```
**********************************************************
   36670 Mon Dec 10 01:35:46 2018
new/usr/src/Makefile.master
Code review comments
**********************************************************
```

```
#
# 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.
# Copyright 2014 Garrett D'Amore <garrett@damore.org>
# Copyright 2015, OmniTI Computer Consulting, Inc. All rights reserved.
# Copyright 2015 Gary Mills
# Copyright 2015 Igor Kozhukhov <ikozhukhov@gmail.com>
# Copyright 2016 Toomas Soome <tsoome@me.com>
# Copyright 2018 OmniOS Community Edition (OmniOSce) Association.
#

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

#
# Adjunct root, containing an additional proto area to be used for headers
# and libraries.
#
ADJUNCT_PROTO=

#
# Adjunct for building things that run on the build machine.
#
NATIVE_ADJUNCT= /usr

#
# RELEASE_BUILD should be cleared for final release builds.
# NOT_RELEASE_BUILD is exactly what the name implies.
#
# __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=
RELEASE_BUILD=                          $(POUND_SIGN)
$(RELEASE_BUILD)NOT_RELEASE_BUILD=      $(POUND_SIGN)
PATCH_BUILD=                            $(POUND_SIGN)

# 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=)

# 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.
# __GNUC and GCC 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.
#

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

# Allow build-time "configuration" to enable or disable some things.
# The default is POUND_SIGN, meaning "not enabled". If the environment
# passes in an override like ENABLE_SMB_PRINTING= (empty) that will
# uncomment things in the lower Makefiles to enable the feature.
ENABLE_SMB_PRINTING=    $(POUND_SIGN)

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

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

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

# define runtime JAVA_HOME, primarily for cmd/pools/poold
JAVA_HOME=      /usr/java
# define buildtime JAVA_ROOT
JAVA_ROOT=      /usr/java
# Build uses java7 by default.  Pass one the variables below set to empty
# string in the environment to override.
BLD_JAVA_6=     $(POUND_SIGN)
BLD_JAVA_8=     $(POUND_SIGN)
```

```
128 GNUC_ROOT=      /opt/gcc/4.4.4
129 GCCLIBDIR=      $(GNUC_ROOT)/lib
130 GCCLIBDIR64=    $(GNUC_ROOT)/lib/$(MACH64)

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

134 RPCGEN=         /usr/bin/rpcgen
135 STABS=          $(ONBLD_TOOLS)/bin/$(MACH)/stabs
136 ELFEXTRACT=     $(ONBLD_TOOLS)/bin/$(MACH)/elfextract
137 MBH_PATCH=      $(ONBLD_TOOLS)/bin/$(MACH)/mbh_patch
138 BTXLD=          $(ONBLD_TOOLS)/bin/$(MACH)/btxld
139 VTFONTCVT=      $(ONBLD_TOOLS)/bin/$(MACH)/vtfontcvt
140 # echo(1) and true(1) are specified without absolute paths, so that the shell
141 # spawned by make(1) may use the built-in versions.  This is minimally
142 # problematic, as the shell spawned by make(1) is known and under control, the
143 # only risk being if the shell falls back to $PATH.
144 #
145 # We specifically want an echo(1) that does interpolation of escape sequences,
146 # which ksh93, /bin/sh, and bash will all provide.
147 ECHO=           echo
148 TRUE=           true
149 INS=            $(ONBLD_TOOLS)/bin/$(MACH)/install
150 SYMLINK=        /usr/bin/ln -s
151 LN=             /usr/bin/ln
152 MKDIR=          /usr/bin/mkdir
153 CHMOD=          /usr/bin/chmod
154 MV=             /usr/bin/mv -f
155 RM=             /usr/bin/rm -f
156 CUT=            /usr/bin/cut
157 NM=             /usr/ccs/bin/nm
158 DIFF=           /usr/bin/diff
159 GREP=           /usr/bin/grep
160 EGREP=          /usr/bin/egrep
161 ELFWRAP=        /usr/bin/elfwrap
162 KSH93=          /usr/bin/ksh93
163 SED=            /usr/bin/sed
164 AWK=            /usr/bin/nawk
165 CP=             /usr/bin/cp -f
166 MCS=            /usr/ccs/bin/mcs
167 CAT=            /usr/bin/cat
168 ELFDUMP=        /usr/ccs/bin/elfdump
169 M4=             /usr/bin/m4
170 STRIP=          /usr/ccs/bin/strip
171 LEX=            /usr/ccs/bin/lex
172 FLEX=           /usr/bin/flex
173 YACC=           /usr/ccs/bin/yacc
174 CPP=            /usr/lib/cpp
175 ANSI_CPP=       $(GNUC_ROOT)/bin/cpp
176 JAVAC=          $(JAVA_ROOT)/bin/javac
177 JAVAH=          $(JAVA_ROOT)/bin/javah
178 JAVADOC=        $(JAVA_ROOT)/bin/javadoc
179 RMIC=           $(JAVA_ROOT)/bin/rmic
180 JAR=            $(JAVA_ROOT)/bin/jar
181 CTFCONVERT=     $(ONBLD_TOOLS)/bin/$(MACH)/ctfconvert
182 CTFMERGE=       $(ONBLD_TOOLS)/bin/$(MACH)/ctfmerge
183 CTFSTABS=       $(ONBLD_TOOLS)/bin/$(MACH)/ctfstabs
184 CTFSTRIP=       $(ONBLD_TOOLS)/bin/$(MACH)/ctfstrip
185 NDRGEN=         $(ONBLD_TOOLS)/bin/$(MACH)/ndrgen
186 GENOFFSETS=     $(ONBLD_TOOLS)/bin/genoffsets
187 XREF=           $(ONBLD_TOOLS)/bin/xref
188 FIND=           /usr/bin/find
189 PERL=           /usr/bin/perl
190 PERL_VERSION=   5.10.0
191 PERL_PKGVERS=   -510
192 PERL_ARCH =             i86pc-solaris-64int
193 $(SPARC_BLD)PERL_ARCH = sun4-solaris-64int
```



```
194 PYTHON_VERSION= 2.7
195 PYTHON_PKGVERS= -27
196 PYTHON_SUFFIX=
197 PYTHON=         /usr/bin/python$(PYTHON_VERSION)
198 PYTHON3_VERSION=        3.5
199 PYTHON3_PKGVERS=        -35
200 PYTHON3_SUFFIX=         m
201 PYTHON3=                /usr/bin/python$(PYTHON3_VERSION)
202 SORT=           /usr/bin/sort
203 TR=             /usr/bin/tr
204 TOUCH=          /usr/bin/touch
205 WC=             /usr/bin/wc
206 XARGS=          /usr/bin/xargs
207 ELFEDIT=        /usr/bin/elfedit
208 DTRACE=         /usr/sbin/dtrace -xnolibs
209 UNIQ=           /usr/bin/uniq
210 TAR=            /usr/bin/tar
211 ASTBINDIR=      /usr/ast/bin
212 MSGCC=          $(ASTBINDIR)/msgcc
213 MSGFMT=         /usr/bin/msgfmt -s
214 LCDEF=          $(ONBLD_TOOLS)/bin/$(MACH)/localedef
215 TIC=            $(ONBLD_TOOLS)/bin/$(MACH)/tic
216 ZIC=            $(ONBLD_TOOLS)/bin/$(MACH)/zic
217 OPENSSL=        /usr/bin/openssl

219 FILEMODE=       644
220 DIRMODE=        755

222 # Declare that nothing should be built in parallel.
223 # Individual Makefiles can use the .PARALLEL target to declare otherwise.
224 .NO_PARALLEL:

226 # For stylistic checks
227 #
228 # Note that the X and C checks are not used at this time and may need
229 # modification when they are actually used.
230 #
231 CSTYLE=         $(ONBLD_TOOLS)/bin/cstyle
232 CSTYLE_TAIL=
233 HDRCHK=         $(ONBLD_TOOLS)/bin/hdrchk
234 HDRCHK_TAIL=
235 JSTYLE=         $(ONBLD_TOOLS)/bin/jstyle

237 DOT_H_CHECK=    \
238         @$(ECHO) "checking $<"; $(CSTYLE) $< $(CSTYLE_TAIL); \
239         $(HDRCHK) $< $(HDRCHK_TAIL)

241 DOT_X_CHECK=    \
242         @$(ECHO) "checking $<"; $(RPCGEN) -C -h $< | $(CSTYLE) $(CSTYLE_TAIL); \
243         $(RPCGEN) -C -h $< | $(HDRCHK) $< $(HDRCHK_TAIL)

245 DOT_C_CHECK=    \
246         @$(ECHO) "checking $<"; $(CSTYLE) $< $(CSTYLE_TAIL)

248 MANIFEST_CHECK= \
249         @$(ECHO) "checking $<"; \
250         SVCCFG_DTD=$(SRC)/cmd/svc/dtd/service_bundle.dtd.1 \
251         SVCCFG_REPOSITORY=$(SRC)/cmd/svc/seed/global.db \
252         SVCCFG_CONFIGD_PATH=$(SRC)/cmd/svc/configd/svc.configd-native \
253         $(SRC)/cmd/svc/svccfg/svccfg-native validate $<

255 INS.file=       $(RM) $@; $(INS) -s -m $(FILEMODE) -f $(@D) $<
256 INS.dir=        $(INS) -s -d -m $(DIRMODE) $@
257 # installs and renames at once
258 #
259 INS.rename=     $(INS.file); $(MV) $(@D)/$(<F) $@
```

