```
**********************************************************
   34635 Sat Aug  3 15:26:09 2013
new/usr/src/Makefile.master
3971 remove EXPORT_RELEASE_BUILD
**********************************************************
```

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

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

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

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



```
# While the majority of users are expected to override the setting below
# with an env file (via nightly or bldenv), if you aren't building that way
# (ie, you're using "ws" or some other bootstrapping method) then you need
# this definition in order to avoid the subshell invocation mentioned above.
#

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

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

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

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

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

STRIP_COMMENTS= $(INTERNAL_RELEASE_BUILD)

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


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

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

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

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

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

126 JAVA_ROOT=      /usr/java

128 SFW_ROOT=       /usr/sfw
129 SFWINCDIR=      $(SFW_ROOT)/include
130 SFWLIBDIR=      $(SFW_ROOT)/lib
131 SFWLIBDIR64=    $(SFW_ROOT)/lib/$(MACH64)

133 GCC_ROOT=       /opt/gcc/4.4.4
134 GCCLIBDIR=      $(GCC_ROOT)/lib
135 GCCLIBDIR64=    $(GCC_ROOT)/lib/$(MACH64)

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

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



```
186 PYTHON_26=      /usr/bin/python2.6
187 PYTHON=         $(PYTHON_26)
188 SORT=           /usr/bin/sort
189 TOUCH=          /usr/bin/touch
190 WC=             /usr/bin/wc
191 XARGS=          /usr/bin/xargs
192 ELFEDIT=        /usr/bin/elfedit
193 ELFSIGN=        /usr/bin/elfsign
194 DTRACE=         /usr/sbin/dtrace -xnolibs
195 UNIQ=           /usr/bin/uniq
196 TAR=            /usr/bin/tar

198 FILEMODE=       644
199 DIRMODE=        755

201 #
202 # The version of the patch makeup table optimized for build-time use.  Used
203 # during patch builds only.
204 $(PATCH_BUILD)PMTMO_FILE=$(SRC)/patch_makeup_table.mo

206 # Declare that nothing should be built in parallel.
207 # Individual Makefiles can use the .PARALLEL target to declare otherwise.
208 .NO_PARALLEL:

210 # For stylistic checks
211 #
212 # Note that the X and C checks are not used at this time and may need
213 # modification when they are actually used.
214 #
215 CSTYLE=         $(ONBLD_TOOLS)/bin/cstyle
216 CSTYLE_TAIL=
217 HDRCHK=         $(ONBLD_TOOLS)/bin/hdrchk
218 HDRCHK_TAIL=
219 JSTYLE=         $(ONBLD_TOOLS)/bin/jstyle

221 DOT_H_CHECK=    \
222         @$(ECHO) "checking $<"; $(CSTYLE) $< $(CSTYLE_TAIL); \
223         $(HDRCHK) $< $(HDRCHK_TAIL)

225 DOT_X_CHECK=    \
226         @$(ECHO) "checking $<"; $(RPCGEN) -C -h $< | $(CSTYLE) $(CSTYLE_TAIL); \
227         $(RPCGEN) -C -h $< | $(HDRCHK) $< $(HDRCHK_TAIL)

229 DOT_C_CHECK=    \
230         @$(ECHO) "checking $<"; $(CSTYLE) $< $(CSTYLE_TAIL)

232 MANIFEST_CHECK= \
233         @$(ECHO) "checking $<"; \
234         SVCCFG_DTD=$(SRC)/cmd/svc/dtd/service_bundle.dtd.1 \
235         SVCCFG_REPOSITORY=$(SRC)/cmd/svc/seed/global.db \
236         SVCCFG_CONFIGD_PATH=$(SRC)/cmd/svc/configd/svc.configd-native \
237         $(SRC)/cmd/svc/svccfg/svccfg-native validate $<

239 #
240 # IMPORTANT:: If you change any of INS.file, INS.dir, INS.rename,
241 # INS.link or INS.symlink here, then you must also change the
242 # corresponding override definitions in $CLOSED/Makefile.tonic.
243 # If you do not do this, then the closedbins build for the OpenSolaris
244 # community will break. PS, the gatekeepers will be upset too.
245 INS.file=       $(RM) $@; $(INS) -s -m $(FILEMODE) -f $(@D) $<
246 INS.dir=        $(INS) -s -d -m $(DIRMODE) $@
247 # installs and renames at once
248 #
249 INS.rename=     $(INS.file); $(MV) $(@D)/$(<F) $@

251 # install a link
```

```
252 INSLINKTARGET=  $<
253 INS.link=       $(RM) $@; $(LN) $(INSLINKTARGET) $@
254 INS.symlink=    $(RM) $@; $(SYMLINK) $(INSLINKTARGET) $@

256 #
257 # Python bakes the mtime of the .py file into the compiled .pyc and
258 # rebuilds if the baked-in mtime != the mtime of the source file
259 # (rather than only if it's less than), thus when installing python
260 # files we must make certain to not adjust the mtime of the source
261 # (.py) file.
262 #
263 INS.pyfile=     $(INS.file); $(TOUCH) -r $< $@

265 # MACH must be set in the shell environment per uname -p on the build host
266 # More specific architecture variables should be set in lower makefiles.
267 #
268 # MACH64 is derived from MACH, and BUILD64 is set to '#' for
269 # architectures on which we do not build 64-bit versions.
270 # (There are no such architectures at the moment.)
271 #
272 # Set BUILD64=# in the environment to disable 64-bit amd64
273 # builds on i386 machines.

275 MACH64_1=       $(MACH:sparc=sparcv9)
276 MACH64=         $(MACH64_1:i386=amd64)

278 MACH32_1=       $(MACH:sparc=sparcv7)
279 MACH32=         $(MACH32_1:i386=i86)

281 sparc_BUILD64=
282 i386_BUILD64=
283 BUILD64=        $($(MACH)_BUILD64)

285 #
286 # C compiler mode. Future compilers may change the default on us,
287 # so force extended ANSI mode globally. Lower level makefiles can
288 # override this by setting CCMODE.
289 #
290 CCMODE=                 -Xa
291 CCMODE64=               -Xa

293 #
294 # C compiler verbose mode. This is so we can enable it globally,
295 # but turn it off in the lower level makefiles of things we cannot
296 # (or aren't going to) fix.
297 #
298 CCVERBOSE=              -v

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

304 # set this to the secret flag "-Wc,-Qiselect-regsym=0" to disable register
305 # symbols (used to detect conflicts between objects that use global registers)
306 # we disable this now for safety, and because genunix doesn't link with
307 # this feature (the v9 default) enabled.
308 #
309 # REGSYM is separate since the C++ driver syntax is different.
310 CCREGSYM=               -Wc,-Qiselect-regsym=0
311 CCCREGSYM=              -Qoption cg -Qiselect-regsym=0

313 # Prevent the removal of static symbols by the SPARC code generator (cg).
314 # The x86 code generator (ube) does not remove such symbols and as such
315 # using this workaround is not applicable for x86.
316 #
317 CCSTATICSYM=            -Wc,-Qassembler-ounrefsym=0
```



```
318 #
319 # generate 32-bit addresses in the v9 kernel. Saves memory.
320 CCABS32=                -Wc,-xcode=abs32
321 #
322 # generate v9 code which tolerates callers using the v7 ABI, for the sake of
323 # system calls.
324 CC32BITCALLERS=         -_gcc=-massume-32bit-callers

326 # GCC, especially, is increasingly beginning to auto-inline functions and
327 # sadly does so separately not under the general -fno-inline-functions
328 # Additionally, we wish to prevent optimisations which cause GCC to clone
329 # functions -- in particular, these may cause unhelpful symbols to be
330 # emitted instead of function names
331 CCNOAUTOINLINE= -_gcc=-fno-inline-small-functions \
332         -_gcc=-fno-inline-functions-called-once \
333         -_gcc=-fno-ipa-cp

335 # One optimization the compiler might perform is to turn this:
336 #       #pragma weak foo
337 #       extern int foo;
338 #       if (&foo)
339 #               foo = 5;
340 # into
341 #       foo = 5;
342 # Since we do some of this (foo might be referenced in common kernel code
343 # but provided only for some cpu modules or platforms), we disable this
344 # optimization.
345 #
346 sparc_CCUNBOUND = -Wd,-xsafe=unboundsym
347 i386_CCUNBOUND  =
348 CCUNBOUND       = $($(MACH)_CCUNBOUND)

350 #
351 # compiler '-xarch' flag. This is here to centralize it and make it
352 # overridable for testing.
353 sparc_XARCH=    -m32
354 sparcv9_XARCH=  -m64
355 i386_XARCH=
356 amd64_XARCH=    -m64 -Ui386 -U__i386

