```
**********************************************************
   34728 Thu Aug 15 11:59:59 2013
new/usr/src/Makefile.master
4029 remove tonic build bits
**********************************************************
```

```
#
# 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.
#
# 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)

# 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)

# 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

JAVA_ROOT=      /usr/java

SFW_ROOT=       /usr/sfw
SFWINCDIR=      $(SFW_ROOT)/include
SFWLIBDIR=      $(SFW_ROOT)/lib
SFWLIBDIR64=    $(SFW_ROOT)/lib/$(MACH64)

GCC_ROOT=       /opt/gcc/4.4.4
GCCLIBDIR=      $(GCC_ROOT)/lib
GCCLIBDIR64=    $(GCC_ROOT)/lib/$(MACH64)
```

```
118 DOCBOOK_XSL_ROOT=       /usr/share/sgml/docbook/xsl-stylesheets

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

179 FILEMODE=       644
180 DIRMODE=        755

182 #
183 # The version of the patch makeup table optimized for build-time use.  Used
```



```
184 # during patch builds only.
185 $(PATCH_BUILD)PMTMO_FILE=$(SRC)/patch_makeup_table.mo

187 # Declare that nothing should be built in parallel.
188 # Individual Makefiles can use the .PARALLEL target to declare otherwise.
189 .NO_PARALLEL:

191 # For stylistic checks
192 #
193 # Note that the X and C checks are not used at this time and may need
194 # modification when they are actually used.
195 #
196 CSTYLE=         $(ONBLD_TOOLS)/bin/cstyle
197 CSTYLE_TAIL=
198 HDRCHK=         $(ONBLD_TOOLS)/bin/hdrchk
199 HDRCHK_TAIL=
200 JSTYLE=         $(ONBLD_TOOLS)/bin/jstyle

202 DOT_H_CHECK=    \
203         @$(ECHO) "checking $<"; $(CSTYLE) $< $(CSTYLE_TAIL); \
204         $(HDRCHK) $< $(HDRCHK_TAIL)

206 DOT_X_CHECK=    \
207         @$(ECHO) "checking $<"; $(RPCGEN) -C -h $< | $(CSTYLE) $(CSTYLE_TAIL); \
208         $(RPCGEN) -C -h $< | $(HDRCHK) $< $(HDRCHK_TAIL)

210 DOT_C_CHECK=    \
211         @$(ECHO) "checking $<"; $(CSTYLE) $< $(CSTYLE_TAIL)

213 MANIFEST_CHECK= \
214         @$(ECHO) "checking $<"; \
215         SVCCFG_DTD=$(SRC)/cmd/svc/dtd/service_bundle.dtd.1 \
216         SVCCFG_REPOSITORY=$(SRC)/cmd/svc/seed/global.db \
217         SVCCFG_CONFIGD_PATH=$(SRC)/cmd/svc/configd/svc.configd-native \
218         $(SRC)/cmd/svc/svccfg/svccfg-native validate $<

230 #
231 # IMPORTANT:: If you change any of INS.file, INS.dir, INS.rename,
232 # INS.link or INS.symlink here, then you must also change the
233 # corresponding override definitions in $CLOSED/Makefile.tonic.
234 # If you do not do this, then the closedbins build for the OpenSolaris
235 # community will break. PS, the gatekeepers will be upset too.
220 INS.file=       $(RM) $@; $(INS) -s -m $(FILEMODE) -f $(@D) $<
221 INS.dir=        $(INS) -s -d -m $(DIRMODE) $@
222 # installs and renames at once
223 #
224 INS.rename=     $(INS.file); $(MV) $(@D)/$(<F) $@

226 # install a link
227 INSLINKTARGET=  $<
228 INS.link=       $(RM) $@; $(LN) $(INSLINKTARGET) $@
229 INS.symlink=    $(RM) $@; $(SYMLINK) $(INSLINKTARGET) $@

231 #
232 # Python bakes the mtime of the .py file into the compiled .pyc and
233 # rebuilds if the baked-in mtime != the mtime of the source file
234 # (rather than only if it’s less than), thus when installing python
235 # files we must make certain to not adjust the mtime of the source
236 # (.py) file.
237 #
238 INS.pyfile=     $(INS.file); $(TOUCH) -r $< $@

240 # MACH must be set in the shell environment per uname -p on the build host
241 # More specific architecture variables should be set in lower makefiles.
242 #
243 # MACH64 is derived from MACH, and BUILD64 is set to ‘#’ for
```

```
244 # architectures on which we do not build 64-bit versions.
245 # (There are no such architectures at the moment.)
246 #
247 # Set BUILD64=# in the environment to disable 64-bit amd64
248 # builds on i386 machines.

250 MACH64_1=       $(MACH:sparc=sparcv9)
251 MACH64=         $(MACH64_1:i386=amd64)

253 MACH32_1=       $(MACH:sparc=sparcv7)
254 MACH32=         $(MACH32_1:i386=i86)

256 sparc_BUILD64=
257 i386_BUILD64=
258 BUILD64=        $($(MACH)_BUILD64)

260 #
261 # C compiler mode. Future compilers may change the default on us,
262 # so force extended ANSI mode globally. Lower level makefiles can
263 # override this by setting CCMODE.
264 #
265 CCMODE=                 -Xa
266 CCMODE64=               -Xa

268 #
269 # C compiler verbose mode. This is so we can enable it globally,
270 # but turn it off in the lower level makefiles of things we cannot
271 # (or aren't going to) fix.
272 #
273 CCVERBOSE=              -v

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

279 # set this to the secret flag "-Wc,-Qiselect-regsym=0" to disable register
280 # symbols (used to detect conflicts between objects that use global registers)
281 # we disable this now for safety, and because genunix doesn't link with
282 # this feature (the v9 default) enabled.
283 #
284 # REGSYM is separate since the C++ driver syntax is different.
285 CCREGSYM=               -Wc,-Qiselect-regsym=0
286 CCCREGSYM=              -Qoption cg -Qiselect-regsym=0

288 # Prevent the removal of static symbols by the SPARC code generator (cg).
289 # The x86 code generator (ube) does not remove such symbols and as such
290 # using this workaround is not applicable for x86.
291 #
292 CCSTATICSYM=            -Wc,-Qassembler-ounrefsym=0
293 #
294 # generate 32-bit addresses in the v9 kernel. Saves memory.
295 CCABS32=                -Wc,-xcode=abs32
296 #
297 # generate v9 code which tolerates callers using the v7 ABI, for the sake of
298 # system calls.
299 CC32BITCALLERS=         -_gcc=-massume-32bit-callers

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



```
310 # One optimization the compiler might perform is to turn this:
311 #       #pragma weak foo
312 #       extern int foo;
313 #       if (&foo)
314 #               foo = 5;
315 # into
316 #       foo = 5;
317 # Since we do some of this (foo might be referenced in common kernel code
318 # but provided only for some cpu modules or platforms), we disable this
319 # optimization.
320 #
321 sparc_CCUNBOUND = -Wd,-xsafe=unboundsym
322 i386_CCUNBOUND  =
323 CCUNBOUND       = $($(MACH)_CCUNBOUND)

325 #
326 # compiler '-xarch' flag. This is here to centralize it and make it
327 # overridable for testing.
328 sparc_XARCH=    -m32
329 sparcv9_XARCH=  -m64
330 i386_XARCH=
331 amd64_XARCH=    -m64 -Ui386 -U__i386

333 # assembler '-xarch' flag.  Different from compiler '-xarch' flag.
334 sparc_AS_XARCH=         -xarch=v8plus
335 sparcv9_AS_XARCH=       -xarch=v9
336 i386_AS_XARCH=
337 amd64_AS_XARCH=         -xarch=amd64 -P -Ui386 -U__i386

339 #
340 # These flags define what we need to be 'standalone' i.e. -not- part
341 # of the rather more cosy userland environment.  This basically means
342 # the kernel.
343 #
344 # XX64  future versions of gcc will make -mcmodel=kernel imply -mno-red-zone
345 #
346 sparc_STAND_FLAGS=      -_gcc=-ffreestanding
347 sparcv9_STAND_FLAGS=    -_gcc=-ffreestanding
348 # Disabling MMX also disables 3DNow, disabling SSE also disables all later
349 # additions to SSE (SSE2, AVX ,etc.)
350 NO_SIMD=                -_gcc=-mno-mmx -_gcc=-mno-sse
351 i386_STAND_FLAGS=       -_gcc=-ffreestanding $(NO_SIMD)
352 amd64_STAND_FLAGS=      -xmodel=kernel $(NO_SIMD)

354 SAVEARGS=               -Wu,-save_args
355 amd64_STAND_FLAGS       += $(SAVEARGS)

357 STAND_FLAGS_32 = $($(MACH)_STAND_FLAGS)
358 STAND_FLAGS_64 = $($(MACH64)_STAND_FLAGS)

360 #
361 # disable the incremental linker
362 ILDOFF=                 -xildoff
363 #
364 XDEPEND=                -xdepend
365 XFFLAG=                 -xF=%all
366 XESS=                   -xs
367 XSTRCONST=              -xstrconst

369 #
370 # turn warnings into errors (C)
371 CERRWARN = -erttags=yes -errwarn=%all
372 CERRWARN += -erroff=E_EMPTY_TRANSLATION_UNIT
373 CERRWARN += -erroff=E_STATEMENT_NOT_REACHED

375 CERRWARN += -_gcc=-Wno-missing-braces
```

```
376 CERRWARN += -_gcc=-Wno-sign-compare
377 CERRWARN += -_gcc=-Wno-unknown-pragmas
378 CERRWARN += -_gcc=-Wno-unused-parameter
379 CERRWARN += -_gcc=-Wno-missing-field-initializers

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

384 # DEBUG v. -nd make for frequent unused variables, empty conditions, etc. in
385 # -nd builds
386 $(RELEASE_BUILD)CERRWARN += -_gcc=-Wno-unused
387 $(RELEASE_BUILD)CERRWARN += -_gcc=-Wno-empty-body

389 #
390 # turn warnings into errors (C++)
391 CCERRWARN=              -xwe

393 # C99 mode
394 C99_ENABLE=     -xc99=%all
395 C99_DISABLE=    -xc99=%none
396 C99MODE=        $(C99_DISABLE)
397 C99LMODE=       $(C99MODE:-xc99%=-Xc99%)

399 # In most places, assignments to these macros should be appended with +=
400 # (CPPFLAGS.master allows values to be prepended to CPPFLAGS).
401 sparc_CFLAGS=   $(sparc_XARCH) $(CCSTATICSYM)
402 sparcv9_CFLAGS= $(sparcv9_XARCH) -dalign $(CCVERBOSE) $(V9ABIWARN) $(CCREGSYM) \
403                 $(CCSTATICSYM)
404 i386_CFLAGS=    $(i386_XARCH)
405 amd64_CFLAGS=   $(amd64_XARCH)