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

266 #
267 # Python bakes the mtime of the .py file into the compiled .pyc and
268 # rebuilds if the baked-in mtime != the mtime of the source file
269 # (rather than only if it's less than), thus when installing python
270 # files we must make certain to not adjust the mtime of the source
271 # (.py) file.
272 #
273 INS.pyfile=     $(RM) $@; $(SED) -e "1s:^\#!@PYTHON@:\#!$(PYTHON):" < $< > $@; $

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

285 MACH64_1=       $(MACH:sparc=sparcv9)
286 MACH64=         $(MACH64_1:i386=amd64)

288 MACH32_1=       $(MACH:sparc=sparcv7)
289 MACH32=         $(MACH32_1:i386=i86)

291 sparc_BUILD64=
292 i386_BUILD64=
293 BUILD64=        $($(MACH)_BUILD64)

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

303 #
304 # C compiler verbose mode. This is so we can enable it globally,
305 # but turn it off in the lower level makefiles of things we cannot
306 # (or aren't going to) fix.
307 #
308 CCVERBOSE=              -v

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

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

323 # Prevent the removal of static symbols by the SPARC code generator (cg).
324 # The x86 code generator (ube) does not remove such symbols and as such
325 # using this workaround is not applicable for x86.
```



```
326 #
327 CCSTATICSYM=            -Wc,-Qassembler-ounrefsym=0
328 #
329 # generate 32-bit addresses in the v9 kernel. Saves memory.
330 CCABS32=                -Wc,-xcode=abs32
331 #
332 # generate v9 code which tolerates callers using the v7 ABI, for the sake of
333 # system calls.
334 CC32BITCALLERS=         -_gcc=-massume-32bit-callers

336 # GCC, especially, is increasingly beginning to auto-inline functions and
337 # sadly does so separately not under the general -fno-inline-functions
338 # Additionally, we wish to prevent optimisations which cause GCC to clone
339 # functions -- in particular, these may cause unhelpful symbols to be
340 # emitted instead of function names
341 CCNOAUTOINLINE= \
342         -_gcc=-fno-inline-small-functions \
343         -_gcc=-fno-inline-functions-called-once \
344         -_gcc=-fno-ipa-cp \
345         -_gcc7=-fno-ipa-icf \
346         -_gcc8=-fno-ipa-icf \
347         -_gcc7=-fno-clone-functions \
348         -_gcc8=-fno-clone-functions