358 # assembler '-xarch' flag.  Different from compiler '-xarch' flag.
359 sparc_AS_XARCH=         -xarch=v8plus
360 sparcv9_AS_XARCH=       -xarch=v9
361 i386_AS_XARCH=
362 amd64_AS_XARCH=         -xarch=amd64 -P -Ui386 -U__i386

364 #
365 # These flags define what we need to be 'standalone' i.e. -not- part
366 # of the rather more cosy userland environment.  This basically means
367 # the kernel.
368 #
369 # XX64  future versions of gcc will make -mcmodel=kernel imply -mno-red-zone
370 #
371 sparc_STAND_FLAGS=      -_gcc=-ffreestanding
372 sparcv9_STAND_FLAGS=    -_gcc=-ffreestanding
373 # Disabling MMX also disables 3DNow, disabling SSE also disables all later
374 # additions to SSE (SSE2, AVX ,etc.)
375 NO_SIMD=                -_gcc=-mno-mmx -_gcc=-mno-sse
376 i386_STAND_FLAGS=       -_gcc=-ffreestanding $(NO_SIMD)
377 amd64_STAND_FLAGS=      -xmodel=kernel $(NO_SIMD)

379 SAVEARGS=               -Wu,-save_args
380 amd64_STAND_FLAGS       += $(SAVEARGS)

382 STAND_FLAGS_32 = $($(MACH)_STAND_FLAGS)
383 STAND_FLAGS_64 = $($(MACH64)_STAND_FLAGS)
```

```
385 #
386 # disable the incremental linker
387 ILDOFF=                 -xildoff
388 #
389 XDEPEND=                -xdepend
390 XFFLAG=                 -xF=%all
391 XESS=                   -xs
392 XSTRCONST=              -xstrconst

394 #
395 # turn warnings into errors (C)
396 CERRWARN = -errtags=yes -errwarn=%all
397 CERRWARN += -erroff=E_EMPTY_TRANSLATION_UNIT
398 CERRWARN += -erroff=E_STATEMENT_NOT_REACHED

400 CERRWARN += -_gcc=-Wno-missing-braces
401 CERRWARN += -_gcc=-Wno-sign-compare
402 CERRWARN += -_gcc=-Wno-unknown-pragmas
403 CERRWARN += -_gcc=-Wno-unused-parameter
404 CERRWARN += -_gcc=-Wno-missing-field-initializers

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

409 # DEBUG v. -nd make for frequent unused variables, empty conditions, etc. in
410 # -nd builds
411 $(RELEASE_BUILD)CERRWARN += -_gcc=-Wno-unused
412 $(RELEASE_BUILD)CERRWARN += -_gcc=-Wno-empty-body

414 #
415 # turn warnings into errors (C++)
416 CCERRWARN=              -xwe

418 # C99 mode
419 C99_ENABLE=     -xc99=%all
420 C99_DISABLE=    -xc99=%none
421 C99MODE=        $(C99_DISABLE)
422 C99LMODE=       $(C99MODE:-xc99%=-Xc99%)

424 # In most places, assignments to these macros should be appended with +=
425 # (CPPFLAGS.master allows values to be prepended to CPPFLAGS).
426 sparc_CFLAGS=   $(sparc_XARCH) $(CCSTATICSYM)
427 sparcv9_CFLAGS= $(sparcv9_XARCH) -dalign $(CCVERBOSE) $(V9ABIWARN) $(CCREGSYM) \
428                 $(CCSTATICSYM)
429 i386_CFLAGS=    $(i386_XARCH)
430 amd64_CFLAGS=   $(amd64_XARCH)

432 sparc_ASFLAGS=  $(sparc_AS_XARCH)
433 sparcv9_ASFLAGS=$(sparcv9_AS_XARCH)
434 i386_ASFLAGS=   $(i386_AS_XARCH)
435 amd64_ASFLAGS=  $(amd64_AS_XARCH)

437 #
438 sparc_COPTFLAG=         -xO3
439 sparcv9_COPTFLAG=       -xO3
440 i386_COPTFLAG=          -O
441 amd64_COPTFLAG=         -xO3