407 sparc_ASFLAGS=  $(sparc_AS_XARCH)
408 sparcv9_ASFLAGS=$(sparcv9_AS_XARCH)
409 i386_ASFLAGS=   $(i386_AS_XARCH)
410 amd64_ASFLAGS=  $(amd64_AS_XARCH)

412 #
413 sparc_COPTFLAG=         -xO3
414 sparcv9_COPTFLAG=       -xO3
415 i386_COPTFLAG=          -O
416 amd64_COPTFLAG=         -xO3

418 COPTFLAG= $($(MACH)_COPTFLAG)
419 COPTFLAG64= $($(MACH64)_COPTFLAG)

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

425 # Direct the Sun Studio compiler to use a static globalization prefix based on t
426 # name of the module rather than something unique. Otherwise, objects
427 # will not build deterministically, as subsequent compilations of identical
428 # source will yeild objects that always look different.
429 #
430 # In the same spirit, this will also remove the date from the N_OPT stab.
431 CGLOBALSTATIC= -W0,-xglobalstatic

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

437 #
438 # Default debug format for Sun Studio 11 is dwarf, so force it to
439 # generate stabs.
440 #
441 DEBUGFORMAT=    -xdebugformat=stabs
```



```
443 #
444 # Flags used to build in debug mode for ctf generation.  Bugs in the Devpro
445 # compilers currently prevent us from building with cc-emitted DWARF.
446 #
447 CTF_FLAGS_sparc = -g -Wc,-Qiselect-T1 $(C99MODE) $(CNOGLOBAL) $(CDWARFSTR)
448 CTF_FLAGS_i386  = -g $(C99MODE) $(CNOGLOBAL) $(CDWARFSTR)

450 CTF_FLAGS_sparcv9       = $(CTF_FLAGS_sparc)
451 CTF_FLAGS_amd64         = $(CTF_FLAGS_i386)

453 # Sun Studio produces broken userland code when saving arguments.
454 $(__GNUC)CTF_FLAGS_amd64 += $(SAVEARGS)

456 CTF_FLAGS_32    = $(CTF_FLAGS_$(MACH)) $(DEBUGFORMAT)
457 CTF_FLAGS_64    = $(CTF_FLAGS_$(MACH64)) $(DEBUGFORMAT)
458 CTF_FLAGS       = $(CTF_FLAGS_32)

460 #
461 # Flags used with genoffsets
462 #
463 GOFLAGS = -_noecho \
464         $(CALLSYMS) \
465         $(CDWARFSTR)

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

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

473 #
474 # tradeoff time for space (smaller is better)
475 #
476 sparc_SPACEFLAG         = -xspace -W0,-Lt
477 sparcv9_SPACEFLAG       = -xspace -W0,-Lt
478 i386_SPACEFLAG          = -xspace
479 amd64_SPACEFLAG         =

481 SPACEFLAG               = $($(MACH)_SPACEFLAG)
482 SPACEFLAG64             = $($(MACH64)_SPACEFLAG)

484 #
485 # The Sun Studio 11 compiler has changed the behaviour of integer
486 # wrap arounds and so a flag is needed to use the legacy behaviour
487 # (without this flag panics/hangs could be exposed within the source).
488 #
489 sparc_IROPTFLAG         = -W2,-xwrap_int
490 sparcv9_IROPTFLAG       = -W2,-xwrap_int
491 i386_IROPTFLAG          =
492 amd64_IROPTFLAG         =

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

497 sparc_XREGSFLAG         = -xregs=no%appl
498 sparcv9_XREGSFLAG       = -xregs=no%appl
499 i386_XREGSFLAG          =
500 amd64_XREGSFLAG         =

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

505 # dmake SOURCEDEBUG=yes ... enables source-level debugging information, and
506 # avoids stripping it.
507 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

CFLAGS=         $(COPTFLAG) $($(MACH)_CFLAGS) $(SPACEFLAG) $(CCMODE) \
                $(ILDOFF) $(CERRWARN) $(C99MODE) $(CCUNBOUND) $(IROPTFLAG) \
                $(CGLOBALSTATIC) $(CCNOAUTOINLINE) $(CSOURCEDEBUGFLAGS) \
                $(CUSERFLAGS)
CFLAGS64=       $(COPTFLAG64) $($(MACH64)_CFLAGS) $(SPACEFLAG64) $(CCMODE64) \
                $(ILDOFF) $(CERRWARN) $(C99MODE) $(CCUNBOUND) $(IROPTFLAG64) \
                $(CGLOBALSTATIC) $(CCNOAUTOINLINE) $(CSOURCEDEBUGFLAGS) \
                $(CUSERFLAGS64)
#
# 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)

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)

#
#
#
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
```

```
640 $(__GNUC64)MAPFILE.NGB_i386= \
641                         $(SRC)/common/mapfiles/gen/i386_gcc_map.noexeglobs
642 MAPFILE.NGB_amd64=      $(SRC)/common/mapfiles/gen/amd64_cc_map.noexeglobs
643 $(__GNUC64)MAPFILE.NGB_amd64= \
644                         $(SRC)/common/mapfiles/gen/amd64_gcc_map.noexeglobs
645 MAPFILE.NGB =           $(MAPFILE.NGB_$(MACH))

647 #
648 # A generic interface mapfile name, used by various dynamic objects to define
649 # the interfaces and interposers the object must export.
650 #
651 MAPFILE.INT =           mapfile-intf

653 #
654 # LDLIBS32 can be set in the environment to override the following assignment.
655 # LDLIBS64 can be set to override the assignment made in Makefile.master.64.
656 # These environment settings make sure that no libraries are searched outside
657 # of the local workspace proto area:
658 #       LDLIBS32=-YP,$ROOT/lib:$ROOT/usr/lib
659 #       LDLIBS64=-YP,$ROOT/lib/$MACH64:$ROOT/usr/lib/$MACH64
660 #
661 LDLIBS32 =      $(ENVLDLIBS1) $(ENVLDLIBS2) $(ENVLDLIBS3)
662 LDLIBS.cmd =    $(LDLIBS32)
663 LDLIBS.lib =    $(LDLIBS32)
664 #
665 # Define compilation macros.
666 #
667 COMPILE.c=      $(CC) $(CFLAGS) $(CPPFLAGS) -c
668 COMPILE64.c=    $(CC) $(CFLAGS64) $(CPPFLAGS) -c
669 COMPILE.cc=     $(CCC) $(CCFLAGS) $(CPPFLAGS) -c
670 COMPILE64.cc=   $(CCC) $(CCFLAGS64) $(CPPFLAGS) -c
671 COMPILE.s=      $(AS) $(ASFLAGS) $(AS_CPPFLAGS)
672 COMPILE64.s=    $(AS) $(ASFLAGS) $($(MACH64)_AS_XARCH) $(AS_CPPFLAGS)
673 COMPILE.d=      $(DTRACE) -G -32
674 COMPILE64.d=    $(DTRACE) -G -64
675 COMPILE.b=      $(ELFWRAP) $(ELFWRAP_FLAGS$(CLASS))
676 COMPILE64.b=    $(ELFWRAP) $(ELFWRAP_FLAGS$(CLASS))

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

681 #
682 # Link time macros
683 #
684 CCNEEDED                = -lC
685 CCEXTNEEDED             = -lCrun -lCstd
686 $(__GNUC)CCNEEDED       = -L$(GCCLIBDIR) -R$(GCCLIBDIR) -lstdc++ -lgcc_s
687 $(__GNUC)CCEXTNEEDED    = $(CCNEEDED)

689 LINK.c=         $(CC) $(CFLAGS) $(CPPFLAGS) $(LDFLAGS)
690 LINK64.c=       $(CC) $(CFLAGS64) $(CPPFLAGS) $(LDFLAGS)
691 NORUNPATH=      -norunpath -nolib
692 LINK.cc=        $(CCC) $(CCFLAGS) $(CPPFLAGS) $(NORUNPATH) \
693                 $(LDFLAGS) $(CCNEEDED)
694 LINK64.cc=      $(CCC) $(CCFLAGS64) $(CPPFLAGS) $(NORUNPATH) \
695                 $(LDFLAGS) $(CCNEEDED)

697 #
698 # lint macros
699 #
700 # Note that the undefine of __PRAGMA_REDEFINE_EXTNAME can be removed once
701 # ON is built with a version of lint that has the fix for 4484186.
702 #
703 ALWAYS_LINT_DEFS =      -errtags=yes -s
704 ALWAYS_LINT_DEFS +=     -erroff=E_PTRDIFF_OVERFLOW
705 ALWAYS_LINT_DEFS +=     -erroff=E_ASSIGN_NARROW_CONV
```



```
706 ALWAYS_LINT_DEFS +=     -U__PRAGMA_REDEFINE_EXTNAME
707 ALWAYS_LINT_DEFS +=     $(C99LMODE)
708 ALWAYS_LINT_DEFS +=     -errsecurity=$(SECLEVEL)
709 ALWAYS_LINT_DEFS +=     -erroff=E_SEC_CREAT_WITHOUT_EXCL
710 ALWAYS_LINT_DEFS +=     -erroff=E_SEC_FORBIDDEN_WARN_CREAT
711 # XX64 -- really only needed for amd64 lint
712 ALWAYS_LINT_DEFS +=     -erroff=E_ASSIGN_INT_TO_SMALL_INT
713 ALWAYS_LINT_DEFS +=     -erroff=E_CAST_INT_CONST_TO_SMALL_INT
714 ALWAYS_LINT_DEFS +=     -erroff=E_CAST_INT_TO_SMALL_INT
715 ALWAYS_LINT_DEFS +=     -erroff=E_CAST_TO_PTR_FROM_INT
716 ALWAYS_LINT_DEFS +=     -erroff=E_COMP_INT_WITH_LARGE_INT
717 ALWAYS_LINT_DEFS +=     -erroff=E_INTEGRAL_CONST_EXP_EXPECTED
718 ALWAYS_LINT_DEFS +=     -erroff=E_PASS_INT_TO_SMALL_INT
719 ALWAYS_LINT_DEFS +=     -erroff=E_PTR_CONV_LOSES_BITS

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

725 SECLEVEL=       core
726 LINT.c=         $(LINT) $(ONLY_LINT_DEFS) $(LINTFLAGS) $(CPPFLAGS) \
727                 $(ALWAYS_LINT_DEFS)
728 LINT64.c=       $(LINT) $(ONLY_LINT_DEFS) $(LINTFLAGS64) $(CPPFLAGS) \
729                 $(ALWAYS_LINT_DEFS)
730 LINT.s=         $(LINT.c)

732 # For some future builds, NATIVE_MACH and MACH might be different.
733 # Therefore, NATIVE_MACH needs to be redefined in the
734 # environment as ‘uname -p‘ to override this macro.
735 #
736 # For now at least, we cross-compile amd64 on i386 machines.
737 NATIVE_MACH=    $(MACH:amd64=i386)

