KMODS     += sn1_brand s10_brand

449 #
450 #       Software Cryptographic Providers (/kernel/crypto):
451 #
452 CRYPTO_KMODS    += aes
453 CRYPTO_KMODS    += arcfour
454 CRYPTO_KMODS    += des
```



```
456 #
457 #       generic-unix module (/kernel/genunix):
458 #
459 GENUNIX_KMODS   += genunix

461 #       'User' "Modules" excluded from the Full Kernel lint target:
462 #
463 $(CLOSED_BUILD)CLOSED_NLMISC_KMODS      += forthdebug

465 #
466 #       Modules eXcluded from the product:
467 #
468 XMODS           +=

470 #
471 #       cpu modules
472 #
473 CPU_KMODS       += cheetah cheetahplus jalapeno serrano spitfire hummingbird

475 #
476 #       sun4u 'TOD' Modules (/platform/.../kernel/tod):
477 #
478 TOD_KMODS       += todds1287 todds1337 todmostek todstarfire
479 TOD_KMODS       += todm5819 todblade todbq4802 todsg todopl
480 TOD_KMODS       += todm5819p_rmc todstarcat

482 $(CLOSED_BUILD)CLOSED_TOD_KMODS += todm5823

484 #
485 #       Performance Counter BackEnd Modules (/usr/kernel/pcbe):
486 #
487 PCBE_KMODS      += us234_pcbe
488 PCBE_KMODS      += opl_pcbe
```

```
**********************************************************
   12338 Sun Jul 28 20:29:09 2013
new/usr/src/uts/sun4v/Makefile.sun4v.shared
3735 should include an empty make variable in the default CFLAGS/CCFLAGS
3844 the build should make source-level debugging easier
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Garrett D'Amore <garrett@damore.org>
**********************************************************
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#

#
# Copyright 2010 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#
#       This makefile contains the common definitions for the sun4v unix
#       and all sun4v implementation architecture dependent modules.
#

#
#       Machine type (implementation architecture):
#
PLATFORM         = sun4v
LINKED_PLATFORMS += SUNW,Sun-Fire-T1000
LINKED_PLATFORMS += SUNW,SPARC-Enterprise-T5120
LINKED_PLATFORMS += SUNW,SPARC-Enterprise-T5220
LINKED_PLATFORMS += SUNW,T5140
LINKED_PLATFORMS += SUNW,T5240
LINKED_PLATFORMS += SUNW,T5440
LINKED_PLATFORMS += SUNW,SPARC-Enterprise-T1000
LINKED_PLATFORMS += SUNW,Sun-Blade-T6300
LINKED_PLATFORMS += SUNW,Sun-Blade-T6320
LINKED_PLATFORMS += SUNW,Netra-CP3260
LINKED_PLATFORMS += SUNW,Netra-T5220
LINKED_PLATFORMS += SUNW,USBRDT-5240
LINKED_PLATFORMS += SUNW,Netra-T5440
LINKED_PLATFORMS += SUNW,Sun-Blade-T6340
PROMIF           = ieee1275
PSMBASE          = $(UTSBASE)/../psm

#
#       uname -m value
#
UNAME_M         = $(PLATFORM)

#
# Definitions for the platform-specific /platform directories.
#
```



```
# PLATFORMS designates those sun4v machines which have no platform
# specific code.
#
# IMPLEMENTATIONS is used to designate sun4v machines which have
# platform specific modules.  All code specific to a given implementation
# resides in the appropriately named subdirectory.   This requires
# these platforms to have their own Makefiles to define ROOT_PLAT_DIRS,
# USR_PLAT_DIRS, etc.
# The number of IMPLEMENTATIONS should not grow!
#
# So if we had an implementation named 'foo', we would need the following
# Makefiles in the foo subdirectory:
#
#       sun4v/foo/Makefile
#       sun4v/foo/Makefile.foo
#       sun4v/foo/Makefile.targ
#

#
# all PLATFORMS that do not belong in the $(IMPLEMENTATIONS) list.
# This list should be empty.  A platform without platform modules
# is a plain, generic sun4v platform.
#
#IMPLEMENTED_PLATFORM   =
#PLATFORMS                = $(IMPLEMENTED_PLATFORM)