350 # GCC may put functions in different named sub-sections of .text based on
351 # their presumed calling frequency.  At least in the kernel, where we actually
352 # deliver relocatable objects, we don't want this to happen.
353 #
354 # Since at present we don't benefit from this even in userland, we disable it gl
355 # but the application of this may move into usr/src/uts/ in future.
356 CCNOREORDER=    \
357         -_gcc7=-fno-reorder-functions \
358         -_gcc8=-fno-reorder-functions

360 # One optimization the compiler might perform is to turn this:
361 #       #pragma weak foo
362 #       extern int foo;
363 #       if (&foo)
364 #               foo = 5;
365 # into
366 #       foo = 5;
367 # Since we do some of this (foo might be referenced in common kernel code
368 # but provided only for some cpu modules or platforms), we disable this
369 # optimization.
370 #
371 sparc_CCUNBOUND = -Wd,-xsafe=unboundsym
372 i386_CCUNBOUND  =
373 CCUNBOUND       = $($(MACH)_CCUNBOUND)

375 #
376 # compiler '-xarch' flag. This is here to centralize it and make it
377 # overridable for testing.
378 sparc_XARCH=    -m32
379 sparcv9_XARCH=  -m64
380 i386_XARCH=     -m32
381 amd64_XARCH=    -m64 -Ui386 -U__i386

383 # assembler '-xarch' flag.  Different from compiler '-xarch' flag.
384 sparc_AS_XARCH=         -xarch=v8plus
385 sparcv9_AS_XARCH=       -xarch=v9
386 i386_AS_XARCH=
387 amd64_AS_XARCH=         -xarch=amd64 -P -Ui386 -U__i386

389 #
390 # These flags define what we need to be 'standalone' i.e. -not- part
391 # of the rather more cosy userland environment.  This basically means
```

```
392 # the kernel.
393 #
394 # XX64  future versions of gcc will make -mcmodel=kernel imply -mno-red-zone
395 #
396 sparc_STAND_FLAGS=      -_gcc=-ffreestanding
397 sparcv9_STAND_FLAGS=    -_gcc=-ffreestanding
398 # Disabling MMX also disables 3DNow, disabling SSE also disables all later
399 # additions to SSE (SSE2, AVX ,etc.)
400 NO_SIMD=                -_gcc=-mno-mmx -_gcc=-mno-sse
401 i386_STAND_FLAGS=       -_gcc=-ffreestanding $(NO_SIMD)
402 amd64_STAND_FLAGS=      -xmodel=kernel $(NO_SIMD)

404 SAVEARGS=               -Wu,-save_args
405 amd64_STAND_FLAGS       += $(SAVEARGS)

407 STAND_FLAGS_32 = $($(MACH)_STAND_FLAGS)
408 STAND_FLAGS_64 = $($(MACH64)_STAND_FLAGS)

410 #
411 # disable the incremental linker
412 ILDOFF=                 -xildoff
413 #
414 XFFLAG=                 -xF=%all
415 XESS=                   -xs
416 XSTRCONST=              -xstrconst

418 #
419 # turn warnings into errors (C)
420 CERRWARN = -errtags=yes -errwarn=%all
421 CERRWARN += -erroff=E_EMPTY_TRANSLATION_UNIT
422 CERRWARN += -erroff=E_STATEMENT_NOT_REACHED

424 CERRWARN += -_gcc=-Wno-missing-braces
425 CERRWARN += -_gcc=-Wno-sign-compare
426 CERRWARN += -_gcc=-Wno-unknown-pragmas
427 CERRWARN += -_gcc=-Wno-unused-parameter
428 CERRWARN += -_gcc=-Wno-missing-field-initializers

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

433 # DEBUG v. -nd make for frequent unused variables, empty conditions, etc. in
434 # -nd builds
435 $(RELEASE_BUILD)CERRWARN += -_gcc=-Wno-unused
436 $(RELEASE_BUILD)CERRWARN += -_gcc=-Wno-empty-body

438 #
439 # turn warnings into errors (C++)
440 CCERRWARN=              -xwe

442 # C standard.  Keep Studio flags until we get rid of lint.
443 CSTD_GNU89=     -xc99=%none
444 CSTD_GNU99=     -xc99=%all
445 CSTD=           $(CSTD_GNU89)
446 C99LMODE=       $(CSTD:-xc99%=-Xc99%)

448 # In most places, assignments to these macros should be appended with +=
449 # (CPPFLAGS.first allows values to be prepended to CPPFLAGS).
450 sparc_CFLAGS=   $(sparc_XARCH) $(CCSTATICSYM)
451 sparcv9_CFLAGS= $(sparcv9_XARCH) -dalign $(CCVERBOSE) $(V9ABIWARN) $(CCREGSYM) \
452                 $(CCSTATICSYM)
453 i386_CFLAGS=    $(i386_XARCH)
454 amd64_CFLAGS=   $(amd64_XARCH)

456 sparc_ASFLAGS=  $(sparc_AS_XARCH)
457 sparcv9_ASFLAGS=$(sparcv9_AS_XARCH)
```



```
458 i386_ASFLAGS=   $(i386_AS_XARCH)
459 amd64_ASFLAGS=  $(amd64_AS_XARCH)

461 #
462 sparc_COPTFLAG=         -xO3
463 sparcv9_COPTFLAG=       -xO3
464 i386_COPTFLAG=          -O
465 amd64_COPTFLAG=         -xO3

467 COPTFLAG= $($(MACH)_COPTFLAG)
468 COPTFLAG64= $($(MACH64)_COPTFLAG)

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

474 # Direct the Sun Studio compiler to use a static globalization prefix based on t
475 # name of the module rather than something unique. Otherwise, objects
476 # will not build deterministically, as subsequent compilations of identical
477 # source will yeild objects that always look different.
478 #
479 # In the same spirit, this will also remove the date from the N_OPT stab.
480 CGLOBALSTATIC= -W0,-xglobalstatic

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

486 #
487 # We force the compilers to generate the debugging information best understood
488 # by the CTF tools.  With Sun Studio this is stabs due to bugs in the Studio
489 # compilers.  With GCC this is DWARF v2.
487 # Default debug format for Sun Studio 11 is dwarf, so force it to
488 # generate stabs.
490 #
491 DEBUGFORMAT=    -_cc=-xdebugformat=stabs -_gcc=-gdwarf-2