739 # Define native compilation macros
740 #

742 # Base directory where compilers are loaded.
743 # Defined here so it can be overridden by developer.
744 #
745 SPRO_ROOT=              $(BUILD_TOOLS)/SUNWspro
746 SPRO_VROOT=             $(SPRO_ROOT)/SS12
747 GNU_ROOT=               $(SFW_ROOT)

749 # Till SS12u1 formally becomes the NV CBE, LINT is hard
750 # coded to be picked up from the $SPRO_ROOT/sunstudio12.1/
751 # location. Impacted variables are sparc_LINT, sparcv9_LINT,
752 # i386_LINT, amd64_LINT.
753 # Reset them when SS12u1 is rolled out.
754 #

756 # Specify platform compiler versions for languages
757 # that we use (currently only c and c++).
758 #
759 sparc_CC=               $(ONBLD_TOOLS)/bin/$(MACH)/cw -_cc
760 $(__GNUC)sparc_CC=      $(ONBLD_TOOLS)/bin/$(MACH)/cw -_gcc
761 sparc_CCC=              $(ONBLD_TOOLS)/bin/$(MACH)/cw -_CC
762 $(__GNUC)sparc_CCC=     $(ONBLD_TOOLS)/bin/$(MACH)/cw -_g++
763 sparc_CPP=              /usr/ccs/lib/cpp
764 sparc_AS=               /usr/ccs/bin/as -xregsym=no
765 sparc_LD=               /usr/ccs/bin/ld
766 sparc_LINT=             $(SPRO_ROOT)/sunstudio12.1/bin/lint

768 sparcv9_CC=             $(ONBLD_TOOLS)/bin/$(MACH)/cw -_cc
769 $(__GNUC64)sparcv9_CC=  $(ONBLD_TOOLS)/bin/$(MACH)/cw -_gcc
770 sparcv9_CCC=            $(ONBLD_TOOLS)/bin/$(MACH)/cw -_CC
771 $(__GNUC64)sparcv9_CCC= $(ONBLD_TOOLS)/bin/$(MACH)/cw -_g++
```

```
772 sparcv9_CPP=            /usr/ccs/lib/cpp
773 sparcv9_AS=             /usr/ccs/bin/as -xregsym=no
774 sparcv9_LD=             /usr/ccs/bin/ld
775 sparcv9_LINT=           $(SPRO_ROOT)/sunstudio12.1/bin/lint

777 i386_CC=               $(ONBLD_TOOLS)/bin/$(MACH)/cw -_cc
778 $(__GNUC)i386_CC=      $(ONBLD_TOOLS)/bin/$(MACH)/cw -_gcc
779 i386_CCC=              $(ONBLD_TOOLS)/bin/$(MACH)/cw -_CC
780 $(__GNUC)i386_CCC=     $(ONBLD_TOOLS)/bin/$(MACH)/cw -_g++
781 i386_CPP=              /usr/ccs/lib/cpp
782 i386_AS=               /usr/ccs/bin/as
783 $(__GNUC)i386_AS=      $(ONBLD_TOOLS)/bin/$(MACH)/aw
784 i386_LD=               /usr/ccs/bin/ld
785 i386_LINT=             $(SPRO_ROOT)/sunstudio12.1/bin/lint

787 amd64_CC=              $(ONBLD_TOOLS)/bin/$(MACH)/cw -_cc
788 $(__GNUC64)amd64_CC=   $(ONBLD_TOOLS)/bin/$(MACH)/cw -_gcc
789 amd64_CCC=             $(ONBLD_TOOLS)/bin/$(MACH)/cw -_CC
790 $(__GNUC64)amd64_CCC=  $(ONBLD_TOOLS)/bin/$(MACH)/cw -_g++
791 amd64_CPP=             /usr/ccs/lib/cpp
792 amd64_AS=              $(ONBLD_TOOLS)/bin/$(MACH)/aw
793 amd64_LD=              /usr/ccs/bin/ld
794 amd64_LINT=            $(SPRO_ROOT)/sunstudio12.1/bin/lint

796 NATIVECC=             $($(NATIVE_MACH)_CC)
797 NATIVECCC=            $($(NATIVE_MACH)_CCC)
798 NATIVECPP=            $($(NATIVE_MACH)_CPP)
799 NATIVEAS=             $($(NATIVE_MACH)_AS)
800 NATIVELD=             $($(NATIVE_MACH)_LD)
801 NATIVELINT=           $($(NATIVE_MACH)_LINT)

803 #
804 # Makefile.master.64 overrides these settings
805 #
806 CC=                   $(NATIVECC)
807 CCC=                  $(NATIVECCC)
808 CPP=                  $(NATIVECPP)
809 AS=                   $(NATIVEAS)
810 LD=                   $(NATIVELD)
811 LINT=                 $(NATIVELINT)

813 # The real compilers used for this build
814 CW_CC_CMD=            $(CC) -_compiler
815 CW_CCC_CMD=           $(CCC) -_compiler
816 REAL_CC=              $(CW_CC_CMD:sh)
817 REAL_CCC=             $(CW_CCC_CMD:sh)

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

822 BDIRECT=        -Bdirect
823 BDYNAMIC=       -Bdynamic
824 BLOCAL=         -Blocal
825 BNODIRECT=      -Bnodirect
826 BREDUCE=        -Breduce
827 BSTATIC=        -Bstatic

829 ZDEFS=          -zdefs
830 ZDIRECT=        -zdirect
831 ZIGNORE=        -zignore
832 ZINITFIRST=     -zinitfirst
833 ZINTERPOSE=     -zinterpose
834 ZLAZYLOAD=      -zlazyload
835 ZLOADFLTR=      -zloadfltr
836 ZMULDEFS=       -zmuldefs
837 ZNODEFAULTLIB=  -znodefaultlib
```



```
838 ZNODEFS=        -znodefs
839 ZNODELETE=      -znodelete
840 ZNODLOPEN=      -znodlopen
841 ZNODUMP=        -znodump
842 ZNOLAZYLOAD=    -znolazyload
843 ZNOLDYNSYM=     -znoldynsym
844 ZNORELOC=       -znoreloc
845 ZNOVERSION=     -znoversion
846 ZRECORD=        -zrecord
847 ZREDLOCSYM=     -zredlocsym
848 ZTEXT=          -ztext
849 ZVERBOSE=       -zverbose

851 GSHARED=        -G
852 CCMT=           -mt

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

857 sparc_C_PICFLAGS =      -K pic
858 sparcv9_C_PICFLAGS =    -K pic
859 i386_C_PICFLAGS =       -K pic
860 amd64_C_PICFLAGS =      -K pic
861 C_PICFLAGS =            $($(MACH)_C_PICFLAGS)
862 C_PICFLAGS64 =          $($(MACH64)_C_PICFLAGS)

864 sparc_C_BIGPICFLAGS =   -K PIC
865 sparcv9_C_BIGPICFLAGS = -K PIC
866 i386_C_BIGPICFLAGS =    -K PIC
867 amd64_C_BIGPICFLAGS =   -K PIC
868 C_BIGPICFLAGS =         $($(MACH)_C_BIGPICFLAGS)
869 C_BIGPICFLAGS64 =       $($(MACH64)_C_BIGPICFLAGS)

871 # CC requires there to be no space between '-K' and 'pic' or 'PIC'.
872 sparc_CC_PICFLAGS =     -Kpic
873 sparcv9_CC_PICFLAGS =   -KPIC
874 i386_CC_PICFLAGS =      -Kpic
875 amd64_CC_PICFLAGS =     -Kpic
876 CC_PICFLAGS =           $($(MACH)_CC_PICFLAGS)
877 CC_PICFLAGS64 =         $($(MACH64)_CC_PICFLAGS)

879 AS_PICFLAGS=           $(C_PICFLAGS)
880 AS_BIGPICFLAGS=        $(C_BIGPICFLAGS)

882 #
883 # Default label for CTF sections
884 #
885 CTFCVTFLAGS=           -i -L VERSION
886 $(SRCDBGBLD)CTFCVTFLAGS        += -g

888 #
889 # Override to pass module-specific flags to ctfmerge.  Currently used only by
890 # krtld to turn on fuzzy matching, and source-level debugging to inhibit
891 # stripping.
892 #
893 CTFMRGFLAGS=
894 $(SRCDBGBLD)CTFMRGFLAGS        += -g


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

899 ELFSIGN_O=      $(TRUE)
900 ELFSIGN_CRYPTO= $(ELFSIGN_O)
901 ELFSIGN_OBJECT= $(ELFSIGN_O)

903 # Rules (normally from make.rules) and macros which are used for post
```

```
904 # processing files. Normally, these do stripping of the comment section
905 # automatically.
906 #    RELEASE_CM:        Should be editted to reflect the release.
907 #    POST_PROCESS_O:    Post-processing for '.o' files.
908 #    POST_PROCESS_A:    Post-processing for '.a' files (currently null).
909 #    POST_PROCESS_SO:   Post-processing for '.so' files.
910 #    POST_PROCESS:      Post-processing for executable files (no suffix).
911 # Note that these macros are not completely generalized as they are to be
912 # used with the file name to be processed following.
913 #
914 # It is left as an exercise to Release Engineering to embellish the generation
915 # of the release comment string.
916 #
917 #       If this is a standard development build:
918 #               compress the comment section (mcs -c)
919 #               add the standard comment (mcs -a $(RELEASE_CM))
920 #               add the development specific comment (mcs -a $(DEV_CM))
921 #
922 #       If this is an installation build:
923 #               delete the comment section (mcs -d)
924 #               add the standard comment (mcs -a $(RELEASE_CM))
925 #               add the development specific comment (mcs -a $(DEV_CM))
926 #
927 #       If this is an release build:
928 #               delete the comment section (mcs -d)
929 #               add the standard comment (mcs -a $(RELEASE_CM))
930 #
931 # The following list of macros are used in the definition of RELEASE_CM
932 # which is used to label all binaries in the build:
933 #
934 #       RELEASE         Specific release of the build, eg: 5.2
935 #       RELEASE_MAJOR   Major version number part of $(RELEASE)
936 #       RELEASE_MINOR   Minor version number part of $(RELEASE)
937 #       VERSION         Version of the build (alpha, beta, Generic)
938 #       PATCHID         If this is a patch this value should contain
939 #                       the patchid value (eg: "Generic 100832-01"), otherwise
940 #                       it will be set to $(VERSION)
941 #       RELEASE_DATE    Date of the Release Build
942 #       PATCH_DATE      Date the patch was created, if this is blank it
943 #                       will default to the RELEASE_DATE
944 #
945 RELEASE_MAJOR=  5
946 RELEASE_MINOR=  11
947 RELEASE=        $(RELEASE_MAJOR).$(RELEASE_MINOR)
948 VERSION=        SunOS Development
949 PATCHID=        $(VERSION)
950 RELEASE_DATE=   release date not set
951 PATCH_DATE=     $(RELEASE_DATE)
952 RELEASE_CM=     "@($(POUND_SIGN))SunOS $(RELEASE) $(PATCHID) $(PATCH_DATE)"
953 DEV_CM=         "@($(POUND_SIGN))SunOS Internal Development: non-nightly build"