443 COPTFLAG= $($(MACH)_COPTFLAG)
444 COPTFLAG64= $($(MACH64)_COPTFLAG)

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



```
450 # Direct the Sun Studio compiler to use a static globalization prefix based on t
451 # name of the module rather than something unique. Otherwise, objects
452 # will not build deterministically, as subsequent compilations of identical
453 # source will yeild objects that always look different.
454 #
455 # In the same spirit, this will also remove the date from the N_OPT stab.
456 CGLOBALSTATIC= -W0,-xglobalstatic

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

462 #
463 # Default debug format for Sun Studio 11 is dwarf, so force it to
464 # generate stabs.
465 #
466 DEBUGFORMAT=    -xdebugformat=stabs

468 #
469 # Flags used to build in debug mode for ctf generation.  Bugs in the Devpro
470 # compilers currently prevent us from building with cc-emitted DWARF.
471 #
472 CTF_FLAGS_sparc = -g -Wc,-Qiselect-T1 $(C99MODE) $(CNOGLOBAL) $(CDWARFSTR)
473 CTF_FLAGS_i386  = -g $(C99MODE) $(CNOGLOBAL) $(CDWARFSTR)

475 CTF_FLAGS_sparcv9       = $(CTF_FLAGS_sparc)
476 CTF_FLAGS_amd64         = $(CTF_FLAGS_i386)

478 # Sun Studio produces broken userland code when saving arguments.
479 $(__GNUC)CTF_FLAGS_amd64 += $(SAVEARGS)

481 CTF_FLAGS_32    = $(CTF_FLAGS_$(MACH)) $(DEBUGFORMAT)
482 CTF_FLAGS_64    = $(CTF_FLAGS_$(MACH64)) $(DEBUGFORMAT)
483 CTF_FLAGS       = $(CTF_FLAGS_32)

485 #
486 # Flags used with genoffsets
487 #
488 GOFLAGS = -_noecho \
489         $(CALLSYMS) \
490         $(CDWARFSTR)

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

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

498 #
499 # tradeoff time for space (smaller is better)
500 #
501 sparc_SPACEFLAG         = -xspace -W0,-Lt
502 sparcv9_SPACEFLAG       = -xspace -W0,-Lt
503 i386_SPACEFLAG          = -xspace
504 amd64_SPACEFLAG         =

506 SPACEFLAG               = $($(MACH)_SPACEFLAG)
507 SPACEFLAG64             = $($(MACH64)_SPACEFLAG)

509 #
510 # The Sun Studio 11 compiler has changed the behaviour of integer
511 # wrap arounds and so a flag is needed to use the legacy behaviour
512 # (without this flag panics/hangs could be exposed within the source).
513 #
514 sparc_IROPTFLAG         = -W2,-xwrap_int
515 sparcv9_IROPTFLAG       = -W2,-xwrap_int
```

```
516 i386_IROPTFLAG          =
517 amd64_IROPTFLAG         =

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

522 sparc_XREGSFLAG         = -xregs=no%appl
523 sparcv9_XREGSFLAG       = -xregs=no%appl
524 i386_XREGSFLAG          =
525 amd64_XREGSFLAG         =

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

530 CFLAGS=         $(COPTFLAG) $($(MACH)_CFLAGS) $(SPACEFLAG) $(CCMODE) \
531                 $(ILDOFF) $(CERRWARN) $(C99MODE) $(CCUNBOUND) $(IROPTFLAG) \
532                 $(CGLOBALSTATIC) $(CCNOAUTOINLINE)
533 CFLAGS64=       $(COPTFLAG64) $($(MACH64)_CFLAGS) $(SPACEFLAG64) $(CCMODE64) \
534                 $(ILDOFF) $(CERRWARN) $(C99MODE) $(CCUNBOUND) $(IROPTFLAG64) \
535                 $(CGLOBALSTATIC) $(CCNOAUTOINLINE)
536 #
537 # Flags that are used to build parts of the code that are subsequently
538 # run on the build machine (also known as the NATIVE_BUILD).
539 #
540 NATIVE_CFLAGS=  $(COPTFLAG) $($(NATIVE_MACH)_CFLAGS) $(CCMODE) \
541                 $(ILDOFF) $(CERRWARN) $(C99MODE) $($(NATIVE_MACH)_CCUNBOUND) \
542                 $(IROPTFLAG) $(CGLOBALSTATIC) $(CCNOAUTOINLINE)

544 DTEXTDOM=-DTEXT_DOMAIN=\"$(TEXT_DOMAIN)\"       # For messaging.
545 DTS_ERRNO=-D_TS_ERRNO
546 CPPFLAGS.master=$(DTEXTDOM) $(DTS_ERRNO) \
547         $(ENVCPPFLAGS1) $(ENVCPPFLAGS2) $(ENVCPPFLAGS3) $(ENVCPPFLAGS4)
548 CPPFLAGS.native=$(ENVCPPFLAGS1) $(ENVCPPFLAGS2) $(ENVCPPFLAGS3) $(ENVCPPFLAGS4)
549 CPPFLAGS=       $(CPPFLAGS.master)
550 AS_CPPFLAGS=    $(CPPFLAGS.master)
551 JAVAFLAGS=      -deprecation

553 #
554 # For source message catalogue
555 #
556 .SUFFIXES: $(SUFFIXES) .i .po
557 MSGROOT= $(ROOT)/catalog
558 MSGDOMAIN= $(MSGROOT)/$(TEXT_DOMAIN)
559 MSGDOMAINPOFILE = $(MSGDOMAIN)/$(POFILE)
560 DCMSGDOMAIN= $(MSGROOT)/LC_TIME/$(TEXT_DOMAIN)
561 DCMSGDOMAINPOFILE = $(DCMSGDOMAIN)/$(DCFILE:.dc=.po)

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

575 #
576 # This is overwritten by local Makefile when PROG is a list.
577 #
578 POFILE= $(PROG).po

580 sparc_CCFLAGS=          -cg92 -compat=4 \
581                         -Qoption ccfe -messages=no%anachronism \
```



```
582                         $(CCERRWARN)
583 sparcv9_CCFLAGS=        $(sparcv9_XARCH) -dalign -compat=5 \
584                         -Qoption ccfe -messages=no%anachronism \
585                         -Qoption ccfe -features=no%conststrings \
586                         $(CCCREGSYM) \
587                         $(CCERRWARN)
588 i386_CCFLAGS=           -compat=4 \
589                         -Qoption ccfe -messages=no%anachronism \
590                         -Qoption ccfe -features=no%conststrings \
591                         $(CCERRWARN)
592 amd64_CCFLAGS=          $(amd64_XARCH) -compat=5 \
593                         -Qoption ccfe -messages=no%anachronism \
594                         -Qoption ccfe -features=no%conststrings \
595                         $(CCERRWARN)

597 sparc_CCOPTFLAG=        -O
598 sparcv9_CCOPTFLAG=      -O
599 i386_CCOPTFLAG=         -O
600 amd64_CCOPTFLAG=        -O

602 CCOPTFLAG=      $($(MACH)_CCOPTFLAG)
603 CCOPTFLAG64=    $($(MACH64)_CCOPTFLAG)
604 CCFLAGS=        $(CCOPTFLAG) $($(MACH)_CCFLAGS)
605 CCFLAGS64=      $(CCOPTFLAG64) $($(MACH64)_CCFLAGS)

607 #
608 #
609 #
610 ELFWRAP_FLAGS   =
611 ELFWRAP_FLAGS64 =       -64

613 #
614 # Various mapfiles that are used throughout the build, and delivered to
615 # /usr/lib/ld.
616 #
617 MAPFILE.NED_i386 =      $(SRC)/common/mapfiles/common/map.noexdata
618 MAPFILE.NED_sparc =
619 MAPFILE.NED =           $(MAPFILE.NED_$(MACH))
620 MAPFILE.PGA =           $(SRC)/common/mapfiles/common/map.pagealign
621 MAPFILE.NES =           $(SRC)/common/mapfiles/common/map.noexstk
622 MAPFILE.FLT =           $(SRC)/common/mapfiles/common/map.filter
623 MAPFILE.LEX =           $(SRC)/common/mapfiles/common/map.lex.yy

625 #
626 # Generated mapfiles that are compiler specific, and used throughout the
627 # build.  These mapfiles are not delivered in /usr/lib/ld.
628 #
629 MAPFILE.NGB_sparc=      $(SRC)/common/mapfiles/gen/sparc_cc_map.noexeglobs
630 $(__GNUC64)MAPFILE.NGB_sparc= \
631                         $(SRC)/common/mapfiles/gen/sparc_gcc_map.noexeglobs
632 MAPFILE.NGB_sparcv9=    $(SRC)/common/mapfiles/gen/sparcv9_cc_map.noexeglobs
633 $(__GNUC64)MAPFILE.NGB_sparcv9= \
634                         $(SRC)/common/mapfiles/gen/sparcv9_gcc_map.noexeglobs