493 #
494 # Ask the compiler to include debugging information
495 #
496 CCGDEBUG= -g $(DEBUGFORMAT)

498 #
499 # Flags used to build in debug mode for ctf generation.
498 # Flags used to build in debug mode for ctf generation.  Bugs in the Devpro
499 # compilers currently prevent us from building with cc-emitted DWARF.
500 #
501 CTF_FLAGS_sparc = $(CCGDEBUG) -Wc,-Qiselect-T1 $(CSTD) $(CNOGLOBAL)
502 CTF_FLAGS_i386  = $(CCGDEBUG) $(CSTD) $(CNOGLOBAL)

504 CTF_FLAGS_sparcv9       = $(CTF_FLAGS_sparc)
505 CTF_FLAGS_amd64         = $(CTF_FLAGS_i386)

507 # Sun Studio produces broken userland code when saving arguments.
508 $(__GNUC)CTF_FLAGS_amd64 += $(SAVEARGS)

510 CTF_FLAGS_32    = $(CTF_FLAGS_$(MACH))
511 CTF_FLAGS_64    = $(CTF_FLAGS_$(MACH64))
512 CTF_FLAGS       = $(CTF_FLAGS_32)

514 #
515 # Flags used with genoffsets
516 #
517 GENOFFSETS_FLAGS = $(CALLSYMS)
517 GENOFFFLAGS = $(CALLSYMS)
```

```
519 OFFSETS_CREATE = $(GENOFFSETS) -s $(CTFSTABS) -r $(CTFCONVERT) \
520         $(CW) --noecho $(CW_CC_COMPILERS) -- $(GENOFFSETS_FLAGS) \
521         $(CFLAGS) $(CPPFLAGS)
520         $(CW) --noecho $(CW_CC_COMPILERS) -- $(GENOFFFLAGS) $(CFLAGS) $(CPPFLAGS

523 OFFSETS_CREATE64 = $(GENOFFSETS) -s $(CTFSTABS) -r $(CTFCONVERT) \
524         $(CW) --noecho $(CW_CC_COMPILERS) -- $(GENOFFSETS_FLAGS) \
525         $(CFLAGS64) $(CPPFLAGS)
523         $(CW) --noecho $(CW_CC_COMPILERS) -- $(GENOFFFLAGS) $(CFLAGS64) $(CPPFLA

527 #
528 # tradeoff time for space (smaller is better)
529 #
530 sparc_SPACEFLAG         = -xspace -W0,-Lt
531 sparcv9_SPACEFLAG       = -xspace -W0,-Lt
532 i386_SPACEFLAG          = -xspace
533 amd64_SPACEFLAG         =

535 SPACEFLAG               = $($(MACH)_SPACEFLAG)
536 SPACEFLAG64             = $($(MACH64)_SPACEFLAG)

538 #
539 # The Sun Studio 11 compiler has changed the behaviour of integer
540 # wrap arounds and so a flag is needed to use the legacy behaviour
541 # (without this flag panics/hangs could be exposed within the source).
542 #
543 sparc_IROPTFLAG         = -W2,-xwrap_int
544 sparcv9_IROPTFLAG       = -W2,-xwrap_int
545 i386_IROPTFLAG          =
546 amd64_IROPTFLAG         =

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

551 sparc_XREGSFLAG         = -xregs=no%appl
552 sparcv9_XREGSFLAG       = -xregs=no%appl
553 i386_XREGSFLAG          =
554 amd64_XREGSFLAG         =

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

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

564 #
565 # These variables are intended ONLY for use by developers to safely pass extra
566 # flags to the compilers without unintentionally overriding Makefile-set
567 # flags.  They should NEVER be set to any value in a Makefile.
568 #
569 # They come last in the associated FLAGS variable such that they can
570 # explicitly override things if necessary, there are gaps in this, but it's
571 # the best we can manage.
572 #
573 CUSERFLAGS              =
574 CUSERFLAGS64            = $(CUSERFLAGS)
575 CCUSERFLAGS             =
576 CCUSERFLAGS64           = $(CCUSERFLAGS)

578 CSOURCEDEBUGFLAGS       =
579 CCSOURCEDEBUGFLAGS      =
580 $(SRCDBGBLD)CSOURCEDEBUGFLAGS   = $(CCGDEBUG) -xs
581 $(SRCDBGBLD)CCSOURCEDEBUGFLAGS  = $(CCGDEBUG) -xs
```



```
583 CFLAGS=         $(COPTFLAG) $($(MACH)_CFLAGS) $(SPACEFLAG) $(CCMODE) \
584                 $(ILDOFF) $(CERRWARN) $(CSTD) $(CCUNBOUND) $(IROPTFLAG) \
585                 $(CGLOBALSTATIC) $(CCNOAUTOINLINE) $(CCNOREORDER) \
586                 $(CSOURCEDEBUGFLAGS) $(CUSERFLAGS)
587 CFLAGS64=       $(COPTFLAG64) $($(MACH64)_CFLAGS) $(SPACEFLAG64) $(CCMODE64) \
588                 $(ILDOFF) $(CERRWARN) $(CSTD) $(CCUNBOUND) $(IROPTFLAG64) \
589                 $(CGLOBALSTATIC) $(CCNOAUTOINLINE) $(CCNOREORDER) \
590                 $(CSOURCEDEBUGFLAGS) $(CUSERFLAGS64)
591 #
592 # Flags that are used to build parts of the code that are subsequently
593 # run on the build machine (also known as the NATIVE_BUILD).
594 #
595 NATIVE_CFLAGS=  $(COPTFLAG) $($(NATIVE_MACH)_CFLAGS) $(CCMODE) \
596                 $(ILDOFF) $(CERRWARN) $(CSTD) $($(NATIVE_MACH)_CCUNBOUND) \
597                 $(IROPTFLAG) $(CGLOBALSTATIC) $(CCNOAUTOINLINE) \
598                 $(CCNOREORDER) $(CSOURCEDEBUGFLAGS) $(CUSERFLAGS)