955 PROCESS_COMMENT=                  @?${MCS} -c -a $(RELEASE_CM) -a $(DEV_CM)
956 $(STRIP_COMMENTS)PROCESS_COMMENT=  @?${MCS} -d -a $(RELEASE_CM) -a $(DEV_CM)
957 $(RELEASE_BUILD)PROCESS_COMMENT=   @?${MCS} -d -a $(RELEASE_CM)

959 STRIP_STABS=                       :
960 $(RELEASE_BUILD)STRIP_STABS=       $(STRIP) -x $@
961 $(SRCDBGBLD)STRIP_STABS=           :

963 POST_PROCESS_O=         $(PROCESS_COMMENT) $@
964 POST_PROCESS_A=
965 POST_PROCESS_SO=        $(PROCESS_COMMENT) $@ ; $(STRIP_STABS) ; \
966                         $(ELFSIGN_OBJECT)
967 POST_PROCESS=           $(PROCESS_COMMENT) $@ ; $(STRIP_STABS) ; \
968                         $(ELFSIGN_OBJECT)
```



```
970 #
971 # chk4ubin is a tool that inspects a module for a symbol table
972 # ELF section size which can trigger an OBP bug on older platforms.
973 # This problem affects only specific sun4u bootable modules.
974 #
975 CHK4UBIN=       $(ONBLD_TOOLS)/bin/$(MACH)/chk4ubin
976 CHK4UBINFLAGS=
977 CHK4UBINARY=    $(CHK4UBIN) $(CHK4UBINFLAGS) $@

979 #
980 # PKGARCHIVE specifies the default location where packages should be
981 # placed if built.
982 #
983 $(RELEASE_BUILD)PKGARCHIVESUFFIX=       -nd
984 PKGARCHIVE=$(SRC)/../../packages/$(MACH)/nightly$(PKGARCHIVESUFFIX)

986 #
987 # The repositories will be created with these publisher settings.  To
988 # update an image to the resulting repositories, this must match the
989 # publisher name provided to "pkg set-publisher."
990 #
991 PKGPUBLISHER_REDIST=    on-nightly
992 PKGPUBLISHER_NONREDIST= on-extra

994 #       Default build rules which perform comment section post-processing.
995 #
996 .c:
997         $(LINK.c) -o $@ $< $(LDLIBS)
998         $(POST_PROCESS)
999 .c.o:
1000        $(COMPILE.c) $(OUTPUT_OPTION) $< $(CTFCONVERT_HOOK)
1001        $(POST_PROCESS_O)
1002 .c.a:
1003        $(COMPILE.c) -o $% $<
1004        $(PROCESS_COMMENT) $%
1005        $(AR) $(ARFLAGS) $@ $%
1006        $(RM) $%
1007 .s.o:
1008        $(COMPILE.s) -o $@ $<
1009        $(POST_PROCESS_O)
1010 .s.a:
1011        $(COMPILE.s) -o $% $<
1012        $(PROCESS_COMMENT) $%
1013        $(AR) $(ARFLAGS) $@ $%
1014        $(RM) $%
1015 .cc:
1016        $(LINK.cc) -o $@ $< $(LDLIBS)
1017        $(POST_PROCESS)
1018 .cc.o:
1019        $(COMPILE.cc) $(OUTPUT_OPTION) $<
1020        $(POST_PROCESS_O)
1021 .cc.a:
1022        $(COMPILE.cc) -o $% $<
1023        $(AR) $(ARFLAGS) $@ $%
1024        $(PROCESS_COMMENT) $%
1025        $(RM) $%
1026 .y:
1027        $(YACC.y) $<
1028        $(LINK.c) -o $@ y.tab.c $(LDLIBS)
1029        $(POST_PROCESS)
1030        $(RM) y.tab.c
1031 .y.o:
1032        $(YACC.y) $<
1033        $(COMPILE.c) -o $@ y.tab.c $(CTFCONVERT_HOOK)
1034        $(POST_PROCESS_O)
1035        $(RM) y.tab.c
```

```
1036 .l:
1037         $(RM) $*.c
1038         $(LEX.l) $< > $*.c
1039         $(LINK.c) -o $@ $*.c -ll $(LDLIBS)
1040         $(POST_PROCESS)
1041         $(RM) $*.c
1042 .l.o:
1043         $(RM) $*.c
1044         $(LEX.l) $< > $*.c
1045         $(COMPILE.c) -o $@ $*.c $(CTFCONVERT_HOOK)
1046         $(POST_PROCESS_O)
1047         $(RM) $*.c

1049 .bin.o:
1050         $(COMPILE.b) -o $@ $<
1051         $(POST_PROCESS_O)

1053 .java.class:
1054         $(COMPILE.java) $<

1056 # Bourne and Korn shell script message catalog build rules.
1057 # We extract all gettext strings with sed(1) (being careful to permit
1058 # multiple gettext strings on the same line), weed out the dups, and
1059 # build the catalogue with awk(1).

1061 .sh.po .ksh.po:
1062         $(SED) -n -e ":a"                                       \
1063                   -e "h"                                        \
1064                   -e "s/.*gettext *\(\"[^\"]*\"\).*/\1/p"       \
1065                   -e "x"                                        \
1066                   -e "s/\(.*\)gettext *\"[^\"]*\"\(.*\)/\1\2/"  \
1067                   -e "t a"                                      \
1068                 $< | sort -u | awk '{ print "msgid\t" $$0 "\nmsgstr" }' > $@

1070 #
1071 # Python and Perl executable and message catalog build rules.
1072 #
1073 .SUFFIXES: .pl .pm .py .pyc

1075 .pl:
1076         $(RM) $@;
1077         $(SED) -e "s@TEXT_DOMAIN@\"$(TEXT_DOMAIN)\"@" $< > $@;
1078         $(CHMOD) +x $@

1080 .py:
1081         $(RM) $@; $(CAT) $< > $@; $(CHMOD) +x $@

1083 .py.pyc:
1084         $(RM) $@
1085         $(PYTHON) -mpy_compile $<
1086         @[ $(<)c = $@ ] || $(MV) $(<)c $@

1088 .py.po:
1089         $(GNUXGETTEXT) $(GNUXGETFLAGS) -d $(<F:%.py=%) $< ;

1091 .pl.po .pm.po:
1092         $(XGETTEXT) $(XGETFLAGS) -d $(<F) $< ;
1093         $(RM)   $@ ;
1094         $(SED) "/^domain/d" < $(<F).po > $@ ;
1095         $(RM) $(<F).po

1097 #
1098 # When using xgettext, we want messages to go to the default domain,
1099 # rather than the specified one.  This special version of the
1100 # COMPILE.cpp macro effectively prevents expansion of TEXT_DOMAIN,
1101 # causing xgettext to put all messages into the default domain.
```



```
1102 #
1103 CPPFORPO=$(COMPILE.cpp:\"$(TEXT_DOMAIN)\"=TEXT_DOMAIN)

1105 .c.i:
1106         $(CPPFORPO) $< > $@

1108 .h.i:
1109         $(CPPFORPO) $< > $@

1111 .y.i:
1112         $(YACC) -d $<
1113         $(CPPFORPO) y.tab.c  > $@
1114         $(RM) y.tab.c

1116 .l.i:
1117         $(LEX) $<
1118         $(CPPFORPO) lex.yy.c  > $@
1119         $(RM) lex.yy.c

1121 .c.po:
1122         $(CPPFORPO) $< > $<.i
1123         $(BUILD.po)

1125 .y.po:
1126         $(YACC) -d $<
1127         $(CPPFORPO) y.tab.c  > $<.i
1128         $(BUILD.po)
1129         $(RM) y.tab.c

1131 .l.po:
1132         $(LEX) $<
1133         $(CPPFORPO) lex.yy.c  > $<.i
1134         $(BUILD.po)
1135         $(RM) lex.yy.c

1137 #
1138 # Rules to perform stylistic checks
1139 #
1140 .SUFFIXES: .x .xml .check .xmlchk

1142 .h.check:
1143         $(DOT_H_CHECK)

1145 .x.check:
1146         $(DOT_X_CHECK)

1148 .xml.xmlchk:
1149         $(MANIFEST_CHECK)

1151 #
1152 # Include rules to render automated sccs get rules "safe".
1153 #
1154 include $(SRC)/Makefile.noget
```

```
**********************************************************
    8584 Thu Aug 15 12:00:00 2013
new/usr/src/lib/Makefile.lib
4029 remove tonic build bits
**********************************************************
```

```
#
# 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

LINTLIB=        llib-l$(LIBNAME).ln
LINTFLAGS=      -uaxm
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 # $CLOSED/Makefile.tonic.
165 #
166 # If you do not do this, then the closedbins build for the OpenSolaris
167 # community will break. PS, the gatekeepers will be upset too.
168 #
162 INS.liblink=    -$(RM) $@; $(SYMLINK) $(LIBLINKPATH)$(LIBLINKS)$(VERS) $@
163 INS.liblinkccc= -$(RM) $@; $(SYMLINK) $(LIBLINKPATH)$(LIBLINKSCCC)$(VERS) $@

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

169 #
170 # If appropriate, augment POST_PROCESS_O and POST_PROCESS_SO to do CTF
171 # processing.  We'd like to just conditionally append to POST_PROCESS_O and
172 # POST_PROCESS_SO, but ParallelMake has a bug which causes the same value to
173 # sometimes get appended more than once, which will cause ctfconvert to fail.
174 # So, instead we introduce CTFCONVERT_POST and CTFMERGE_POST, which are always
175 # appended to POST_PROCESS_O and POST_PROCESS_SO but are no-ops unless CTF
176 # processing should be done.
177 #
178 CTFCONVERT_POST = :
179 CTFMERGE_POST   = :
180 POST_PROCESS_O += ; $(CTFCONVERT_POST)
181 POST_PROCESS_SO += ; $(CTFMERGE_POST)