635 MAPFILE.NGB_i386=       $(SRC)/common/mapfiles/gen/i386_cc_map.noexeglobs
636 $(__GNUC64)MAPFILE.NGB_i386= \
637                         $(SRC)/common/mapfiles/gen/i386_gcc_map.noexeglobs
638 MAPFILE.NGB_amd64=      $(SRC)/common/mapfiles/gen/amd64_cc_map.noexeglobs
639 $(__GNUC64)MAPFILE.NGB_amd64= \
640                         $(SRC)/common/mapfiles/gen/amd64_gcc_map.noexeglobs
641 MAPFILE.NGB =           $(MAPFILE.NGB_$(MACH))

643 #
644 # A generic interface mapfile name, used by various dynamic objects to define
645 # the interfaces and interposers the object must export.
646 #
647 MAPFILE.INT =           mapfile-intf
```

```
649 #
650 # LDLIBS32 can be set in the environment to override the following assignment.
651 # LDLIBS64 can be set to override the assignment made in Makefile.master.64.
652 # These environment settings make sure that no libraries are searched outside
653 # of the local workspace proto area:
654 #       LDLIBS32=-YP,$ROOT/lib:$ROOT/usr/lib
655 #       LDLIBS64=-YP,$ROOT/lib/$MACH64:$ROOT/usr/lib/$MACH64
656 #
657 LDLIBS32 =      $(ENVLDLIBS1) $(ENVLDLIBS2) $(ENVLDLIBS3)
658 LDLIBS.cmd =    $(LDLIBS32)
659 LDLIBS.lib =    $(LDLIBS32)
660 #
661 # Define compilation macros.
662 #
663 COMPILE.c=      $(CC) $(CFLAGS) $(CPPFLAGS) -c
664 COMPILE64.c=    $(CC) $(CFLAGS64) $(CPPFLAGS) -c
665 COMPILE.cc=     $(CCC) $(CCFLAGS) $(CPPFLAGS) -c
666 COMPILE64.cc=   $(CCC) $(CCFLAGS64) $(CPPFLAGS) -c
667 COMPILE.s=      $(AS) $(ASFLAGS) $(AS_CPPFLAGS)
668 COMPILE64.s=    $(AS) $(ASFLAGS) $($(MACH64)_AS_XARCH) $(AS_CPPFLAGS)
669 COMPILE.d=      $(DTRACE) -G -32
670 COMPILE64.d=    $(DTRACE) -G -64
671 COMPILE.b=      $(ELFWRAP) $(ELFWRAP_FLAGS$(CLASS))
672 COMPILE64.b=    $(ELFWRAP) $(ELFWRAP_FLAGS$(CLASS))

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

677 #
678 # Link time macros
679 #
680 CCNEEDED                = -lC
681 CCEXTNEEDED             = -lCrun -lCstd
682 $(__GNUC)CCNEEDED       = -L$(GCCLIBDIR) -R$(GCCLIBDIR) -lstdc++ -lgcc_s
683 $(__GNUC)CCEXTNEEDED    = $(CCNEEDED)

685 LINK.c=         $(CC) $(CFLAGS) $(CPPFLAGS) $(LDFLAGS)
686 LINK64.c=       $(CC) $(CFLAGS64) $(CPPFLAGS) $(LDFLAGS)
687 NORUNPATH=      -norunpath -nolib
688 LINK.cc=        $(CCC) $(CCFLAGS) $(CPPFLAGS) $(NORUNPATH) \
689                 $(LDFLAGS) $(CCNEEDED)
690 LINK64.cc=      $(CCC) $(CCFLAGS64) $(CPPFLAGS) $(NORUNPATH) \
691                 $(LDFLAGS) $(CCNEEDED)

693 #
694 # lint macros
695 #
696 # Note that the undefine of __PRAGMA_REDEFINE_EXTNAME can be removed once
697 # ON is built with a version of lint that has the fix for 4484186.
698 #
699 ALWAYS_LINT_DEFS =      -errtags=yes -s
700 ALWAYS_LINT_DEFS +=     -erroff=E_PTRDIFF_OVERFLOW
701 ALWAYS_LINT_DEFS +=     -erroff=E_ASSIGN_NARROW_CONV
702 ALWAYS_LINT_DEFS +=     -U__PRAGMA_REDEFINE_EXTNAME
703 ALWAYS_LINT_DEFS +=     $(C99LMODE)
704 ALWAYS_LINT_DEFS +=     -errsecurity=$(SECLEVEL)
705 ALWAYS_LINT_DEFS +=     -erroff=E_SEC_CREAT_WITHOUT_EXCL
706 ALWAYS_LINT_DEFS +=     -erroff=E_SEC_FORBIDDEN_WARN_CREAT
707 # XX64 -- really only needed for amd64 lint
708 ALWAYS_LINT_DEFS +=     -erroff=E_ASSIGN_INT_TO_SMALL_INT
709 ALWAYS_LINT_DEFS +=     -erroff=E_CAST_INT_CONST_TO_SMALL_INT
710 ALWAYS_LINT_DEFS +=     -erroff=E_CAST_INT_TO_SMALL_INT
711 ALWAYS_LINT_DEFS +=     -erroff=E_CAST_TO_PTR_FROM_INT
712 ALWAYS_LINT_DEFS +=     -erroff=E_COMP_INT_WITH_LARGE_INT
713 ALWAYS_LINT_DEFS +=     -erroff=E_INTEGRAL_CONST_EXP_EXPECTED
```



```
714 ALWAYS_LINT_DEFS +=     -erroff=E_PASS_INT_TO_SMALL_INT
715 ALWAYS_LINT_DEFS +=     -erroff=E_PTR_CONV_LOSES_BITS

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

721 SECLEVEL=       core
722 LINT.c=         $(LINT) $(ONLY_LINT_DEFS) $(LINTFLAGS) $(CPPFLAGS) \
723                 $(ALWAYS_LINT_DEFS)
724 LINT64.c=       $(LINT) $(ONLY_LINT_DEFS) $(LINTFLAGS64) $(CPPFLAGS) \
725                 $(ALWAYS_LINT_DEFS)
726 LINT.s=         $(LINT.c)

728 # For some future builds, NATIVE_MACH and MACH might be different.
729 # Therefore, NATIVE_MACH needs to be redefined in the
730 # environment as ‘uname -p‘ to override this macro.
731 #
732 # For now at least, we cross-compile amd64 on i386 machines.
733 NATIVE_MACH=    $(MACH:amd64=i386)

735 # Define native compilation macros
736 #

738 # Base directory where compilers are loaded.
739 # Defined here so it can be overridden by developer.
740 #
741 SPRO_ROOT=              $(BUILD_TOOLS)/SUNWspro
742 SPRO_VROOT=             $(SPRO_ROOT)/SS12
743 GNU_ROOT=               $(SFW_ROOT)

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

752 # Specify platform compiler versions for languages
753 # that we use (currently only c and c++).
754 #
755 sparc_CC=               $(ONBLD_TOOLS)/bin/$(MACH)/cw -_cc
756 $(__GNUC)sparc_CC=      $(ONBLD_TOOLS)/bin/$(MACH)/cw -_gcc
757 sparc_CCC=              $(ONBLD_TOOLS)/bin/$(MACH)/cw -_CC
758 $(__GNUC)sparc_CCC=     $(ONBLD_TOOLS)/bin/$(MACH)/cw -_g++
759 sparc_CPP=              /usr/ccs/lib/cpp
760 sparc_AS=               /usr/ccs/bin/as -xregsym=no
761 sparc_LD=               /usr/ccs/bin/ld
762 sparc_LINT=             $(SPRO_ROOT)/sunstudio12.1/bin/lint

764 sparcv9_CC=             $(ONBLD_TOOLS)/bin/$(MACH)/cw -_cc
765 $(__GNUC64)sparcv9_CC=  $(ONBLD_TOOLS)/bin/$(MACH)/cw -_gcc
766 sparcv9_CCC=            $(ONBLD_TOOLS)/bin/$(MACH)/cw -_CC
767 $(__GNUC64)sparcv9_CCC= $(ONBLD_TOOLS)/bin/$(MACH)/cw -_g++
768 sparcv9_CPP=            /usr/ccs/lib/cpp
769 sparcv9_AS=             /usr/ccs/bin/as -xregsym=no
770 sparcv9_LD=             /usr/ccs/bin/ld
771 sparcv9_LINT=           $(SPRO_ROOT)/sunstudio12.1/bin/lint

773 i386_CC=               $(ONBLD_TOOLS)/bin/$(MACH)/cw -_cc
774 $(__GNUC)i386_CC=       $(ONBLD_TOOLS)/bin/$(MACH)/cw -_gcc
775 i386_CCC=              $(ONBLD_TOOLS)/bin/$(MACH)/cw -_CC
776 $(__GNUC)i386_CCC=      $(ONBLD_TOOLS)/bin/$(MACH)/cw -_g++
777 i386_CPP=              /usr/ccs/lib/cpp
778 i386_AS=               /usr/ccs/bin/as
779 $(__GNUC)i386_AS=       $(ONBLD_TOOLS)/bin/$(MACH)/aw
```

```
780 i386_LD=                /usr/ccs/bin/ld
781 i386_LINT=              $(SPRO_ROOT)/sunstudio12.1/bin/lint

783 amd64_CC=               $(ONBLD_TOOLS)/bin/$(MACH)/cw -_cc
784 $(__GNUC64)amd64_CC=    $(ONBLD_TOOLS)/bin/$(MACH)/cw -_gcc
785 amd64_CCC=              $(ONBLD_TOOLS)/bin/$(MACH)/cw -_CC
786 $(__GNUC64)amd64_CCC=   $(ONBLD_TOOLS)/bin/$(MACH)/cw -_g++
787 amd64_CPP=              /usr/ccs/lib/cpp
788 amd64_AS=               $(ONBLD_TOOLS)/bin/$(MACH)/aw
789 amd64_LD=               /usr/ccs/bin/ld
790 amd64_LINT=             $(SPRO_ROOT)/sunstudio12.1/bin/lint

792 NATIVECC=              $($(NATIVE_MACH)_CC)
793 NATIVECCC=             $($(NATIVE_MACH)_CCC)