IMPLEMENTATIONS         = ontario montoya huron maramba

#ROOT_PLAT_DIRS         = $(PLATFORMS:%=$(ROOT_PLAT_DIR)/%)
#USR_PLAT_DIRS          = $(PLATFORMS:%=$(USR_PLAT_DIR)/%)

#USR_DESKTOP_DIR                = $(USR_PLAT_DIR)/$(IMPLEMENTED_PLATFORM)
#USR_DESKTOP_INC_DIR    = $(USR_DESKTOP_DIR)/include
#USR_DESKTOP_SBIN_DIR   = $(USR_DESKTOP_DIR)/sbin
#USR_DESKTOP_LIB_DIR    = $(USR_DESKTOP_DIR)/lib

#
#       Define supported builds
#
DEF_BUILDS      = $(DEF_BUILDS64)
ALL_BUILDS      = $(ALL_BUILDS64)

#
#       Everybody needs to know how to build modstubs.o and to locate unix.o
#
UNIX_DIR         = $(UTSBASE)/$(PLATFORM)/unix
GENLIB_DIR       = $(UTSBASE)/$(PLATFORM)/genunix
MODSTUBS_DIR     = $(UNIX_DIR)
DSF_DIR          = $(UTSBASE)/$(PLATFORM)/genassym
LINTS_DIR        = $(OBJS_DIR)
LINT_LIB_DIR     = $(UTSBASE)/$(PLATFORM)/lint-libs/$(OBJS_DIR)

DTRACESTUBS_O    = $(OBJS_DIR)/dtracestubs.o
DTRACESTUBS      = $(OBJS_DIR)/libdtracestubs.so

UNIX_O           = $(UNIX_DIR)/$(OBJS_DIR)/unix.o
MODSTUBS_O       = $(MODSTUBS_DIR)/$(OBJS_DIR)/modstubs.o
GENLIB           = $(GENLIB_DIR)/$(OBJS_DIR)/libgenunix.so

LINT_LIB         = $(LINT_LIB_DIR)/llib-lunix.ln
GEN_LINT_LIB     = $(LINT_LIB_DIR)/llib-lgenunix.ln

LINT64_DIRS      = $(LINT64_BUILDS:%=$(UTSBASE)/$(PLATFORM)/lint-libs/%)
LINT64_FILES     = $(LINT64_DIRS:%=%/llib-l$(MODULE).ln)

#
```

```
125 #       cpu and platform modules need to know how to build their own symcheck mo
126 #
127 PLATMOD          = platmod
128 PLATLIB          = $(PLAT_DIR)/$(OBJS_DIR)/libplatmod.so

130 CPUNAME          = cpu
131 CPULIB           = $(CPU_DIR)/$(OBJS_DIR)/libcpu.so

133 SYM_MOD          = $(OBJS_DIR)/unix.sym

135 #
136 #       Include the makefiles which define build rule templates, the
137 #       collection of files per module, and a few specific flags. Note
138 #       that order is significant, just as with an include path. The
139 #       first build rule template which matches the files name will be
140 #       used. By including these in order from most machine dependent
141 #       to most machine independent, we allow a machine dependent file
142 #       to be used in preference over a machine independent version
143 #       (Such as a machine specific optimization, which preserves the
144 #       interfaces.)
145 #
146 include $(UTSBASE)/sun4/Makefile.files
147 include $(UTSTREE)/$(PLATFORM)/Makefile.files
148 include $(UTSBASE)/sfmmu/Makefile.files
149 include $(UTSBASE)/sparc/v9/Makefile.files
150 include $(UTSBASE)/sparc/Makefile.files
151 include $(UTSTREE)/sun/Makefile.files
152 include $(SRC)/psm/promif/$(PROMIF)/common/Makefile.files
153 include $(SRC)/psm/promif/$(PROMIF)/$(PLATFORM)/Makefile.files
154 include $(UTSTREE)/common/Makefile.files

156 #
157 #       Include machine independent rules. Note that this does not imply
158 #       that the resulting module from rules in Makefile.uts is machine
159 #       independent. Only that the build rules are machine independent.
160 #
161 include $(UTSBASE)/Makefile.uts