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

185 # conditional assignments
```



```
187 $(OBJS)  :=     sparc_CFLAGS += -xregs=no%appl

189 $(PICS)  :=     sparc_CFLAGS += -xregs=no%appl $(sparc_C_PICFLAGS)
190 $(PICS)  :=     sparcv9_CFLAGS += -xregs=no%appl $(sparcv9_C_PICFLAGS)
191 $(PICS)  :=     i386_CFLAGS += $(i386_C_PICFLAGS)
192 $(PICS)  :=     amd64_CFLAGS += $(amd64_C_PICFLAGS)
193 $(PICS)  :=     CCFLAGS += $(CC_PICFLAGS)
194 $(PICS)  :=     CPPFLAGS += -DPIC -D_REENTRANT
195 $(PICS)  :=     sparcv9_CCFLAGS += -xregs=no%appl $(sparcv9_CC_PICFLAGS)
196 $(PICS)  :=     amd64_CCFLAGS += $(amd64_CC_PICFLAGS)
197 $(PICS)  :=     CFLAGS += $(CTF_FLAGS)
198 $(PICS)  :=     CFLAGS64 += $(CTF_FLAGS)
199 $(PICS)  :=     CTFCONVERT_POST = $(CTFCONVERT_O)
200 $(DYNLIB) :=    CTFMERGE_POST = $(CTFMERGE_LIB)

202 $(LINTLIB):=    LOG = -DLOGGING
203 $(LIBRARY):=    AROBJS = $(OBJS)
204 $(LIBRARY):=    DIR = objs
205 $(DYNLIB):=     DIR = pics
206 $(DYNLIBCCC):=  DIR = pics

208 SONAMECCC=      $(DYNLIBCCC)
209 HSONAMECCC=     -h $(SONAMECCC)
210 #
211 # Keep in sync with the standard DYNFLAGS
212 #
213 $(DYNLIBCCC):=  DYNFLAGS = $(HSONAMECCC) $(ZTEXT) $(ZDEFS) \
214                 $(MAPFILES:%=-M%) $(MAPFILE.PGA:%=-M%) $(MAPFILE.NED:%=-M%) \
215                 $(BDIRECT) $(NORUNPATH)


218 # build rule for "portable" source
219 objs/%.o pics/%.o: %.c
220         $(COMPILE.c) -o $@ $<
221         $(POST_PROCESS_O)

223 objs/%.o pics/%.o: %.cc
224         $(COMPILE.cc) -o $@ $<
225         $(POST_PROCESS_O)

227 .PRECIOUS: $(LIBS)

229 # Define the majority text domain in this directory.
230 TEXT_DOMAIN= SUNW_OST_OSLIB

232 $(ROOTMAN3)/%: %.sunman
233         $(INS.rename)

235 #
236 # For library source code, we expect that some symbols may not be used or
237 # may *appear* to be able to rescoped to static; shut lint up.  Never add
238 # a flag here unless you're *sure* that all libraries need to be linted
239 # with it.
240 #
241 LINTCHECKFLAGS = -m -erroff=E_NAME_DEF_NOT_USED2
242 LINTCHECKFLAGS += -erroff=E_NAME_DECL_NOT_USED_DEF2

244 #
245 # Target Architecture
246 #
247 TARGETMACH=     $(MACH)

249 #
250 # Allow people to define their own clobber rules.  Normal makefiles
251 # shouldn't override this - they should override $(CLOBBERFILES) instead.
252 #
```

```
 253 CLOBBERTARGFILES= $(LIBS) $(DYNLIB) $(CLOBBERFILES)
```

```
**********************************************************
    2213 Thu Aug 15 12:00:00 2013
new/usr/src/tools/findunref/exception_list.closed
4029 remove tonic build bits
**********************************************************
#
# 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) 2010, Oracle and/or its affiliates. All rights reserved.
#

#
# closed-tree exception list
#
# See README.exception_lists for details
#

#
# Ignore files that get used during a EXPORT_SRC or CRYPT_SRC build only.
#
./usr/closed/uts/sun4v/io/n2cp/Makefile
./usr/closed/uts/sun4v/io/ncp/Makefile

#
# Ignore files that are only used for warlock
#
./usr/closed/uts/sparc/marvell88sx/Makefile

#
# An unfortunate artifact of the bridged, split gate is that closed-source
# deleted files go where findunref can accidentally find them...
#
# And an unfortunate artifact of using these tools with both full Teamware
# and split, non-Teamware workspaces is that sometimes closed/deleted_files
# won't exist, so we need the ISUSED directive here.
#
# ISUSED - let checkpaths know that the next entry is good.
./usr/closed/deleted_files

#
# Ignore some files originally shared by Broadcom as part of bnxe driver
#
./usr/closed/uts/common/io/bnxe/577xx
./usr/closed/uts/common/io/bnxe/src/bnxe_debug.h

#
# Ignore some files originally shared by Broadcom as part of bnx driver
```



```
#
./usr/closed/uts/common/io/bnx/hsi.h
./usr/closed/uts/common/io/bnx/invm_cfg.h
./usr/closed/uts/common/io/bnx/iparms.h
./usr/closed/uts/common/io/bnx/itypes.h
./usr/closed/uts/common/io/bnx/status_code.h
./usr/closed/uts/common/io/bnx/tcp_ctx.h
./usr/closed/uts/common/io/bnx/toe_ctx.h
./usr/closed/uts/common/io/bnx/bnxdbg.c

#
# Ignore Makefile.tonic - not used unless we're running an
# OpenSolaris closedbins delivery build
#
./usr/closed/Makefile.tonic
```

```
**********************************************************
    5024 Thu Aug 15 12:00:00 2013
new/usr/src/tools/install.bin/install.bin.c
4029 remove tonic build bits
**********************************************************
```

```
/*
 * 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) 1999, 2010, Oracle and/or its affiliates. All rights reserved.
 */


#include <stdio.h>
#include <stdlib.h>
#include <strings.h>
#include <sys/param.h>
#include <fcntl.h>
#include <sys/errno.h>
#include <sys/types.h>
#include <sys/uio.h>
#include <unistd.h>
#include <sys/stat.h>
#include <errno.h>
#include <libgen.h>
#include "stdusers.h"


#define FILE_BUFF       40960

static int suppress = 0;

static void usage(void);
static void file_copy(char *src_file, char *dest_file);
static void chown_file(const char *file, const char *group, const char *owner);
static void formclosed(char *root, char *closedroot);
static char *find_basename(const char *str);
static int creatdir(char *fn);


void
usage(void)
{
        (void) fprintf(stderr,
            "usage: install [-sd][-m mode][-g group][-u owner] "
            "usage: install [-sdO][-m mode][-g group][-u owner] "
            "-f dir file ...\n");
}
```

_____*unchanged_portion_omitted_*


```
void
formclosed(char *root, char *closedroot)
{
        int wholelen, residlen;
        char *temp;

        wholelen = strlen(root);
        temp = strstr(strstr(root, "proto/root_"), "/");
        temp++;
        temp = strstr(temp, "/");
        residlen = strlen(temp);
        (void) strlcpy(closedroot, root, wholelen - residlen + 1);
        (void) strlcat(closedroot, "-closed", MAXPATHLEN);
        (void) strlcat(closedroot, temp, MAXPATHLEN);
}


char *
find_basename(const char *str)
{
        int     i;
        int     len;

        len = strlen(str);

        for (i = len-1; i >= 0; i--)
                if (str[i] == '/')
                        return ((char *)(str + i + 1));
        return ((char *)str);
}
```

_____*unchanged_portion_omitted_*

```
int
main(int argc, char **argv)
{
        int     c;
        int     errflg = 0;
        int     dirflg = 0;
        char    *group = NULL;
        char    *owner = NULL;
        char    *dirb = NULL;
        char    *ins_file = NULL;
        int     mode = -1;
        char    dest_file[MAXPATHLEN];
        char    shadow_dest[MAXPATHLEN];
        char    shadow_dirb[MAXPATHLEN];
        int     tonic = 0;
        int     rv = 0;

        while ((c = getopt(argc, argv, "f:sm:du:g:")) != EOF) {
        while ((c = getopt(argc, argv, "f:sm:du:g:O")) != EOF) {
                switch (c) {
                case 'f':
                        dirb = optarg;
                        break;
                case 'g':
                        group = optarg;
                        break;
                case 'u':
                        owner = optarg;
                        break;
                case 'd':
                        dirflg = 1;
```

```
183                         break;
184                 case 'm':
185                         mode = strtol(optarg, NULL, 8);
186                         break;
187                 case 's':
188                         suppress = 1;
189                         break;
211                 case 'O':
212                         tonic = 1;
213                         break;
190                 case '?':
191                         errflg++;
192                         break;
193                 }
194         }

196         if (errflg) {
197                 usage();
198                 return (1);
199         }

201         if (argc == optind) {
202                 usage();
203                 return (1);
204         }

206         if (!dirflg && (dirb == NULL)) {
207                 (void) fprintf(stderr,
208                     "install: no destination directory specified.\n");
209                 return (1);
210         }

212         for (c = optind; c < argc; c++) {
213                 ins_file = argv[c];

215                 if (dirflg) {
240                         if (tonic) {
241                                 formclosed(ins_file, shadow_dest);
242                                 rv = creatdir(shadow_dest);
243                                 if (rv) {
244                                         (void) fprintf(stderr,
245                                             "install: tonic creatdir "
246                                             "%s (%d): (%s)\n",
247                                             shadow_dest, errno,
248                                             strerror(errno));
249                                         return (rv);
250                                 }
251                         }
216                         rv = creatdir(ins_file);
217                         if (rv) {
218                                 (void) fprintf(stderr,
219                                     "install: creatdir %s (%d): %s\n",
220                                     ins_file, errno, strerror(errno));
221                                 return (rv);
222                         }
223                         (void) strlcpy(dest_file, ins_file, MAXPATHLEN);

225                 } else {
226                         (void) strcat(strcat(strcpy(dest_file, dirb), "/"),
227                             find_basename(ins_file));
228                         file_copy(ins_file, dest_file);

266                         if (tonic) {
267                                 formclosed(dirb, shadow_dirb);
268                                 /*
269                                  * The standard directories in the proto
```



```
270                                  * area are created as part of "make setup",
271                                  * but that doesn't create them in the
272                                  * closed proto area. So if the target
273                                  * directory doesn't exist, we need to
274                                  * create it now.
275                                  */
276                                 rv = creatdir(shadow_dirb);
277                                 if (rv) {
278                                         (void) fprintf(stderr,
279                                             "install: tonic creatdir(f) "
280                                             "%s (%d): %s\n",
281                                             shadow_dirb, errno,
282                                             strerror(errno));
283                                         return (rv);
284                                 }
285                                 (void) strcat(strcat(strcpy(shadow_dest,
286                                     shadow_dirb), "/"),
287                                     find_basename(ins_file));
288                                 file_copy(ins_file, shadow_dest);
289                         }
229                 }

231                 if (group || owner)
292                 if (group || owner) {
232                         chown_file(dest_file, group, owner);