600 DTEXTDOM=-DTEXT_DOMAIN=\"$(TEXT_DOMAIN)\"       # For messaging.
601 DTS_ERRNO=-D_TS_ERRNO
602 CPPFLAGS.first= # Please keep empty.  Only lower makefiles should set this.
603 CPPFLAGS.master=$(DTEXTDOM) $(DTS_ERRNO) \
604         $(ENVCPPFLAGS1) $(ENVCPPFLAGS2) $(ENVCPPFLAGS3) $(ENVCPPFLAGS4) \
605         $(ADJUNCT_PROTO:%=-I%/usr/include)
606 CPPFLAGS.native=$(ENVCPPFLAGS1) $(ENVCPPFLAGS2) $(ENVCPPFLAGS3) \
607                 $(ENVCPPFLAGS4) -I$(NATIVE_ADJUNCT)/include
608 CPPFLAGS=       $(CPPFLAGS.first) $(CPPFLAGS.master)
609 AS_CPPFLAGS=    $(CPPFLAGS.first) $(CPPFLAGS.master)
610 JAVAFLAGS=      -source 1.6 -target 1.6 -Xlint:deprecation,-options

612 #
613 # For source message catalogue
614 #
615 .SUFFIXES: $(SUFFIXES) .i .po
616 MSGROOT= $(ROOT)/catalog
617 MSGDOMAIN= $(MSGROOT)/$(TEXT_DOMAIN)
618 MSGDOMAINPOFILE = $(MSGDOMAIN)/$(POFILE)
619 DCMSGDOMAIN= $(MSGROOT)/LC_TIME/$(TEXT_DOMAIN)
620 DCMSGDOMAINPOFILE = $(DCMSGDOMAIN)/$(DCFILE:.dc=.po)

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

634 #
635 # This is overwritten by local Makefile when PROG is a list.
636 #
637 POFILE= $(PROG).po

639 sparc_CCFLAGS=          -cg92 -compat=4 \
640                         -Qoption ccfe -messages=no%anachronism \
641                         $(CCERRWARN)
642 sparcv9_CCFLAGS=        $(sparcv9_XARCH) -dalign -compat=5 \
643                         -Qoption ccfe -messages=no%anachronism \
644                         -Qoption ccfe -features=no%conststrings \
645                         $(CCCREGSYM) \
646                         $(CCERRWARN)
647 i386_CCFLAGS=           -compat=4 \
648                         -Qoption ccfe -messages=no%anachronism \
```

```
649                         -Qoption ccfe -features=no%conststrings \
650                         $(CCERRWARN)
651 amd64_CCFLAGS=          $(amd64_XARCH) -compat=5 \
652                         -Qoption ccfe -messages=no%anachronism \
653                         -Qoption ccfe -features=no%conststrings \
654                         $(CCERRWARN)

656 sparc_CCOPTFLAG=        -O
657 sparcv9_CCOPTFLAG=      -O
658 i386_CCOPTFLAG=         -O
659 amd64_CCOPTFLAG=        -O

661 CCOPTFLAG=      $($(MACH)_CCOPTFLAG)
662 CCOPTFLAG64=    $($(MACH64)_CCOPTFLAG)
663 CCFLAGS=        $(CCOPTFLAG) $($(MACH)_CCFLAGS) $(CCSOURCEDEBUGFLAGS) \
664                 $(CCUSERFLAGS)
665 CCFLAGS64=      $(CCOPTFLAG64) $($(MACH64)_CCFLAGS) $(CCSOURCEDEBUGFLAGS) \
666                 $(CCUSERFLAGS64)

668 #
669 #
670 #
671 ELFWRAP_FLAGS   =
672 ELFWRAP_FLAGS64 =       -64

674 #
675 # Various mapfiles that are used throughout the build, and delivered to
676 # /usr/lib/ld.
677 #
678 MAPFILE.NED_i386 =      $(SRC)/common/mapfiles/common/map.noexdata
679 MAPFILE.NED_sparc =
680 MAPFILE.NED =           $(MAPFILE.NED_$(MACH))
681 MAPFILE.PGA =           $(SRC)/common/mapfiles/common/map.pagealign
682 MAPFILE.NES =           $(SRC)/common/mapfiles/common/map.noexstk
683 MAPFILE.FLT =           $(SRC)/common/mapfiles/common/map.filter
684 MAPFILE.LEX =           $(SRC)/common/mapfiles/common/map.lex.yy

686 #
687 # Generated mapfiles that are compiler specific, and used throughout the
688 # build.  These mapfiles are not delivered in /usr/lib/ld.
689 #
690 MAPFILE.NGB_sparc=      $(SRC)/common/mapfiles/gen/sparc_cc_map.noexeglobs
691 $(__GNUC64)MAPFILE.NGB_sparc= \
692                         $(SRC)/common/mapfiles/gen/sparc_gcc_map.noexeglobs
693 MAPFILE.NGB_sparcv9=    $(SRC)/common/mapfiles/gen/sparcv9_cc_map.noexeglobs
694 $(__GNUC64)MAPFILE.NGB_sparcv9= \
695                         $(SRC)/common/mapfiles/gen/sparcv9_gcc_map.noexeglobs
696 MAPFILE.NGB_i386=       $(SRC)/common/mapfiles/gen/i386_cc_map.noexeglobs
697 $(__GNUC64)MAPFILE.NGB_i386= \
698                         $(SRC)/common/mapfiles/gen/i386_gcc_map.noexeglobs
699 MAPFILE.NGB_amd64=      $(SRC)/common/mapfiles/gen/amd64_cc_map.noexeglobs
700 $(__GNUC64)MAPFILE.NGB_amd64= \
701                         $(SRC)/common/mapfiles/gen/amd64_gcc_map.noexeglobs
702 MAPFILE.NGB =           $(MAPFILE.NGB_$(MACH))