794 NATIVECPP=             $($(NATIVE_MACH)_CPP)
795 NATIVEAS=              $($(NATIVE_MACH)_AS)
796 NATIVELD=              $($(NATIVE_MACH)_LD)
797 NATIVELINT=            $($(NATIVE_MACH)_LINT)

799 #
800 # Makefile.master.64 overrides these settings
801 #
802 CC=                    $(NATIVECC)
803 CCC=                   $(NATIVECCC)
804 CPP=                   $(NATIVECPP)
805 AS=                    $(NATIVEAS)
806 LD=                    $(NATIVELD)
807 LINT=                  $(NATIVELINT)

809 # The real compilers used for this build
810 CW_CC_CMD=             $(CC) -_compiler
811 CW_CCC_CMD=            $(CCC) -_compiler
812 REAL_CC=               $(CW_CC_CMD:sh)
813 REAL_CCC=              $(CW_CCC_CMD:sh)

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

818 BDIRECT=        -Bdirect
819 BDYNAMIC=       -Bdynamic
820 BLOCAL=         -Blocal
821 BNODIRECT=      -Bnodirect
822 BREDUCE=        -Breduce
823 BSTATIC=        -Bstatic

825 ZDEFS=          -zdefs
826 ZDIRECT=        -zdirect
827 ZIGNORE=        -zignore
828 ZINITFIRST=     -zinitfirst
829 ZINTERPOSE=     -zinterpose
830 ZLAZYLOAD=      -zlazyload
831 ZLOADFLTR=      -zloadfltr
832 ZMULDEFS=       -zmuldefs
833 ZNODEFAULTLIB=  -znodefaultlib
834 ZNODEFS=        -znodefs
835 ZNODELETE=      -znodelete
836 ZNODLOPEN=      -znodlopen
837 ZNODUMP=        -znodump
838 ZNOLAZYLOAD=    -znolazyload
839 ZNOLDYNSYM=     -znoldynsym
840 ZNORELOC=       -znoreloc
841 ZNOVERSION=     -znoversion
842 ZRECORD=        -zrecord
843 ZREDLOCSYM=     -zredlocsym
844 ZTEXT=          -ztext
845 ZVERBOSE=       -zverbose
```



```
847 GSHARED=        -G
848 CCMT=           -mt

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

853 sparc_C_PICFLAGS =      -K pic
854 sparcv9_C_PICFLAGS =    -K pic
855 i386_C_PICFLAGS =       -K pic
856 amd64_C_PICFLAGS =      -K pic
857 C_PICFLAGS =            $($(MACH)_C_PICFLAGS)
858 C_PICFLAGS64 =          $($(MACH64)_C_PICFLAGS)

860 sparc_C_BIGPICFLAGS =   -K PIC
861 sparcv9_C_BIGPICFLAGS = -K PIC
862 i386_C_BIGPICFLAGS =    -K PIC
863 amd64_C_BIGPICFLAGS =   -K PIC
864 C_BIGPICFLAGS =         $($(MACH)_C_BIGPICFLAGS)
865 C_BIGPICFLAGS64 =       $($(MACH64)_C_BIGPICFLAGS)

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

875 AS_PICFLAGS=            $(C_PICFLAGS)
876 AS_BIGPICFLAGS=         $(C_BIGPICFLAGS)

878 #
879 # Default label for CTF sections
880 #
881 CTFCVTFLAGS=            -i -L VERSION

883 #
884 # Override to pass module-specific flags to ctfmerge.  Currently used
885 # only by krtld to turn on fuzzy matching.
886 #
887 CTFMRGFLAGS=

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

891 ELFSIGN_O=      $(TRUE)
892 ELFSIGN_CRYPTO= $(ELFSIGN_O)
893 ELFSIGN_OBJECT= $(ELFSIGN_O)
902 $(EXPORT_RELEASE_BUILD)ELFSIGN_O =      $(ELFSIGN)
903 $(EXPORT_RELEASE_BUILD)ELFSIGN_CFNAME = SUNWosnetCF
904 $(EXPORT_RELEASE_BUILD)ELFSIGN_KEY =    \
905                         $(CLOSED)/cmd/cmd-crypto/etc/keys/$(ELFSIGN_CFNAME)
906 $(EXPORT_RELEASE_BUILD)ELFSIGN_CERT=    \
907                         $(CLOSED)/cmd/cmd-crypto/etc/certs/$(ELFSIGN_CFNAME)
908 $(EXPORT_RELEASE_BUILD)ELFSIGN_SENAME = SUNWosnetSE
909 $(EXPORT_RELEASE_BUILD)ELFSIGN_SEKEY =  \
910                         $(CLOSED)/cmd/cmd-crypto/etc/keys/$(ELFSIGN_SENAME)
911 $(EXPORT_RELEASE_BUILD)ELFSIGN_SECERT=  \
912                         $(CLOSED)/cmd/cmd-crypto/etc/certs/$(ELFSIGN_SENAME)
913 $(EXPORT_RELEASE_BUILD)ELFSIGN_CRYPTO=  $(ELFSIGN_O) sign \
914                         $(ELFSIGN_FORMAT_OPTION) \
915                         -k $(ELFSIGN_KEY) -c $(ELFSIGN_CERT) -e $@
916 $(EXPORT_RELEASE_BUILD)ELFSIGN_OBJECT=  $(ELFSIGN_O) sign \
917                         $(ELFSIGN_FORMAT_OPTION) \
918                         -k $(ELFSIGN_SEKEY) -c $(ELFSIGN_SECERT) -e $@
```

```
# Rules (normally from make.rules) and macros which are used for post
# processing files. Normally, these do stripping of the comment section
# automatically.
#    RELEASE_CM:        Should be editted to reflect the release.
#    POST_PROCESS_O:    Post-processing for '.o' files.
#    POST_PROCESS_A:    Post-processing for '.a' files (currently null).
#    POST_PROCESS_SO:   Post-processing for '.so' files.
#    POST_PROCESS:      Post-processing for executable files (no suffix).
# Note that these macros are not completely generalized as they are to be
# used with the file name to be processed following.
#
# It is left as an exercise to Release Engineering to embellish the generation
# of the release comment string.
#
#       If this is a standard development build:
#               compress the comment section (mcs -c)
#               add the standard comment (mcs -a $(RELEASE_CM))
#               add the development specific comment (mcs -a $(DEV_CM))
#
#       If this is an installation build:
#               delete the comment section (mcs -d)
#               add the standard comment (mcs -a $(RELEASE_CM))
#               add the development specific comment (mcs -a $(DEV_CM))
#
#       If this is an release build:
#               delete the comment section (mcs -d)
#               add the standard comment (mcs -a $(RELEASE_CM))
#
# The following list of macros are used in the definition of RELEASE_CM
# which is used to label all binaries in the build:
#
#       RELEASE         Specific release of the build, eg: 5.2
#       RELEASE_MAJOR   Major version number part of $(RELEASE)
#       RELEASE_MINOR   Minor version number part of $(RELEASE)
#       VERSION         Version of the build (alpha, beta, Generic)
#       PATCHID         If this is a patch this value should contain
#                       the patchid value (eg: "Generic 100832-01"), otherwise
#                       it will be set to $(VERSION)
#       RELEASE_DATE    Date of the Release Build
#       PATCH_DATE      Date the patch was created, if this is blank it
#                       will default to the RELEASE_DATE
#
RELEASE_MAJOR=  5
RELEASE_MINOR=  11
RELEASE=        $(RELEASE_MAJOR).$(RELEASE_MINOR)
VERSION=        SunOS Development
PATCHID=        $(VERSION)
RELEASE_DATE=   release date not set
PATCH_DATE=     $(RELEASE_DATE)
RELEASE_CM=     "@($(POUND_SIGN))SunOS $(RELEASE) $(PATCHID) $(PATCH_DATE)"
DEV_CM=         "@($(POUND_SIGN))SunOS Internal Development: non-nightly build"

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

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

POST_PROCESS_O=         $(PROCESS_COMMENT) $@
POST_PROCESS_A=
POST_PROCESS_SO=        $(PROCESS_COMMENT) $@ ; $(STRIP_STABS) ; \
                        $(ELFSIGN_OBJECT)
POST_PROCESS=           $(PROCESS_COMMENT) $@ ; $(STRIP_STABS) ; \
                        $(ELFSIGN_OBJECT)
```