294                         if (tonic)
295                                 chown_file(shadow_dest, group, owner);
296                 }
234                 if (mode != -1) {
235                         (void) umask(0);
236                         if (chmod(dest_file, mode) == -1) {
237                                 (void) fprintf(stderr,
238                                     "install: chmod of %s to mode %o failed "
239                                     "(%d): %s\n",
240                                     dest_file, mode, errno, strerror(errno));
241                                 return (1);
242                         }
306                         if (tonic) {
307                                 if (chmod(shadow_dest, mode) == -1) {
308                                         (void) fprintf(stderr,
309                                             "install: tonic chmod of %s "
310                                             "to mode %o failed (%d): %s\n",
311                                             shadow_dest, mode,
312                                             errno, strerror(errno));
313                                         return (1);
314                                 }
315                         }
243                 }
244         }
245         return (0);
246 }
```

*_____unchanged_portion_omitted_*

```
**********************************************************
   12468 Thu Aug 15 12:00:00 2013
new/usr/src/tools/scripts/bldenv.sh
4029 remove tonic build bits
**********************************************************
______unchanged_portion_omitted_
 154 [+SEE ALSO?\bnightly\b(1)]
 155 '

 157 # main
 158 builtin basename

 160 # boolean flags (true/false)
 161 typeset flags=(
 162         typeset c=false
 163         typeset f=false
 164         typeset d=false
 165         typeset O=false
 166         typeset o=false
 167         typeset t=true
 168         typeset s=(
 169                 typeset e=false
 170                 typeset h=false
 171                 typeset d=false
 172                 typeset o=false
 173         )
 174 )

 176 typeset progname="$(basename -- "${0}")"

 178 OPTIND=1
 179 SUFFIX="-nd"

 181 while getopts -a "${progname}" "${USAGE}" OPT ; do
 182     case ${OPT} in
 183           c)    flags.c=true  ;;
 184           +c)   flags.c=false ;;
 185           f)    flags.f=true  ;;
 186           +f)   flags.f=false ;;
 187           d)    flags.d=true  SUFFIX=""    ;;
 188           +d)   flags.d=false SUFFIX="-nd" ;;
 189           t)    flags.t=true  ;;
 190           +t)   flags.t=false ;;
 191           S)    set_S_flag "$OPTARG" ;;
 192           \?)   usage ;;
 193     esac
 194 done
 195 shift $((OPTIND-1))

 197 # test that the path to the environment-setting file was given
 198 if (( $# < 1 )) ; then
 199         usage
 200 fi

 202 # force locale to C
 203 export \
 204         LC_COLLATE=C \
 205         LC_CTYPE=C \
 206         LC_MESSAGES=C \
 207         LC_MONETARY=C \
 208         LC_NUMERIC=C \
 209         LC_TIME=C

 211 # clear environment variables we know to be bad for the build
 212 unset \
 213         LD_OPTIONS \
```



```
 214         LD_LIBRARY_PATH \
 215         LD_AUDIT \
 216         LD_BIND_NOW \
 217         LD_BREADTH \
 218         LD_CONFIG \
 219         LD_DEBUG \
 220         LD_FLAGS \
 221         LD_LIBRARY_PATH_64 \
 222         LD_NOVERSION \
 223         LD_ORIGIN \
 224         LD_LOADFLTR \
 225         LD_NOAUXFLTR \
 226         LD_NOCONFIG \
 227         LD_NODIRCONFIG \
 228         LD_NOOBJALTER \
 229         LD_PRELOAD \
 230         LD_PROFILE \
 231         CONFIG \
 232         GROUP \
 233         OWNER \
 234         REMOTE \
 235         ENV \
 236         ARCH \
 237         CLASSPATH

 239 #
 240 # Setup environment variables
 241 #
 242 if [[ -f /etc/nightly.conf ]]; then
 243         source /etc/nightly.conf
 244 fi

 246 if [[ -f "$1" ]]; then
 247         if [[ "$1" == */* ]]; then
 248                 source "$1"
 249         else
 250                 source "./$1"
 251         fi
 252 else
 253         if [[ -f "/opt/onbld/env/$1" ]]; then
 254                 source "/opt/onbld/env/$1"
 255         else
 256                 printf \
 257                     'Cannot find env file as either %s or /opt/onbld/env/%s\n' \
 258                     "$1" "$1"
 259                 exit 1
 260         fi
 261 fi
 262 shift

 264 # contents of stdenv.sh inserted after next line:
 265 # STDENV_START
 266 # STDENV_END

 268 # Check if we have sufficient data to continue...
 269 [[ -v CODEMGR_WS ]] || fatal_error "Error: Variable CODEMGR_WS not set."
 270 [[ -d "${CODEMGR_WS}" ]] || fatal_error "Error: ${CODEMGR_WS} is not a directory
 271 [[ -f "${CODEMGR_WS}/usr/src/Makefile" ]] || fatal_error "Error: ${CODEMGR_WS}/u

 273 # must match the getopts in nightly.sh
 274 OPTIND=1
 275 NIGHTLY_OPTIONS="-${NIGHTLY_OPTIONS#-}"
 276 while getopts '+0AaBCDdFfGIilMmNnOopRrS:tUuWwXxz' FLAG "$NIGHTLY_OPTIONS"
 277 do
 278         case "$FLAG" in
 279           O)    flags.O=true  ;;
```

```
280           +O)   flags.O=false ;;
281           o)    flags.o=true  ;;
282           +o)   flags.o=false ;;
283           t)    flags.t=true  ;;
284           +t)   flags.t=false ;;
285           S)    set_S_flag "$OPTARG" ;;
286           *)    ;;
287         esac
288 done

290 POUND_SIGN="#"
291 # have we set RELEASE_DATE in our env file?
292 if [ -z "$RELEASE_DATE" ]; then
293         RELEASE_DATE=$(LC_ALL=C date +"%B %Y")
294 fi
295 BUILD_DATE=$(LC_ALL=C date +%Y-%b-%d)
296 BASEWSDIR=$(basename -- "${CODEMGR_WS}")
297 DEV_CM="\"@(#)SunOS Internal Development: $LOGNAME $BUILD_DATE [$BASEWSDIR]\""
298 export DEV_CM RELEASE_DATE POUND_SIGN

300 export INTERNAL_RELEASE_BUILD=

302 print 'Build type   is  \c'
303 if ${flags.d} ; then
304         print 'DEBUG'
305         unset RELEASE_BUILD
306         unset EXTRA_OPTIONS
307         unset EXTRA_CFLAGS
308 else
309         # default is a non-DEBUG build
310         print 'non-DEBUG'
311         export RELEASE_BUILD=
312         unset EXTRA_OPTIONS
313         unset EXTRA_CFLAGS
314 fi

316 [[ "${flags.O}" == "true" ]] && export MULTI_PROTO="yes"

318 # update build-type variables
319 PKGARCHIVE="${PKGARCHIVE}${SUFFIX}"

321 # Append source version
322 if "${flags.s.e}" ; then
323         VERSION+=":EXPORT"
324         SRC="${EXPORT_SRC}/usr/src"
325 fi
326
327 if "${flags.s.d}" ; then
328         VERSION+=":DOMESTIC"
329         SRC="${EXPORT_SRC}/usr/src"
330 fi

332 if "${flags.s.h}" ; then
333         VERSION+=":HYBRID"
334         SRC="${EXPORT_SRC}/usr/src"
335 fi
336
337 if "${flags.s.o}" ; then
338         VERSION+=":OPEN_ONLY"
339         SRC="${OPEN_SRCDIR}/usr/src"
340 fi

342 #       Set PATH for a build
343 PATH="/opt/onbld/bin:/opt/onbld/bin/${MACH}:/opt/SUNWspro/bin:/usr/ccs/bin:/usr/
344 if [[ "${SUNWSPRO}" != "" ]]; then
345         export PATH="${SUNWSPRO}/bin:$PATH"
```



```
346 fi

348 TOOLS="${SRC}/tools"
349 TOOLS_PROTO="${TOOLS}/proto/root_${MACH}-nd" ; export TOOLS_PROTO

351 if "${flags.t}" ; then
352         export ONBLD_TOOLS="${ONBLD_TOOLS:=${TOOLS_PROTO}/opt/onbld}"

354         export STABS="${TOOLS_PROTO}/opt/onbld/bin/${MACH}/stabs"
355         export CTFSTABS="${TOOLS_PROTO}/opt/onbld/bin/${MACH}/ctfstabs"
356         export GENOFFSETS="${TOOLS_PROTO}/opt/onbld/bin/genoffsets"

358         export CTFCONVERT="${TOOLS_PROTO}/opt/onbld/bin/${MACH}/ctfconvert"
359         export CTFMERGE="${TOOLS_PROTO}/opt/onbld/bin/${MACH}/ctfmerge"

361         export CTFCVTPTBL="${TOOLS_PROTO}/opt/onbld/bin/ctfcvtptbl"
362         export CTFFINDMOD="${TOOLS_PROTO}/opt/onbld/bin/ctffindmod"

364         PATH="${TOOLS_PROTO}/opt/onbld/bin/${MACH}:${PATH}"
365         PATH="${TOOLS_PROTO}/opt/onbld/bin:${PATH}"
366         export PATH
367 fi

369 export DMAKE_MODE=${DMAKE_MODE:-parallel}

371 if "${flags.o}" ; then
372         export CH=
373 else
374         unset CH
375 fi
376 DEF_STRIPFLAG="-s"

378 TMPDIR="/tmp"

380 # "o_FLAG" is used by "nightly.sh" (it may be useful to rename this
381 # variable using a more descriptive name later)
382 export o_FLAG="$(${flags.o} && print 'y' || print 'n')"

384 export \
385         PATH TMPDIR \
386         POUND_SIGN \
387         DEF_STRIPFLAG \
388         RELEASE_DATE
389 unset \
390         CFLAGS \
391         LD_LIBRARY_PATH

393 # a la ws
394 ENVLDLIBS1=
395 ENVLDLIBS2=
396 ENVLDLIBS3=
397 ENVCPPFLAGS1=
398 ENVCPPFLAGS2=
399 ENVCPPFLAGS3=
400 ENVCPPFLAGS4=
401 PARENT_ROOT=
402 PARENT_TOOLS_ROOT=

404 if [[ "$MULTI_PROTO" != "yes" && "$MULTI_PROTO" != "no" ]]; then
405         printf \
406             'WARNING: invalid value for MULTI_PROTO (%s); setting to "no".\n' \
407             "$MULTI_PROTO"
408         export MULTI_PROTO="no"
409 fi

411 [[ "$MULTI_PROTO" == "yes" ]] && export ROOT="${ROOT}${SUFFIX}"
```

```
413 export TONICBUILD="#"
```

```
413 if "${flags.O}" ; then
414         print "OpenSolaris closed binary generation requires "
415         print "closed tree"
416         exit 1
417 fi

419 ENVLDLIBS1="-L$ROOT/lib -L$ROOT/usr/lib"
420 ENVCPPFLAGS1="-I$ROOT/usr/include"
421 MAKEFLAGS=e

423 export \
424         ENVLDLIBS1 \
425         ENVLDLIBS2 \
426         ENVLDLIBS3 \
427         ENVCPPFLAGS1 \
428         ENVCPPFLAGS2 \
429         ENVCPPFLAGS3 \
430         ENVCPPFLAGS4 \
431         MAKEFLAGS \
432         PARENT_ROOT \
433         PARENT_TOOLS_ROOT

435 printf 'RELEASE      is %s\n'   "$RELEASE"
436 printf 'VERSION      is %s\n'   "$VERSION"
437 printf 'RELEASE_DATE is %s\n\n' "$RELEASE_DATE"

439 if [[ -f "$SRC/Makefile" ]] && egrep -s '^setup:' "$SRC/Makefile" ; then
440         print "The top-level 'setup' target is available \c"
441         print "to build headers and tools."
442         print ""

444 elif "${flags.t}" ; then
445         printf \
446             'The tools can be (re)built with the install target in %s.\n\n' \
447             "${TOOLS}"
448 fi

450 #
451 # place ourselves in a new task, respecting BUILD_PROJECT if set.
452 #
453 /usr/bin/newtask -c $$ ${BUILD_PROJECT:+-p$BUILD_PROJECT}

455 if [[ "${flags.c}" == "false" && -x "$SHELL" && \
456     "$(basename -- "${SHELL}")" != "csh" ]]; then
457         # $SHELL is set, and it's not csh.