704 #
705 # A generic interface mapfile name, used by various dynamic objects to define
706 # the interfaces and interposers the object must export.
707 #
708 MAPFILE.INT =           mapfile-intf

710 #
711 # LDLIBS32 and LDLIBS64 can be set in the environment to override the following
712 # assignments.
713 #
714 # These environment settings make sure that no libraries are searched outside
```



```
715 # of the local workspace proto area:
716 #       LDLIBS32=-YP,$ROOT/lib:$ROOT/usr/lib
717 #       LDLIBS64=-YP,$ROOT/lib/$MACH64:$ROOT/usr/lib/$MACH64
718 #
719 LDLIBS32 =      $(ENVLDLIBS1) $(ENVLDLIBS2) $(ENVLDLIBS3)
720 LDLIBS32 +=     $(ADJUNCT_PROTO:%=-L%/usr/lib -L%/lib)
721 LDLIBS.cmd =    $(LDLIBS32)
722 LDLIBS.lib =    $(LDLIBS32)

724 LDLIBS64 =      $(ENVLDLIBS1:%=%/$(MACH64)) \
725                 $(ENVLDLIBS2:%=%/$(MACH64)) \
726                 $(ENVLDLIBS3:%=%/$(MACH64))
727 LDLIBS64 +=     $(ADJUNCT_PROTO:%=-L%/usr/lib/$(MACH64) -L%/lib/$(MACH64))

729 #
730 # Define compilation macros.
731 #
732 COMPILE.c=      $(CC) $(CFLAGS) $(CPPFLAGS) -c
733 COMPILE64.c=    $(CC) $(CFLAGS64) $(CPPFLAGS) -c
734 COMPILE.cc=     $(CCC) $(CCFLAGS) $(CPPFLAGS) -c
735 COMPILE64.cc=   $(CCC) $(CCFLAGS64) $(CPPFLAGS) -c
736 COMPILE.s=      $(AS) $(ASFLAGS) $(AS_CPPFLAGS)
737 COMPILE64.s=    $(AS) $(ASFLAGS) $($(MACH64)_AS_XARCH) $(AS_CPPFLAGS)
738 COMPILE.d=      $(DTRACE) -G -32
739 COMPILE64.d=    $(DTRACE) -G -64
740 COMPILE.b=      $(ELFWRAP) $(ELFWRAP_FLAGS$(CLASS))
741 COMPILE64.b=    $(ELFWRAP) $(ELFWRAP_FLAGS$(CLASS))

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

746 #
747 # Link time macros
748 #
749 CCNEEDED                = -lC
750 CCEXTNEEDED             = -lCrun -lCstd
751 $(__GNUC)CCNEEDED       = -L$(GCCLIBDIR) -lstdc++ -lgcc_s
752 $(__GNUC)CCEXTNEEDED    = $(CCNEEDED)

754 LINK.c=         $(CC) $(CFLAGS) $(CPPFLAGS) $(LDFLAGS)
755 LINK64.c=       $(CC) $(CFLAGS64) $(CPPFLAGS) $(LDFLAGS)
756 NORUNPATH=      -norunpath -nolib
757 LINK.cc=        $(CCC) $(CCFLAGS) $(CPPFLAGS) $(NORUNPATH) \
758                 $(LDFLAGS) $(CCNEEDED)
759 LINK64.cc=      $(CCC) $(CCFLAGS64) $(CPPFLAGS) $(NORUNPATH) \
760                 $(LDFLAGS) $(CCNEEDED)

762 #
763 # lint macros
764 #
765 # Note that the undefine of __PRAGMA_REDEFINE_EXTNAME can be removed once
766 # ON is built with a version of lint that has the fix for 4484186.
767 #
768 ALWAYS_LINT_DEFS =      -errtags=yes -s
769 ALWAYS_LINT_DEFS +=     -erroff=E_PTRDIFF_OVERFLOW
770 ALWAYS_LINT_DEFS +=     -erroff=E_ASSIGN_NARROW_CONV
771 ALWAYS_LINT_DEFS +=     -U__PRAGMA_REDEFINE_EXTNAME
772 ALWAYS_LINT_DEFS +=     $(C99LMODE)
773 ALWAYS_LINT_DEFS +=     -errsecurity=$(SECLEVEL)
774 ALWAYS_LINT_DEFS +=     -erroff=E_SEC_CREAT_WITHOUT_EXCL
775 ALWAYS_LINT_DEFS +=     -erroff=E_SEC_FORBIDDEN_WARN_CREAT
776 # XX64 -- really only needed for amd64 lint
777 ALWAYS_LINT_DEFS +=     -erroff=E_ASSIGN_INT_TO_SMALL_INT
778 ALWAYS_LINT_DEFS +=     -erroff=E_CAST_INT_CONST_TO_SMALL_INT
779 ALWAYS_LINT_DEFS +=     -erroff=E_CAST_INT_TO_SMALL_INT
780 ALWAYS_LINT_DEFS +=     -erroff=E_CAST_TO_PTR_FROM_INT
```

```
781 ALWAYS_LINT_DEFS +=     -erroff=E_COMP_INT_WITH_LARGE_INT
782 ALWAYS_LINT_DEFS +=     -erroff=E_INTEGRAL_CONST_EXP_EXPECTED
783 ALWAYS_LINT_DEFS +=     -erroff=E_PASS_INT_TO_SMALL_INT
784 ALWAYS_LINT_DEFS +=     -erroff=E_PTR_CONV_LOSES_BITS

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

790 SECLEVEL=       core
791 LINT.c=         $(LINT) $(ONLY_LINT_DEFS) $(LINTFLAGS) $(CPPFLAGS) \
792                 $(ALWAYS_LINT_DEFS)
793 LINT64.c=       $(LINT) $(ONLY_LINT_DEFS) $(LINTFLAGS64) $(CPPFLAGS) \
794                 $(ALWAYS_LINT_DEFS)
795 LINT.s=         $(LINT.c)