163 CTFMERGE_GUDIR  = sun4v

165 #
166 #       machine specific optimization, override default in Makefile.master
167 #
168 CC_XARCH        = -m64 -xarch=sparcvis
169 AS_XARCH        = -xarch=v9v
170 COPTIMIZE       = -xO3
171 CCMODE          = -Xa

173 CFLAGS          = -xchip=ultra $(CCABS32) $(CCREGSYM)
174 CFLAGS          += $(CC_XARCH)
175 CFLAGS          += $(COPTIMIZE)
176 CFLAGS          += $(EXTRA_CFLAGS)
177 CFLAGS          += $(XAOPT)
178 CFLAGS          += $(INLINES) -D_ASM_INLINES
179 CFLAGS          += $(CCMODE)
180 CFLAGS          += $(SPACEFLAG)
181 CFLAGS          += $(CERRWARN)
182 CFLAGS          += $(CTF_FLAGS_$(CLASS))
183 CFLAGS          += $(C99MODE)
184 CFLAGS          += $(CCUNBOUND)
185 CFLAGS          += $(CCNOAUTOINLINE)
186 CFLAGS          += $(CCSTATICSYM)
187 CFLAGS          += $(CC32BITCALLERS)
188 CFLAGS          += $(IROPTFLAG)
189 CFLAGS          += $(CGLOBALSTATIC)
190 CFLAGS          += -xregs=no%float
```



```
191 CFLAGS          += -xstrconst
192 CFLAGS          += $(CSOURCEDEBUGFLAGS)
193 CFLAGS          += $(CUSERFLAGS)

195 #endif /* ! codereview */
196 CPPFLAGS        += -DGLREG

198 ASFLAGS         += $(AS_XARCH) -DGLREG

200 AS_INC_PATH     += -I$(DSF_DIR)/$(OBJS_DIR)

202 LINT_KMODS      += $(GENUNIX_KMODS)

204 LINT_DEFS       = -m64

206 #
207 #       The following must be defined for all implementations:
208 #
209 #       MAPFILE:                ld mapfile for the build of kernel/unix.
210 #       MODSTUBS:               Module stubs source file.
211 #       GENCONST_SRC:           genconst.c
212 #       OFFSETS:                offsets.in
213 #       PLATFORM_OFFSETS:       Platform specific mach_offsets.in
214 #       FDOFFSETS:              fd_offsets.in
215 #
216 MAPFILE          = $(UTSBASE)/sun4/conf/Mapfile
217 MODSTUBS         = $(UTSBASE)/sparc/ml/modstubs.s
218 GENCONST_SRC     = $(UTSBASE)/sun4/ml/genconst.c
219 OFFSETS          = $(UTSBASE)/sun4/ml/offsets.in
220 PLATFORM_OFFSETS = $(UTSBASE)/sun4v/ml/mach_offsets.in
221 FDOFFSETS        = $(UTSBASE)/sun/io/fd_offsets.in

223 #
224 #       Define the actual specific platforms
225 #

227 MACHINE_DEFS    = -D$(PLATFORM) -D_MACHDEP -DSFMMU
228 MACHINE_DEFS    += -DMAX_MEM_NODES=8

230 #
231 #       Software workarounds for hardware "features"
232 #

234 include $(UTSBASE)/$(PLATFORM)/Makefile.workarounds

236 #
237 #       Debugging level
238 #
239 #       Special knowledge of which special debugging options effect which
240 #       file is used to optimize the build if these flags are changed.
241 #
242 #       XXX: The above could possibly be done for more flags and files, but
243 #            is left as an experiment to the interested reader. Be forewarned,
244 #            that excessive use could lead to maintenance difficulties.
245 #
246 #       Note: kslice can be enabled for the sun4v, but is disabled by default
247 #             in all cases.
248 #

250 DEBUG_DEFS_OBJ64        =
251 DEBUG_DEFS_DBG64        = -DDEBUG
252 DEBUG_DEFS              = $(DEBUG_DEFS_$(BUILD_TYPE))

254 DEBUG_COND_OBJ64        :sh = echo \\043
255 DEBUG_COND_DBG64        =
256 IF_DEBUG_OBJ            = $(DEBUG_COND_$(BUILD_TYPE))$(OBJS_DIR)/
```