459         if "${flags.f}" ; then
460                 print 'WARNING: -f is ignored when $SHELL is not csh'
461         fi

463         printf 'Using %s as shell.\n' "$SHELL"
464         exec "$SHELL" ${@:+-c "$@"}

466 elif "${flags.f}" ; then
467         print 'Using csh -f as shell.'
468         exec csh -f ${@:+-c "$@"}

470 else
471         print 'Using csh as shell.'
472         exec csh ${@:+-c "$@"}
473 fi

475 # not reached
```

```
**********************************************************
   80166 Thu Aug 15 12:00:00 2013
new/usr/src/tools/scripts/nightly.sh
4029 remove tonic build bits
**********************************************************
______unchanged_portion_omitted_

1041 OPTIND=1
1042 while getopts +inS:tV: FLAG
1043 do
1044         case $FLAG in
1045           i )   i_FLAG=y; i_CMD_LINE_FLAG=y
1046                 ;;
1047           n )   n_FLAG=y
1048                 ;;
1049           S )
1050                 set_S_flag $OPTARG
1051                 ;;
1052          +t )   t_FLAG=n
1053                 ;;
1054           V )   V_FLAG=y
1055                 V_ARG="$OPTARG"
1056                 ;;
1057          \? )   echo "$USAGE"
1058                 exit 1
1059                 ;;
1060         esac
1061 done

1063 # correct argument count after options
1064 shift ‘expr $OPTIND - 1‘

1066 # test that the path to the environment-setting file was given
1067 if [ $# -ne 1 ]; then
1068         echo "$USAGE"
1069         exit 1
1070 fi

1072 # check if user is running nightly as root
1073 # ISUSER is set non-zero if an ordinary user runs nightly, or is zero
1074 # when root invokes nightly.
1075 /usr/bin/id | grep ’^uid=0(’ >/dev/null 2>&1
1076 ISUSER=$?;      export ISUSER

1078 #
1079 # force locale to C
1080 LC_COLLATE=C;   export LC_COLLATE
1081 LC_CTYPE=C;     export LC_CTYPE
1082 LC_MESSAGES=C;  export LC_MESSAGES
1083 LC_MONETARY=C;  export LC_MONETARY
1084 LC_NUMERIC=C;   export LC_NUMERIC
1085 LC_TIME=C;      export LC_TIME

1087 # clear environment variables we know to be bad for the build
1088 unset LD_OPTIONS
1089 unset LD_AUDIT          LD_AUDIT_32             LD_AUDIT_64
1090 unset LD_BIND_NOW       LD_BIND_NOW_32          LD_BIND_NOW_64
1091 unset LD_BREADTH        LD_BREADTH_32           LD_BREADTH_64
1092 unset LD_CONFIG         LD_CONFIG_32            LD_CONFIG_64
1093 unset LD_DEBUG          LD_DEBUG_32             LD_DEBUG_64
1094 unset LD_DEMANGLE       LD_DEMANGLE_32          LD_DEMANGLE_64
1095 unset LD_FLAGS          LD_FLAGS_32             LD_FLAGS_64
1096 unset LD_LIBRARY_PATH   LD_LIBRARY_PATH_32      LD_LIBRARY_PATH_64
1097 unset LD_LOADFLTR       LD_LOADFLTR_32          LD_LOADFLTR_64
1098 unset LD_NOAUDIT        LD_NOAUDIT_32           LD_NOAUDIT_64
1099 unset LD_NOAUXFLTR      LD_NOAUXFLTR_32         LD_NOAUXFLTR_64
```



```
1100 unset LD_NOCONFIG       LD_NOCONFIG_32          LD_NOCONFIG_64
1101 unset LD_NODIRCONFIG    LD_NODIRCONFIG_32       LD_NODIRCONFIG_64
1102 unset LD_NODIRECT       LD_NODIRECT_32          LD_NODIRECT_64
1103 unset LD_NOLAZYLOAD     LD_NOLAZYLOAD_32        LD_NOLAZYLOAD_64
1104 unset LD_NOOBJALTER     LD_NOOBJALTER_32        LD_NOOBJALTER_64
1105 unset LD_NOVERSION      LD_NOVERSION_32         LD_NOVERSION_64
1106 unset LD_ORIGIN         LD_ORIGIN_32            LD_ORIGIN_64
1107 unset LD_PRELOAD        LD_PRELOAD_32           LD_PRELOAD_64
1108 unset LD_PROFILE        LD_PROFILE_32           LD_PROFILE_64

1110 unset CONFIG
1111 unset GROUP
1112 unset OWNER
1113 unset REMOTE
1114 unset ENV
1115 unset ARCH
1116 unset CLASSPATH
1117 unset NAME

1119 #
1120 # To get ONBLD_TOOLS from the environment, it must come from the env file.
1121 # If it comes interactively, it is generally TOOLS_PROTO, which will be
1122 # clobbered before the compiler version checks, which will therefore fail.
1123 #
1124 unset ONBLD_TOOLS

1126 #
1127 #       Setup environmental variables
1128 #
1129 if [ -f /etc/nightly.conf ]; then
1130         . /etc/nightly.conf
1131 fi

1133 if [ -f $1 ]; then
1134         if [[ $1 = */* ]]; then
1135                 . $1
1136         else
1137                 . ./$1
1138         fi
1139 else
1140         if [ -f $OPTHOME/onbld/env/$1 ]; then
1141                 . $OPTHOME/onbld/env/$1
1142         else
1143                 echo "Cannot find env file as either $1 or $OPTHOME/onbld/env/$1
1144                 exit 1
1145         fi
1146 fi

1148 # contents of stdenv.sh inserted after next line:
1149 # STDENV_START
1150 # STDENV_END

1152 # Check if we have sufficient data to continue...
1153 [[ -v CODEMGR_WS ]] || fatal_error "Error: Variable CODEMGR_WS not set."
1154 if  [[ "${NIGHTLY_OPTIONS}" == ~(F)n ]] ; then
1155         # Check if the gate data are valid if we don’t do a "bringover" below
1156         [[ -d "${CODEMGR_WS}" ]] || \
1157                 fatal_error "Error: ${CODEMGR_WS} is not a directory."
1158         [[ -f "${CODEMGR_WS}/usr/src/Makefile" ]] || \
1159                 fatal_error "Error: ${CODEMGR_WS}/usr/src/Makefile not found."
1160 fi

1162 #
1163 # place ourselves in a new task, respecting BUILD_PROJECT if set.
1164 #
1165 if [ -z "$BUILD_PROJECT" ]; then
```

```
1166         /usr/bin/newtask -c $$
1167 else
1168         /usr/bin/newtask -c $$ -p $BUILD_PROJECT
1169 fi

1171 ps -o taskid= -p $$ | read build_taskid
1172 ps -o project= -p $$ | read build_project

1174 #
1175 # See if NIGHTLY_OPTIONS is set
1176 #
1177 if [ "$NIGHTLY_OPTIONS" = "" ]; then
1178         NIGHTLY_OPTIONS="-aBm"
1179 fi

1181 #
1182 # If BRINGOVER_WS was not specified, let it default to CLONE_WS
1183 #
1184 if [ "$BRINGOVER_WS" = "" ]; then
1185         BRINGOVER_WS=$CLONE_WS
1186 fi

1188 #
1189 # If CLOSED_BRINGOVER_WS was not specified, let it default to CLOSED_CLONE_WS
1190 #
1191 if [ "$CLOSED_BRINGOVER_WS" = "" ]; then
1192         CLOSED_BRINGOVER_WS=$CLOSED_CLONE_WS
1193 fi

1195 #
1196 # If BRINGOVER_FILES was not specified, default to usr
1197 #
1198 if [ "$BRINGOVER_FILES" = "" ]; then
1199         BRINGOVER_FILES="usr"
1200 fi

1202 check_closed_tree

1204 #
1205 # Note: changes to the option letters here should also be applied to the
1206 #       bldenv script.  'd' is listed for backward compatibility.
1207 #
1208 NIGHTLY_OPTIONS=-${NIGHTLY_OPTIONS#-}
1209 OPTIND=1
1210 while getopts +ABCDdFfGIilMmNnOoPpRrS:TtUuWwXxz FLAG $NIGHTLY_OPTIONS
1211 do
1212         case $FLAG in
1213           A )   A_FLAG=y
1214                 ;;
1215           B )   D_FLAG=y
1216                 ;; # old version of D
1217           C )   C_FLAG=y
1218                 ;;
1219           D )   D_FLAG=y
1220                 ;;
1221           F )   F_FLAG=y
1222                 ;;
1223           f )   f_FLAG=y
1224                 ;;
1225           G )   u_FLAG=y
1226                 ;;
1227           I )   m_FLAG=y
1228                 p_FLAG=y
1229                 u_FLAG=y
1230                 ;;
1231           i )   i_FLAG=y
```