797 # For some future builds, NATIVE_MACH and MACH might be different.
798 # Therefore, NATIVE_MACH needs to be redefined in the
799 # environment as ‘uname -p‘ to override this macro.
800 #
801 # For now at least, we cross-compile amd64 on i386 machines.
802 NATIVE_MACH=    $(MACH:amd64=i386)

804 # Define native compilation macros
805 #

807 # Base directory where compilers are loaded.
808 # Defined here so it can be overridden by developer.
809 #
810 SPRO_ROOT=              $(BUILD_TOOLS)/SUNWspro
811 SPRO_VROOT=             $(SPRO_ROOT)/SS12
812 GNU_ROOT=               /usr

814 $(__GNUC)PRIMARY_CC=    gcc4,$(GNUC_ROOT)/bin/gcc,gnu
815 $(__SUNC)PRIMARY_CC=    studio12,$(SPRO_VROOT)/bin/cc,sun
816 $(__GNUC)PRIMARY_CCC=   gcc4,$(GNUC_ROOT)/bin/g++,gnu
817 $(__SUNC)PRIMARY_CCC=   studio12,$(SPRO_VROOT)/bin/CC,sun

819 CW_CC_COMPILERS=        $(PRIMARY_CC:%=--primary %) $(SHADOW_CCS:%=--shadow %)
820 CW_CCC_COMPILERS=      $(PRIMARY_CCC:%=--primary %) $(SHADOW_CCCS:%=--shadow %)


823 # Till SS12u1 formally becomes the NV CBE, LINT is hard
824 # coded to be picked up from the $SPRO_ROOT/sunstudio12.1/
825 # location. Impacted variables are sparc_LINT, sparcv9_LINT,
826 # i386_LINT, amd64_LINT.
827 # Reset them when SS12u1 is rolled out.
828 #

830 # Specify platform compiler versions for languages
831 # that we use (currently only c and c++).
832 #
833 CW=                     $(ONBLD_TOOLS)/bin/$(MACH)/cw

835 BUILD_CC=               $(CW) $(CW_CC_COMPILERS) --
836 BUILD_CCC=              $(CW) -C $(CW_CCC_COMPILERS) --
837 BUILD_CPP=              /usr/ccs/lib/cpp
838 BUILD_LD=               /usr/ccs/bin/ld
839 BUILD_LINT=             $(SPRO_ROOT)/sunstudio12.1/bin/lint

841 $(MACH)_CC=             $(BUILD_CC)
842 $(MACH)_CCC=            $(BUILD_CCC)
843 $(MACH)_CPP=            $(BUILD_CPP)
844 $(MACH)_LD=             $(BUILD_LD)
845 $(MACH)_LINT=           $(BUILD_LINT)
846 $(MACH64)_CC=           $(BUILD_CC)
```



```
847 $(MACH64)_CCC=          $(BUILD_CCC)
848 $(MACH64)_CPP=          $(BUILD_CPP)
849 $(MACH64)_LD=           $(BUILD_LD)
850 $(MACH64)_LINT=         $(BUILD_LINT)

852 sparc_AS=              /usr/ccs/bin/as -xregsym=no
853 sparcv9_AS=            $($(MACH)_AS)

855 i386_AS=               /usr/ccs/bin/as
856 $(__GNUC)i386_AS=      $(ONBLD_TOOLS)/bin/$(MACH)/aw
857 amd64_AS=              $(ONBLD_TOOLS)/bin/$(MACH)/aw

859 NATIVECC=             $($(NATIVE_MACH)_CC)
860 NATIVECCC=            $($(NATIVE_MACH)_CCC)
861 NATIVECPP=            $($(NATIVE_MACH)_CPP)
862 NATIVEAS=             $($(NATIVE_MACH)_AS)
863 NATIVELD=             $($(NATIVE_MACH)_LD)
864 NATIVELINT=           $($(NATIVE_MACH)_LINT)

866 #
867 # Makefile.master.64 overrides these settings
868 #
869 CC=                    $(NATIVECC)
870 CCC=                   $(NATIVECCC)
871 CPP=                   $(NATIVECPP)
872 AS=                    $(NATIVEAS)
873 LD=                    $(NATIVELD)
874 LINT=                  $(NATIVELINT)

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

879 BDIRECT=        -Bdirect
880 BDYNAMIC=       -Bdynamic
881 BLOCAL=         -Blocal
882 BNODIRECT=      -Bnodirect
883 BREDUCE=        -Breduce
884 BSTATIC=        -Bstatic

886 ZDEFS=          -zdefs
887 ZDIRECT=        -zdirect
888 ZIGNORE=        -zignore
889 ZINITFIRST=     -zinitfirst
890 ZINTERPOSE=     -zinterpose
891 ZLAZYLOAD=      -zlazyload
892 ZLOADFLTR=      -zloadfltr
893 ZMULDEFS=       -zmuldefs
894 ZNODEFAULTLIB=  -znodefaultlib
895 ZNODEFS=        -znodefs
896 ZNODELETE=      -znodelete
897 ZNODLOPEN=      -znodlopen
898 ZNODUMP=        -znodump
899 ZNOLAZYLOAD=    -znolazyload
900 ZNOLDYNSYM=     -znoldynsym
901 ZNORELOC=       -znoreloc
902 ZNOVERSION=     -znoversion
903 ZRECORD=        -zrecord
904 ZREDLOCSYM=     -zredlocsym
905 ZTEXT=          -ztext
906 ZVERBOSE=       -zverbose

908 GSHARED=        -G
909 CCMT=           -mt

911 # Handle different PIC models on different ISAs
912 # (May be overridden by lower-level Makefiles)
```

```
914 sparc_C_PICFLAGS =      -fpic
915 sparcv9_C_PICFLAGS =    -fpic
916 i386_C_PICFLAGS =       -fpic
917 amd64_C_PICFLAGS =      -fpic
918 C_PICFLAGS =            $($(MACH)_C_PICFLAGS)
919 C_PICFLAGS64 =          $($(MACH64)_C_PICFLAGS)