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

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

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

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

```
1027 .l:
1028         $(RM) $*.c
1029         $(LEX.l) $< > $*.c
1030         $(LINK.c) -o $@ $*.c -ll $(LDLIBS)
1031         $(POST_PROCESS)
1032         $(RM) $*.c
1033 .l.o:
1034         $(RM) $*.c
1035         $(LEX.l) $< > $*.c
1036         $(COMPILE.c) -o $@ $*.c $(CTFCONVERT_HOOK)
1037         $(POST_PROCESS_O)
1038         $(RM) $*.c

1040 .bin.o:
1041         $(COMPILE.b) -o $@ $<
1042         $(POST_PROCESS_O)

1044 .java.class:
1045         $(COMPILE.java) $<

1047 # Bourne and Korn shell script message catalog build rules.
1048 # We extract all gettext strings with sed(1) (being careful to permit
1049 # multiple gettext strings on the same line), weed out the dups, and
1050 # build the catalogue with awk(1).

1052 .sh.po .ksh.po:
1053         $(SED) -n -e ":a"                                       \
1054                   -e "h"                                        \
1055                   -e "s/.*gettext *\(\"[^\"]*\"\).*/\1/p"       \
1056                   -e "x"                                        \
1057                   -e "s/\(.*\)gettext *\"[^\"]*\"\(.*\)/\1\2/"  \
1058                   -e "t a"                                      \
1059                 $< | sort -u | awk '{ print "msgid\t" $$0 "\nmsgstr" }' > $@

1061 #
1062 # Python and Perl executable and message catalog build rules.
1063 #
1064 .SUFFIXES: .pl .pm .py .pyc

1066 .pl:
1067         $(RM) $@;
1068         $(SED) -e "s@TEXT_DOMAIN@\"$(TEXT_DOMAIN)\"@" $< > $@;
1069         $(CHMOD) +x $@

1071 .py:
1072         $(RM) $@; $(CAT) $< > $@; $(CHMOD) +x $@

1074 .py.pyc:
1075         $(RM) $@
1076         $(PYTHON) -mpy_compile $<
1077         @[ $(<)c = $@ ] || $(MV) $(<)c $@

1079 .py.po:
1080         $(GNUXGETTEXT) $(GNUXGETFLAGS) -d $(<F:%.py=%) $< ;

1082 .pl.po .pm.po:
1083         $(XGETTEXT) $(XGETFLAGS) -d $(<F) $< ;
1084         $(RM)   $@ ;
1085         $(SED) "/^domain/d" < $(<F).po > $@ ;
1086         $(RM) $(<F).po

1088 #
1089 # When using xgettext, we want messages to go to the default domain,
1090 # rather than the specified one.  This special version of the
1091 # COMPILE.cpp macro effectively prevents expansion of TEXT_DOMAIN,
1092 # causing xgettext to put all messages into the default domain.
```



```
1093 #
1094 CPPFORPO=$(COMPILE.cpp:\"$(TEXT_DOMAIN)\"=TEXT_DOMAIN)

1096 .c.i:
1097         $(CPPFORPO) $< > $@

1099 .h.i:
1100         $(CPPFORPO) $< > $@

1102 .y.i:
1103         $(YACC) -d $<
1104         $(CPPFORPO) y.tab.c  > $@
1105         $(RM) y.tab.c

1107 .l.i:
1108         $(LEX) $<
1109         $(CPPFORPO) lex.yy.c  > $@
1110         $(RM) lex.yy.c

1112 .c.po:
1113         $(CPPFORPO) $< > $<.i
1114         $(BUILD.po)

1116 .y.po:
1117         $(YACC) -d $<
1118         $(CPPFORPO) y.tab.c  > $<.i
1119         $(BUILD.po)
1120         $(RM) y.tab.c

1122 .l.po:
1123         $(LEX) $<
1124         $(CPPFORPO) lex.yy.c  > $<.i
1125         $(BUILD.po)
1126         $(RM) lex.yy.c

1128 #
1129 # Rules to perform stylistic checks
1130 #
1131 .SUFFIXES: .x .xml .check .xmlchk

1133 .h.check:
1134         $(DOT_H_CHECK)

1136 .x.check:
1137         $(DOT_X_CHECK)

1139 .xml.xmlchk:
1140         $(MANIFEST_CHECK)

1142 #
1143 # Include rules to render automated sccs get rules "safe".
1144 #
1145 include $(SRC)/Makefile.noget
```

```
**********************************************************
    2674 Sat Aug  3 15:26:11 2013
new/usr/src/lib/gss_mechs/mech_dh/backend/Makefile.com
3971 remove EXPORT_RELEASE_BUILD
**********************************************************
#
# 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 2008 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#

#
# This make file will build mech_dh.so.1. This shared object
# contains all the functionality needed to support Diffie-Hellman GSS-API
# mechanism.
#

LIBRARY= mech_dh.a
VERS = .1

MECH =  context.o context_establish.o cred.o crypto.o dhmech.o \
        MICwrap.o name.o oid.o seq.o token.o support.o validate.o

DERIVED_OBJS = xdr_token.o

CRYPTO = md5.o

OBJECTS= $(MECH) $(CRYPTO) $(DERIVED_OBJS)

# include library definitions
include ../../../../Makefile.lib

MAKEFILE_EXPORT = $(CLOSED)/lib/gss_mechs/mech_dh/backend/Makefile.export
$(EXPORT_RELEASE_BUILD)include $(MAKEFILE_EXPORT)


CPPFLAGS += -I../mech -I../crypto -I$(SRC)/uts/common/gssapi/include

CERRWARN +=     -_gcc=-Wno-parentheses
CERRWARN +=     -_gcc=-Wno-unused-variable
CERRWARN +=     -_gcc=-Wno-uninitialized

$(PICS) :=      CFLAGS += $(XFFLAG)
$(PICS) :=      CCFLAGS += $(XFFLAG)
$(PICS) :=      CFLAGS64 += $(XFFLAG)
$(PICS) :=      CCFLAGS64 += $(XFFLAG)
```



```
DYNFLAGS +=     $(ZIGNORE)

LIBS = $(DYNLIB)
LIBNAME = $(LIBRARY:%.a=%)

MAPFILES =      ../mapfile-vers

LDLIBS +=  -lgss -lnsl -lc

RPCGEN += -C
SED = sed

.KEEP_STATE:

CSRCS= $(MECH:%.o=../mech/%.c) $(CRYPTO:%.o=../crypto/%.c)
SRCS=   $(CSRCS)

ROOTLIBDIR = $(ROOT)/usr/lib/gss
ROOTLIBDIR64 = $(ROOT)/usr/lib/$(MACH64)/gss

#LINTFLAGS += -dirout=lint -errfmt=simple
#LINTFLAGS64 += -dirout=lint -errfmt=simple -errchk all
LINTOUT =       lint.out
LINTSRC =       $(LINTLIB:%.ln=%)
ROOTLINTDIR =   $(ROOTLIBDIR)
#ROOTLINT =     $(LINTSRC:%=$(ROOTLINTDIR)/%)

CLEANFILES += $(LINTOUT) $(LINTLIB)

lint: lintcheck

$(ROOTLIBDIR):
        $(INS.dir)

$(ROOTLIBDIR64):
        $(INS.dir)

$(OBJS): ../mech/dh_gssapi.h ../mech/token.h ../mech/oid.h


objs/%.o pics/%.o: ../crypto/%.c
        $(COMPILE.c)  -o $@ $<
        $(POST_PROCESS_O)

objs/%.o pics/%.o: ../mech/%.c
        $(COMPILE.c)  -o $@ $<
        $(POST_PROCESS_O)

objs/%.o pics/%.o: ../profile/%.c
        $(COMPILE.c)  -o $@ $<
        $(POST_PROCESS_O)

# include library targets
include ../../../../Makefile.targ
```

```
**********************************************************
    2480 Sat Aug  3 15:26:12 2013
new/usr/src/lib/gss_mechs/mech_dh/dh1024/Makefile.com
3971 remove EXPORT_RELEASE_BUILD