```
1232                 ;;
1233           l )   l_FLAG=y
1234                 ;;
1235           M )   M_FLAG=y
1236                 ;;
1237           m )   m_FLAG=y
1238                 ;;
1239           N )   N_FLAG=y
1240                 ;;
1241           n )   n_FLAG=y
1242                 ;;
1243           O )   O_FLAG=y
1244                 ;;
1245           o )   o_FLAG=y
1246                 ;;
1247           P )   P_FLAG=y
1248                 ;; # obsolete
1249           p )   p_FLAG=y
1250                 ;;
1251           R )   m_FLAG=y
1252                 p_FLAG=y
1253                 ;;
1254           r )   r_FLAG=y
1255                 ;;
1256           S )
1257                 set_S_flag $OPTARG
1258                 ;;
1259           T )   T_FLAG=y
1260                 ;; # obsolete
1261          +t )   t_FLAG=n
1262                 ;;
1263           U )   if [ -z "${PARENT_ROOT}" ]; then
1264                         echo "PARENT_ROOT must be set if the U flag is" \
1265                             "present in NIGHTLY_OPTIONS."
1266                         exit 1
1267                 fi
1268                 NIGHTLY_PARENT_ROOT=$PARENT_ROOT
1269                 if [ -n "${PARENT_TOOLS_ROOT}" ]; then
1270                         NIGHTLY_PARENT_TOOLS_ROOT=$PARENT_TOOLS_ROOT
1271                 fi
1272                 U_FLAG=y
1273                 ;;
1274           u )   u_FLAG=y
1275                 ;;
1276           W )   W_FLAG=y
1277                 ;;

1279           w )   w_FLAG=y
1280                 ;;
1281           X )   # now that we no longer need realmode builds, just
1282                 # copy IHV packages.  only meaningful on x86.
1283                 if [ "$MACH" = "i386" ]; then
1284                         X_FLAG=y
1285                 fi
1286                 ;;
1287           x )   XMOD_OPT="-x"
1288                 ;;
1289          \? )   echo "$USAGE"
1290                 exit 1
1291                 ;;
1292         esac
1293 done

1295 if [ $ISUSER -ne 0 ]; then
1296         if [ "$o_FLAG" = "y" ]; then
1297                 echo "Old-style build requires root permission."
```

```
1298                 exit 1
1299         fi

1301         # Set default value for STAFFER, if needed.
1302         if [ -z "$STAFFER" -o "$STAFFER" = "nobody" ]; then
1303                 STAFFER=‘/usr/xpg4/bin/id -un‘
1304                 export STAFFER
1305         fi
1306 fi

1308 if [ -z "$MAILTO" -o "$MAILTO" = "nobody" ]; then
1309         MAILTO=$STAFFER
1310         export MAILTO
1311 fi

1313 PATH="$OPTHOME/onbld/bin:$OPTHOME/onbld/bin/${MACH}:/usr/ccs/bin"
1314 PATH="$PATH:$OPTHOME/SUNWspro/bin:$TEAMWARE/bin:/usr/bin:/usr/sbin:/usr/ucb"
1315 PATH="$PATH:/usr/openwin/bin:/usr/sfw/bin:/opt/sfw/bin:."
1316 export PATH

1318 # roots of source trees, both relative to $SRC and absolute.
1319 relsrcdirs="."
1320 abssrcdirs="$SRC"

1322 unset CH
1323 if [ "$o_FLAG" = "y" ]; then
1324 # root invoked old-style build -- make sure it works as it always has
1325 # by exporting ’CH’.  The current Makefile.master doesn’t use this, but
1326 # the old ones still do.
1327         PROTOCMPTERSE="protocmp.terse"
1328         CH=
1329         export CH
1330 else
1331         PROTOCMPTERSE="protocmp.terse -gu"
1332 fi
1333 POUND_SIGN="#"
1334 # have we set RELEASE_DATE in our env file?
1335 if [ -z "$RELEASE_DATE" ]; then
1336         RELEASE_DATE=$(LC_ALL=C date +"%B %Y")
1337 fi
1338 BUILD_DATE=$(LC_ALL=C date +%Y-%b-%d)
1339 BASEWSDIR=$(basename $CODEMGR_WS)
1340 DEV_CM="\"@(#)SunOS Internal Development: $LOGNAME $BUILD_DATE [$BASEWSDIR]\""

1342 # we export POUND_SIGN, RELEASE_DATE and DEV_CM to speed up the build process
1343 # by avoiding repeated shell invocations to evaluate Makefile.master definitions
1344 # we export o_FLAG and X_FLAG for use by makebfu, and by usr/src/pkg/Makefile
1345 export o_FLAG X_FLAG POUND_SIGN RELEASE_DATE DEV_CM

1347 maketype="distributed"
1348 MAKE=dmake
1349 # get the dmake version string alone
1350 DMAKE_VERSION=$( $MAKE -v )
1351 DMAKE_VERSION=${DMAKE_VERSION#*: }
1352 # focus in on just the dotted version number alone
1353 DMAKE_MAJOR=$( echo $DMAKE_VERSION | \
1354         sed -e ’s/.*\<\([^.]*\.[^   ]*\).*$/\1/’ )
1355 # extract the second (or final) integer
1356 DMAKE_MINOR=${DMAKE_MAJOR#*.}
1357 DMAKE_MINOR=${DMAKE_MINOR%%.*}
1358 # extract the first integer
1359 DMAKE_MAJOR=${DMAKE_MAJOR%%.*}
1360 CHECK_DMAKE=${CHECK_DMAKE:-y}
1361 # x86 was built on the 12th, sparc on the 13th.
1362 if [ "$CHECK_DMAKE" = "y" -a \
1363      "$DMAKE_VERSION" != "Sun Distributed Make 7.3 2003/03/12" -a \
```



```
1364      "$DMAKE_VERSION" != "Sun Distributed Make 7.3 2003/03/13" -a \( \
1365      "$DMAKE_MAJOR" -lt 7 -o \
1366      "$DMAKE_MAJOR" -eq 7 -a "$DMAKE_MINOR" -lt 4 \) ]; then
1367         if [ -z "$DMAKE_VERSION" ]; then
1368                 echo "$MAKE is missing."
1369                 exit 1
1370         fi
1371         echo ‘whence $MAKE‘" version is:"
1372         echo "  ${DMAKE_VERSION}"
1373         cat <<EOF

1375 This version may not be safe for use.  Either set TEAMWARE to a better
1376 path or (if you really want to use this version of dmake anyway), add
1377 the following to your environment to disable this check:

1379   CHECK_DMAKE=n
1380 EOF
1381         exit 1
1382 fi
1383 export PATH
1384 export MAKE

1386 if [[ "$O_FLAG" = y ]]; then
1387         export TONICBUILD=""
1388 else
1389         export TONICBUILD="#"
1390 fi

1386 if [ "${SUNWSPRO}" != "" ]; then
1387         PATH="${SUNWSPRO}/bin:$PATH"
1388         export PATH
1389 fi

1391 hostname=$(uname -n)
1392 if [[ $DMAKE_MAX_JOBS != +([0-9]) || $DMAKE_MAX_JOBS -eq 0 ]]
1393 then
1394         maxjobs=
1395         if [[ -f $HOME/.make.machines ]]
1396         then
1397                 # Note: there is a hard tab and space character in the []s
1398                 # below.
1399                 egrep -i "^[    ]*$hostname[    \.]" \
1400                         $HOME/.make.machines | read host jobs
1401                 maxjobs=${jobs##*=}
1402         fi

1404         if [[ $maxjobs != +([0-9]) || $maxjobs -eq 0 ]]
1405         then
1406                 # default
1407                 maxjobs=4
1408         fi

1410         export DMAKE_MAX_JOBS=$maxjobs
1411 fi

1413 DMAKE_MODE=parallel;
1414 export DMAKE_MODE

1416 if [ -z "${ROOT}" ]; then
1417         echo "ROOT must be set."
1418         exit 1
1419 fi

1421 #
1422 # if -V flag was given, reset VERSION to V_ARG
1423 #
```

```
1424 if [ "$V_FLAG" = "y" ]; then
1425         VERSION=$V_ARG
1426 fi

1428 #
1429 # Check for IHV root for copying ihv proto area
1430 #
1431 if [ "$X_FLAG" = "y" ]; then
1432         if [ "$IA32_IHV_ROOT" = "" ]; then
1433                 echo "IA32_IHV_ROOT: must be set for copying ihv proto"
1434                 args_ok=n
1435         fi
1436         if [ ! -d "$IA32_IHV_ROOT" ]; then
1437                 echo "$IA32_IHV_ROOT: not found"
1438                 args_ok=n
1439         fi
1440         if [ "$IA32_IHV_WS" = "" ]; then
1441                 echo "IA32_IHV_WS: must be set for copying ihv proto"
1442                 args_ok=n
1443         fi
1444         if [ ! -d "$IA32_IHV_WS" ]; then
1445                 echo "$IA32_IHV_WS: not found"
1446                 args_ok=n
1447         fi
1448 fi

1450 # Append source version
1451 if [ "$SE_FLAG" = "y" ]; then
1452         VERSION="${VERSION}:EXPORT"
1453 fi

1455 if [ "$SD_FLAG" = "y" ]; then
1456         VERSION="${VERSION}:DOMESTIC"
1457 fi

1459 if [ "$SH_FLAG" = "y" ]; then
1460         VERSION="${VERSION}:MODIFIED_SOURCE_PRODUCT"
1461 fi

1463 if [ "$SO_FLAG" = "y" ]; then
1464         VERSION="${VERSION}:OPEN_ONLY"
1465 fi

1467 TMPDIR="/tmp/nightly.tmpdir.$$"
1468 export TMPDIR
1469 rm -rf ${TMPDIR}
1470 mkdir -p $TMPDIR || exit 1
1471 chmod 777 $TMPDIR

1473 #
1474 # Keep elfsign's use of pkcs11_softtoken from looking in the user home
1475 # directory, which doesn't always work.   Needed until all build machines
1476 # have the fix for 6271754
1477 #
1478 SOFTTOKEN_DIR=$TMPDIR
1479 export SOFTTOKEN_DIR

1481 #
1482 # Tools should only be built non-DEBUG.  Keep track of the tools proto
1483 # area path relative to $TOOLS, because the latter changes in an
1484 # export build.
1485 #
1486 # TOOLS_PROTO is included below for builds other than usr/src/tools
1487 # that look for this location.  For usr/src/tools, this will be
1488 # overridden on the $MAKE command line in build_tools().
1489 #
```



```
1490 TOOLS=${SRC}/tools
1491 TOOLS_PROTO_REL=proto/root_${MACH}-nd
1492 TOOLS_PROTO=${TOOLS}/${TOOLS_PROTO_REL};        export TOOLS_PROTO

1494 unset   CFLAGS LD_LIBRARY_PATH LDFLAGS

1496 # create directories that are automatically removed if the nightly script
1497 # fails to start correctly
1498 function newdir {
1499         dir=$1
1500         toadd=
1501         while [ ! -d $dir ]; do
1502                 toadd="$dir $toadd"
1503                 dir=`dirname $dir`
1504         done
1505         torm=
1506         newlist=
1507         for dir in $toadd; do
1508                 if staffer mkdir $dir; then
1509                         newlist="$ISUSER $dir $newlist"
1510                         torm="$dir $torm"
1511                 else
1512                         [ -z "$torm" ] || staffer rmdir $torm
1513                         return 1
1514                 fi
1515         done
1516         newdirlist="$newlist $newdirlist"
1517         return 0
1518 }
```

_____***unchanged_portion_omitted_***