921 sparc_C_BIGPICFLAGS =   -fPIC
922 sparcv9_C_BIGPICFLAGS = -fPIC
923 i386_C_BIGPICFLAGS =    -fPIC
924 amd64_C_BIGPICFLAGS =   -fPIC
925 C_BIGPICFLAGS =         $($(MACH)_C_BIGPICFLAGS)
926 C_BIGPICFLAGS64 =       $($(MACH64)_C_BIGPICFLAGS)

928 # CC requires there to be no space between '-K' and 'pic' or 'PIC'.
929 # and does not support -f
930 sparc_CC_PICFLAGS =     -_cc=-Kpic -_gcc=-fpic
931 sparcv9_CC_PICFLAGS =   -_cc=-KPIC -_gcc=-fPIC
932 i386_CC_PICFLAGS =      -_cc=-Kpic -_gcc=-fpic
933 amd64_CC_PICFLAGS =     -_cc=-Kpic -_gcc=-fpic
934 CC_PICFLAGS =           $($(MACH)_CC_PICFLAGS)
935 CC_PICFLAGS64 =         $($(MACH64)_CC_PICFLAGS)

937 AS_PICFLAGS=            -K pic
938 AS_BIGPICFLAGS=         -K PIC

940 #
941 # Default label for CTF sections
942 #
943 CTFCVTFLAGS=            -i -L VERSION

945 #
946 # Override to pass module-specific flags to ctfmerge.  Currently used only by
947 # krtld to turn on fuzzy matching, and source-level debugging to inhibit
948 # stripping.
949 #
950 CTFMRGFLAGS=

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

954 # Rules (normally from make.rules) and macros which are used for post
955 # processing files. Normally, these do stripping of the comment section
956 # automatically.
957 #    RELEASE_CM:        Should be editted to reflect the release.
958 #    POST_PROCESS_O:    Post-processing for '.o' files.
959 #    POST_PROCESS_A:    Post-processing for '.a' files (currently null).
960 #    POST_PROCESS_SO:   Post-processing for '.so' files.
961 #    POST_PROCESS:      Post-processing for executable files (no suffix).
962 # Note that these macros are not completely generalized as they are to be
963 # used with the file name to be processed following.
964 #
965 # It is left as an exercise to Release Engineering to embellish the generation
966 # of the release comment string.
967 #
968 #       If this is a standard development build:
969 #               compress the comment section (mcs -c)
970 #               add the standard comment (mcs -a $(RELEASE_CM))
971 #               add the development specific comment (mcs -a $(DEV_CM))
972 #
973 #       If this is an installation build:
974 #               delete the comment section (mcs -d)
975 #               add the standard comment (mcs -a $(RELEASE_CM))
976 #               add the development specific comment (mcs -a $(DEV_CM))
977 #
978 #       If this is an release build:
```



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

1006 PROCESS_COMMENT=  @?${MCS} -d -a $(RELEASE_CM) -a $(DEV_CM)
1007 $(RELEASE_BUILD)PROCESS_COMMENT=   @?${MCS} -d -a $(RELEASE_CM)

1009 STRIP_STABS=                       $(STRIP) -x $@
1010 $(SRCDBGBLD)STRIP_STABS=           :

1012 POST_PROCESS_O=
1013 POST_PROCESS_A=
1014 POST_PROCESS_SO=       $(PROCESS_COMMENT) $@ ; $(STRIP_STABS) ; \
1015                        $(ELFSIGN_OBJECT)
1016 POST_PROCESS=          $(PROCESS_COMMENT) $@ ; $(STRIP_STABS) ; \
1017                        $(ELFSIGN_OBJECT)

1019 #
1020 # chk4ubin is a tool that inspects a module for a symbol table
1021 # ELF section size which can trigger an OBP bug on older platforms.
1022 # This problem affects only specific sun4u bootable modules.
1023 #
1024 CHK4UBIN=       $(ONBLD_TOOLS)/bin/$(MACH)/chk4ubin
1025 CHK4UBINFLAGS=
1026 CHK4UBINARY=    $(CHK4UBIN) $(CHK4UBINFLAGS) $@

1028 #
1029 # PKGARCHIVE specifies the default location where packages should be
1030 # placed if built.
1031 #
1032 $(RELEASE_BUILD)PKGARCHIVESUFFIX=       -nd
1033 PKGARCHIVE=$(SRC)/../../packages/$(MACH)/nightly$(PKGARCHIVESUFFIX)

1035 #
1036 # The repositories will be created with these publisher settings.  To
1037 # update an image to the resulting repositories, this must match the
1038 # publisher name provided to "pkg set-publisher."
1039 #
1040 PKGPUBLISHER_REDIST=    on-nightly
1041 PKGPUBLISHER_NONREDIST= on-extra

1043 #       Default build rules which perform comment section post-processing.
1044 #
```

```
.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) $@; $(SED) -e "1s:^\#!@PYTHON@:\#!$(PYTHON):" < $< > $@; $(CHMOD)

.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  > $@
	$(RM) lex.yy.c

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

.cc.po:
	$(CPPFORPO) $< > $<.i
	$(BUILD.po)
```

```
1178 .y.po:
1179         $(YACC) -d $<
1180         $(CPPFORPO) y.tab.c  > $<.i
1181         $(BUILD.po)
1182         $(RM) y.tab.c

1184 .l.po:
1185         $(LEX) $<
1186         $(CPPFORPO) lex.yy.c  > $<.i
1187         $(BUILD.po)
1188         $(RM) lex.yy.c

1190 #
1191 # Rules to perform stylistic checks
1192 #
1193 .SUFFIXES: .x .xml .check .xmlchk

1195 .h.check:
1196         $(DOT_H_CHECK)

1198 .x.check:
1199         $(DOT_X_CHECK)

1201 .xml.xmlchk:
1202         $(MANIFEST_CHECK)

1204 #
1205 # Include rules to render automated sccs get rules "safe".
1206 #
1207 include $(SRC)/Makefile.noget
```

```
**********************************************************
    1784 Mon Dec 10 01:35:47 2018
new/usr/src/cmd/hal/addons/cpufreq/Makefile
Code review comments
**********************************************************
______unchanged_portion_omitted_
```