**********************************************************
#
# 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 2008 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#

#
# This make file will build dh1024.so.1. This shared object
# contains the functionality needed to initialize the  Diffie-Hellman GSS-API
# mechanism with 1024 bit key length. This library, in turn, loads the
# generic Diffie-Hellman GSS-API backend, dhmech.so
#

LIBRARY= dh1024-0.a
VERS = .1

DH1024= dh1024.o dh_common.o generic_key.o

OBJECTS= $(DH1024)

# include library definitions
include ../../../../Makefile.lib

MAKEFILE_EXPORT = $(CLOSED)/lib/gss_mechs/mech_dh/dh1024/Makefile.export
$(EXPORT_RELEASE_BUILD)include $(MAKEFILE_EXPORT)

CPPFLAGS += -I../../backend/mech -I../../backend/crypto
CPPFLAGS += -I$(SRC)/lib/libnsl/include
CPPFLAGS += -I$(SRC)/uts/common/gssapi/include

$(PICS) :=      CFLAGS += $(XFFLAG)
$(PICS) :=      CCFLAGS += $(XFFLAG)
$(PICS) :=      CFLAGS64 += $(XFFLAG)
$(PICS) :=      CCFLAGS64 += $(XFFLAG)

DYNFLAGS +=     $(ZIGNORE)

LIBS = $(DYNLIB)
LIBNAME = $(LIBRARY:%.a=%)

MAPFILES =      ../mapfile-vers
```



```
LDLIBS += -lnsl -lmp -lc

.KEEP_STATE:

SRCS=   ../dh1024.c ../../dh_common/dh_common.c ../../dh_common/generic_key.c

ROOTLIBDIR = $(ROOT)/usr/lib/gss
ROOTLIBDIR64 = $(ROOT)/usr/lib/$(MACH64)/gss

#LINTFLAGS += -errfmt=simple
#LINTFLAGS64 += -errfmt=simple
LINTOUT =       lint.out
LINTSRC =       $(LINTLIB:%.ln=%)
ROOTLINTDIR =   $(ROOTLIBDIR)
#ROOTLINT =     $(LINTSRC:%=$(ROOTLINTDIR)/%)

CLEANFILES += $(LINTOUT) $(LINTLIB)

lint: lintcheck

$(ROOTLIBDIR):
        $(INS.dir)

$(ROOTLIBDIR64):
        $(INS.dir)

# include library targets
include ../../../../Makefile.targ

objs/%.o pics/%.o: ../%.c
        $(COMPILE.c)  -o $@ $<
        $(POST_PROCESS_O)

objs/%.o pics/%.o: ../../dh_common/%.c
        $(COMPILE.c)  -o $@ $<
        $(POST_PROCESS_O)

objs/%.o pics/%.o: ../profile/%.c
        $(COMPILE.c)  -o $@ $<
        $(POST_PROCESS_O)
```

```
**********************************************************
    2472 Sat Aug  3 15:26:13 2013
new/usr/src/lib/gss_mechs/mech_dh/dh192/Makefile.com
3971 remove EXPORT_RELEASE_BUILD
**********************************************************
#
# 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 2008 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#

#
# This make file will build dh192.so.1. This shared object
# contains the functionality needed to initialize the  Diffie-Hellman GSS-API
# mechanism with 192 bit key length. This library, in turn, loads the
# generic Diffie-Hellman GSS-API backend, dhmech.so
#

LIBRARY= dh192-0.a
VERS = .1

DH192=  dh192.o dh_common.o generic_key.o

OBJECTS= $(DH192)

# include library definitions
include ../../../../Makefile.lib

MAKEFILE_EXPORT = $(CLOSED)/lib/gss_mechs/mech_dh/dh192/Makefile.export
$(EXPORT_RELEASE_BUILD)include $(MAKEFILE_EXPORT)

CPPFLAGS += -I../../backend/mech -I../../backend/crypto
CPPFLAGS += -I$(SRC)/lib/libnsl/include
CPPFLAGS += -I$(SRC)/uts/common/gssapi/include

$(PICS) :=      CFLAGS += $(XFFLAG)
$(PICS) :=      CCFLAGS += $(XFFLAG)
$(PICS) :=      CFLAGS64 += $(XFFLAG)
$(PICS) :=      CCFLAGS64 += $(XFFLAG)

DYNFLAGS +=     $(ZIGNORE)

LIBS = $(DYNLIB)
LIBNAME = $(LIBRARY:%.a=%)

MAPFILES =      ../mapfile-vers
```



```
LDLIBS += -lnsl -lmp -lc

.KEEP_STATE:

SRCS=   ../dh192.c ../../dh_common/dh_common.c ../../dh_common/generic_key.c

ROOTLIBDIR = $(ROOT)/usr/lib/gss
ROOTLIBDIR64 = $(ROOT)/usr/lib/$(MACH64)/gss

#LINTFLAGS += -errfmt=simple
#LINTFLAGS64 += -errfmt=simple
LINTOUT =       lint.out
LINTSRC =       $(LINTLIB:%.ln=%)
ROOTLINTDIR =   $(ROOTLIBDIR)
#ROOTLINT =     $(LINTSRC:%=$(ROOTLINTDIR)/%)

CLEANFILES += $(LINTOUT) $(LINTLIB)

lint: lintcheck

$(ROOTLIBDIR):
        $(INS.dir)

$(ROOTLIBDIR64):
        $(INS.dir)

# include library targets
include ../../../../Makefile.targ

objs/%.o pics/%.o: ../%.c
        $(COMPILE.c)  -o $@ $<
        $(POST_PROCESS_O)

objs/%.o pics/%.o: ../../dh_common/%.c
        $(COMPILE.c)  -o $@ $<
        $(POST_PROCESS_O)

objs/%.o pics/%.o: ../profile/%.c
        $(COMPILE.c)  -o $@ $<
        $(POST_PROCESS_O)
```

```
**********************************************************
    2473 Sat Aug  3 15:26:14 2013
new/usr/src/lib/gss_mechs/mech_dh/dh640/Makefile.com
3971 remove EXPORT_RELEASE_BUILD
**********************************************************
#
# 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 2008 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#

#
# This make file will build dh640.so.1. This shared object
# contains the functionality needed to initialize the  Diffie-Hellman GSS-API
# mechanism with 640 bit key length. This library, in turn, loads the
# generic Diffie-Hellman GSS-API backend, dhmech.so
#

LIBRARY= dh640-0.a
VERS = .1

DH640=  dh640.o dh_common.o generic_key.o

OBJECTS= $(DH640)

# include library definitions
include ../../../../Makefile.lib

MAKEFILE_EXPORT = $(CLOSED)/lib/gss_mechs/mech_dh/dh640/Makefile.export
$(EXPORT_RELEASE_BUILD)include $(MAKEFILE_EXPORT)

CPPFLAGS += -I../../backend/mech -I../../backend/crypto
CPPFLAGS += -I$(SRC)/lib/libnsl/include
CPPFLAGS += -I$(SRC)/uts/common/gssapi/include

$(PICS) :=      CFLAGS += $(XFFLAG)
$(PICS) :=      CCFLAGS += $(XFFLAG)
$(PICS) :=      CFLAGS64 += $(XFFLAG)
$(PICS) :=      CCFLAGS64 += $(XFFLAG)

DYNFLAGS +=     $(ZIGNORE)

LIBS = $(DYNLIB)
LIBNAME = $(LIBRARY:%.a=%)

MAPFILES =      ../mapfile-vers
```



```
LDLIBS += -lnsl -lmp -lc

.KEEP_STATE:

SRCS=   ../dh640.c ../../dh_common/dh_common.c ../../dh_common/generic_key.c

ROOTLIBDIR = $(ROOT)/usr/lib/gss
ROOTLIBDIR64 = $(ROOT)/usr/lib/$(MACH64)/gss

#LINTFLAGS += -errfmt=simple
#LINTFLAGS64 += -errfmt=simple
LINTOUT =       lint.out
LINTSRC =       $(LINTLIB:%.ln=%)
ROOTLINTDIR =   $(ROOTLIBDIR)
#ROOTLINT =     $(LINTSRC:%=$(ROOTLINTDIR)/%)

CLEANFILES += $(LINTOUT) $(LINTLIB)

lint: lintcheck

$(ROOTLIBDIR):
        $(INS.dir)

$(ROOTLIBDIR64):
        $(INS.dir)

# include library targets
include ../../../../Makefile.targ

objs/%.o pics/%.o: ../%.c
        $(COMPILE.c)  -o $@ $<
        $(POST_PROCESS_O)

objs/%.o pics/%.o: ../../dh_common/%.c
        $(COMPILE.c)  -o $@ $<
        $(POST_PROCESS_O)

objs/%.o pics/%.o: ../profile/%.c
        $(COMPILE.c)  -o $@ $<
        $(POST_PROCESS_O)
```