```
$(IF_DEBUG_OBJ)trap.o           :=      DEBUG_DEFS      += -DTRAPDEBUG
$(IF_DEBUG_OBJ)mach_trap.o      :=      DEBUG_DEFS      += -DTRAPDEBUG
$(IF_DEBUG_OBJ)syscall_trap.o   :=      DEBUG_DEFS      += -DSYSCALLTRACE
$(IF_DEBUG_OBJ)clock.o          :=      DEBUG_DEFS      += -DKSLICE=0

IF_TRAPTRACE_OBJ = $(IF_DEBUG_OBJ)
# comment this out for a non-debug kernel with TRAPTRACE
#IF_TRAPTRACE_OBJ = $(OBJS_DIR)/

$(IF_TRAPTRACE_OBJ)mach_locore.o        :=      DEBUG_DEFS      += -DTRAPTRACE
$(IF_TRAPTRACE_OBJ)mlsetup.o            :=      DEBUG_DEFS      += -DTRAPTRACE
$(IF_TRAPTRACE_OBJ)syscall_trap.o       :=      DEBUG_DEFS      += -DTRAPTRACE
$(IF_TRAPTRACE_OBJ)startup.o            :=      DEBUG_DEFS      += -DTRAPTRACE
$(IF_TRAPTRACE_OBJ)mach_startup.o       :=      DEBUG_DEFS      += -DTRAPTRACE
$(IF_TRAPTRACE_OBJ)mp_startup.o         :=      DEBUG_DEFS      += -DTRAPTRACE
$(IF_TRAPTRACE_OBJ)cpu_states.o         :=      DEBUG_DEFS      += -DTRAPTRACE
$(IF_TRAPTRACE_OBJ)mach_cpu_states.o    :=      DEBUG_DEFS      += -DTRAPTRACE
$(IF_TRAPTRACE_OBJ)interrupt.o          :=      DEBUG_DEFS      += -DTRAPTRACE
$(IF_TRAPTRACE_OBJ)mach_interrupt.o     :=      DEBUG_DEFS      += -DTRAPTRACE
$(IF_TRAPTRACE_OBJ)sfmmu_asm.o          :=      DEBUG_DEFS      += -DTRAPTRACE
$(IF_TRAPTRACE_OBJ)trap_table.o         :=      DEBUG_DEFS      += -DTRAPTRACE
$(IF_TRAPTRACE_OBJ)xc.o                 :=      DEBUG_DEFS      += -DTRAPTRACE
$(IF_TRAPTRACE_OBJ)mach_xc.o            :=      DEBUG_DEFS      += -DTRAPTRACE
$(IF_TRAPTRACE_OBJ)wbuf.o               :=      DEBUG_DEFS      += -DTRAPTRACE
$(IF_TRAPTRACE_OBJ)trap.o               :=      DEBUG_DEFS      += -DTRAPTRACE
$(IF_TRAPTRACE_OBJ)mach_trap.o          :=      DEBUG_DEFS      += -DTRAPTRACE
$(IF_TRAPTRACE_OBJ)x_call.o             :=      DEBUG_DEFS      += -DTRAPTRACE

# Comment these out if you don't want dispatcher lock statistics.

#$(IF_DEBUG_OBJ)lock_prim.o     := DEBUG_DEFS   += -DDISP_LOCK_STATS
#$(IF_DEBUG_OBJ)disp.o          := DEBUG_DEFS   += -DDISP_LOCK_STATS

# Comment these out if you don't want dispatcher debugging

#$(IF_DEBUG_OBJ)lock_prim.o     := DEBUG_DEFS   += -DDISP_DEBUG

#
#       Collect the preprocessor definitions to be associated with *all*
#       files.
#
ALL_DEFS         = $(MACHINE_DEFS) $(WORKAROUND_DEFS) $(DEBUG_DEFS) \
                   $(OPTION_DEFS)
GENCONST_DEFS    = $(MACHINE_DEFS) $(OPTION_DEFS)

#
# ----- TRANSITIONAL SECTION --------------------------------------------------
#

#
#       Not everything which *should* be a module is a module yet. The
#       following is a list of such objects which are currently part of
#       the base kernel but should soon become kmods.
#
MACH_NOT_YET_KMODS       = $(AUTOCONF_OBJS)

#
# ----- END OF TRANSITIONAL SECTION -------------------------------------------
#

#
#       The kernels modules which are "implementation architecture"
#       specific for this machine are enumerated below. Note that most
#       of these modules must exist (in one form or another) for each
#       architecture.
```



```
#
#       Common Drivers (usually pseudo drivers) (/kernel/drv):
#

#
#       Machine Specific Driver Modules (/kernel/drv):
#
DRV_KMODS       += bge
DRV_KMODS       += cnex
DRV_KMODS       += cpc
DRV_KMODS       += drctl
DRV_KMODS       += ds_pri
DRV_KMODS       += ds_snmp
DRV_KMODS       += ebus
DRV_KMODS       += fpc
DRV_KMODS       += glvc
DRV_KMODS       += mdesc
DRV_KMODS       += niumx
DRV_KMODS       += ntwdt
DRV_KMODS       += nxge
DRV_KMODS       += n2piupc
DRV_KMODS       += iospc
DRV_KMODS       += n2rng
DRV_KMODS       += px
DRV_KMODS       += qcn
DRV_KMODS       += rootnex
DRV_KMODS       += su
DRV_KMODS       += tpm
DRV_KMODS       += trapstat
DRV_KMODS       += vcc
DRV_KMODS       += vdc
DRV_KMODS       += vds
DRV_KMODS       += vldc
DRV_KMODS       += vlds
DRV_KMODS       += vnet
DRV_KMODS       += vnex
DRV_KMODS       += vsw

$(CLOSED_BUILD)CLOSED_DRV_KMODS += bmc
$(CLOSED_BUILD)CLOSED_DRV_KMODS += memtest
$(CLOSED_BUILD)CLOSED_DRV_KMODS += ncp
$(CLOSED_BUILD)CLOSED_DRV_KMODS += n2cp

#
#       Exec Class Modules (/kernel/exec):
#
EXEC_KMODS      +=

#
#       Scheduling Class Modules (/kernel/sched):
#
SCHED_KMODS     +=

#
#       File System Modules (/kernel/fs):
#
FS_KMODS        +=

#
#       Streams Modules (/kernel/strmod):
#
# STRMOD_KMODS  += kb

#
#       'System' Modules (/kernel/sys):
#
```

```
389 SYS_KMODS       +=

391 #
392 #       'User' Modules (/kernel/misc):
393 #
394 MISC_KMODS      += bootdev
395 MISC_KMODS      += dr_cpu
396 MISC_KMODS      += dr_io
397 MISC_KMODS      += dr_mem
398 MISC_KMODS      += ds
399 MISC_KMODS      += fault_iso
400 MISC_KMODS      += ldc
401 MISC_KMODS      += obpsym
402 MISC_KMODS      += platmod
403 MISC_KMODS      += platsvc
404 MISC_KMODS      += vis
405 MISC_KMODS      += pcie

407 #       md5 optimized for Niagara
408 #
409 MISC_KMODS      += md5

411 #
412 #       Brand modules
413 #
414 BRAND_KMODS     += sn1_brand s10_brand

416 #
417 #       Software Cryptographic Providers (/kernel/crypto):
418 #
419 CRYPTO_KMODS    += arcfour

421 #
422 #       generic-unix module (/kernel/genunix):
423 #
424 GENUNIX_KMODS   += genunix

426 #       'User' "Modules" excluded from the Full Kernel lint target:
427 #
428 $(CLOSED_BUILD)CLOSED_NLMISC_KMODS      += forthdebug

430 #
431 #       Modules eXcluded from the product:
432 #
433 XMODS           +=

435 #
436 #       cpu modules
437 #
438 CPU_KMODS       += generic niagara niagara2 vfalls kt

440 LINT_CPU_KMODS  += generic

442 #
443 #       Performance Counter BackEnd Modules (/usr/kernel/pcbe):
444 #
445 PCBE_KMODS      += niagara_pcbe
446 PCBE_KMODS      += niagara2_pcbe
447 PCBE_KMODS      += vfalls_pcbe
448 PCBE_KMODS      += kt_pcbe
```