****
1455 Sat Aug 3 15:26:15 2013
new/usr/src/lib/gss_mechs/mech_dummy/Makefile.com
3971 remove EXPORT_RELEASE_BUILD
****

```
#
# 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 2006 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#

#
# The mech_dummy shared object contains all the functionality needed to
# support the Dummy GSS-API mechanism.
#

LIBRARY =       mech_dummy.a
VERS =          .1
OBJECTS =       dmech.o

include $(SRC)/lib/Makefile.lib

# There should be a mapfile here
MAPFILES =

CPPFLAGS +=     -I../../libgss -I$(SRC)/uts/common/gssapi/include \
                -I$(ROOT)/usr/include/gssapi

CERRWARN +=     -_gcc=-Wno-parentheses
CERRWARN +=     -_gcc=-Wno-uninitialized

MAKEFILE_EXPORT = $(CLOSED)/lib/gss_mechs/mech_dummy/Makefile.export
$(EXPORT_RELEASE_BUILD)include $(MAKEFILE_EXPORT)

SRCDIR =        ../mech
LIBS =          $(DYNLIB)
LDLIBS +=       -lgss -lc

.KEEP_STATE:

all: $(LIBS)

lint: lintcheck

include $(SRC)/lib/Makefile.targ
```

```
**********************************************************
    1702 Sat Aug  3 15:26:16 2013
new/usr/src/lib/gss_mechs/mech_spnego/Makefile.com
3971 remove EXPORT_RELEASE_BUILD
**********************************************************
```

```
   1 #
   2 # CDDL HEADER START
   3 #
   4 # The contents of this file are subject to the terms of the
   5 # Common Development and Distribution License (the "License").
   6 # You may not use this file except in compliance with the License.
   7 #
   8 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
   9 # or http://www.opensolaris.org/os/licensing.
  10 # See the License for the specific language governing permissions
  11 # and limitations under the License.
  12 #
  13 # When distributing Covered Code, include this CDDL HEADER in each
  14 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  15 # If applicable, add the following below this CDDL HEADER, with the
  16 # fields enclosed by brackets "[]" replaced with your own identifying
  17 # information: Portions Copyright [yyyy] [name of copyright owner]
  18 #
  19 # CDDL HEADER END
  20 #
  21 #
  22 # Copyright (c) 1999, 2010, Oracle and/or its affiliates. All rights reserved.
  23 #


  26 #
  27 # This make file will build mech_spnego.so.1. This shared object
  28 # contains all the functionality needed to support the SPNEGO GSS-API
  29 # mechanism.
  30 #

  32 LIBRARY =       mech_spnego.a
  33 VERS =          .1
  34 OBJECTS =       spnego_mech.o spnego_disp_status.o spnego_kerrs.o

  36 # include library definitions
  37 include ../../../Makefile.lib

  39 LIBS =          $(DYNLIB)
  40 ROOTLIBDIR =    $(ROOT)/usr/lib/gss
  41 ROOTLIBDIR64 =  $(ROOT)/usr/lib/$(MACH64)/gss
  42 SRCDIR =        ../mech

  44 MAPFILES =      ../mapfile-vers

  46 CPPFLAGS += -I$(SRC)/uts/common/gssapi/include $(DEBUG) -I$(SRC)/lib/gss_mechs/m

  48 CERRWARN +=     -_gcc=-Wno-unused-function
  49 CERRWARN +=     -_gcc=-Wno-type-limits

  51 MAKEFILE_EXPORT = $(CLOSED)/lib/gss_mechs/mech_spnego/Makefile.export
  52 $(EXPORT_RELEASE_BUILD)include $(MAKEFILE_EXPORT)

  51 .KEEP_STATE:

  53 all: $(LIBS)

  55 lint: lintcheck

  57 # include library targets
  58 include ../../../Makefile.targ
```

```
**********************************************************
    7767 Sat Aug  3 15:26:16 2013
new/usr/src/lib/libgss/Makefile.com
3971 remove EXPORT_RELEASE_BUILD
**********************************************************
#
# 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.
#

LIBRARY = libgss.a
VERS = .1

GSSOBJECTS      = g_acquire_cred.o \
          g_acquire_cred_with_pw.o \
          g_store_cred.o \
          g_rel_cred.o \
          g_init_sec_context.o \
          g_accept_sec_context.o \
          g_process_context.o \
          g_delete_sec_context.o \
          g_imp_sec_context.o \
          g_exp_sec_context.o \
          g_context_time.o \
          g_sign.o \
          g_verify.o \
          g_seal.o \
          g_unseal.o \
          g_dsp_status.o \
          g_compare_name.o \
          g_dsp_name.o \
          g_imp_name.o \
          g_rel_name.o \
          g_rel_buffer.o \
          g_rel_oid_set.o \
          g_oid_ops.o \
          g_inquire_cred.o \
          g_inquire_context.o \
          g_inquire_names.o \
          g_initialize.o \
          g_glue.o \
          gssd_pname_to_uid.o \
          oid_ops.o \
          g_canon_name.o \
          g_dup_name.o \
          g_export_name.o \
          g_utils.o \
```



```
          g_userok.o \
          g_buffer_set.o \
          g_inq_context_oid.o \


# defines the duplicate sources we share with gsscred
GSSCRED_DIR =   $(SRC)/cmd/gss/gsscred
GSSCREDOBJ =    gsscred_utils.o gsscred_file.o
# defines the duplicate sources we share with krb5 mech
KRB5DIR= $(SRC)/lib/gss_mechs/mech_krb5/mech
KRB5OBJ= rel_buffer.o util_buffer_set.o disp_com_err_status.o \
              util_buffer.o  util_errmap.o
# defines the duplicate sources we share with krb5 mech error table
KRB5ETDIR= $(SRC)/lib/gss_mechs/mech_krb5/et
KRB5ETOBJ= error_message.o adb_err.o adm_err.o asn1_err.o \
              chpass_util_strings.o \
              gssapi_err_krb5.o gssapi_err_generic.o \
              import_err.o \
              kadm_err.o kdb5_err.o kdc5_err.o kpasswd_strings.o krb5_err.o \
              kv5m_err.o prof_err.o pty_err.o ss_err.o
# defines the duplicate sources we share with kernel module
UTSGSSDIR =     $(SRC)/uts/common/gssapi
UTSGSSOBJ =     gen_oids.o

SRCS +=         $(GSSCREDOBJ:%.o=$(GSSCRED_DIR)/%.c) \
                $(KRB5OBJ:%.o=$(KRB5DIR)/%.c) \
                $(KRB5ETOBJ:%.o=$(KRB5ETDIR)/%.c) \
                $(UTSGSSOBJ:%.o=$(UTSGSSDIR)/%.c)
GSSLINTSRC =    $(GSSOBJECTS:%.o=$(SRCDIR)/%.c) \
                $(GSSCREDOBJ:%.o=$(GSSCRED_DIR)/%.c) \
                $(UTSGSSOBJ:%.o=$(UTSGSSDIR)/%.c)
OBJECTS =       $(GSSOBJECTS) $(GSSCREDOBJ) $(KRB5OBJ) $(UTSGSSOBJ) $(KRB5ETOBJ)

# include library definitions
include ../../Makefile.lib

LIBS =  $(DYNLIB) $(LINTLIB)

$(LINTLIB):=    SRCS = $(SRCDIR)/$(LINTSRC)
LDLIBS +=       -lc

CPPFLAGS +=     -I$(GSSCRED_DIR) -I$(SRC)/uts/common/gssapi/include \
                 -I$(SRC)/uts/common/gssapi/mechs/krb5/include \
                 -I$(SRC)/uts/common/gssapi/ \
                 -I$(SRC)/lib/gss_mechs/mech_krb5/include/ \
                -DHAVE_STDLIB_H

CERRWARN +=     -_gcc=-Wno-unused-function
CERRWARN +=     -_gcc=-Wno-uninitialized
CERRWARN +=     -_gcc=-Wno-parentheses
CERRWARN +=     -_gcc=-Wno-empty-body

$(EXPORT_RELEASE_BUILD)include $(CLOSED)/lib/libgss/Makefile.export

.KEEP_STATE:

all: $(LIBS)

lintcheck:=     SRCS= $(GSSLINTSRC)

lint:  lintcheck

$(GSSCREDOBJ:%.o=pics/%.o):
        $(COMPILE.c) -o $@ $(@:pics/%.o=$(GSSCRED_DIR)/%.c)
        $(POST_PROCESS_O)
```

```
126 # we need this in libgss so we don't have to link against mech_krb5
127 pics/rel_buffer.o: $(KRB5DIR)/rel_buffer.c
128         $(COMPILE.c) -o $@ $(KRB5DIR)/rel_buffer.c
129         $(POST_PROCESS_O)

131 # we need this in libgss so we don't have to link against mech_krb5
132 pics/util_buffer_set.o: $(KRB5DIR)/util_buffer_set.c
133         $(COMPILE.c) -o $@ $(KRB5DIR)/util_buffer_set.c
134         $(POST_PROCESS_O)

136 # we need this in libgss so we don't have to link against mech_krb5
137 pics/disp_com_err_status.o: $(KRB5DIR)/disp_com_err_status.c
138         $(COMPILE.c) -o $@ $(KRB5DIR)/disp_com_err_status.c
139         $(POST_PROCESS_O)

141 # we need this in libgss so we don't have to link against mech_krb5
142 pics/util_buffer.o: $(KRB5DIR)/util_buffer.c
143         $(COMPILE.c) -o $@ $(KRB5DIR)/util_buffer.c
144         $(POST_PROCESS_O)

146 # we need this in libgss so we don't have to link against mech_krb5
147 pics/util_errmap.o: $(KRB5DIR)/util_errmap.c
148         $(COMPILE.c) -o $@ $(KRB5DIR)/util_errmap.c
149         $(POST_PROCESS_O)

151 # we need this in libgss so we don't have to link against mech_krb5
152 pics/error_message.o: $(KRB5ETDIR)/error_message.c
153         $(COMPILE.c) -o $@ $(KRB5ETDIR)/error_message.c
154         $(POST_PROCESS_O)

156 # we need this in libgss so we don't have to link against mech_krb5
157 pics/adb_err.o: $(KRB5ETDIR)/adb_err.c
158         $(COMPILE.c) -o $@ $(KRB5ETDIR)/adb_err.c
159         $(POST_PROCESS_O)

161 pics/adm_err.o: $(KRB5ETDIR)/adm_err.c
162         $(COMPILE.c) -o $@ $(KRB5ETDIR)/adm_err.c
163         $(POST_PROCESS_O)

165 # we need this in libgss so we don't have to link against mech_krb5
166 pics/asn1_err.o: $(KRB5ETDIR)/asn1_err.c
167         $(COMPILE.c) -o $@ $(KRB5ETDIR)/asn1_err.c
168         $(POST_PROCESS_O)

170 # we need this in libgss so we don't have to link against mech_krb5
171 pics/chpass_util_strings.o: $(KRB5ETDIR)/chpass_util_strings.c
172         $(COMPILE.c) -o $@ $(KRB5ETDIR)/chpass_util_strings.c
173         $(POST_PROCESS_O)

175 # we need this in libgss so we don't have to link against mech_krb5
176 pics/gssapi_err_generic.o: $(KRB5ETDIR)/gssapi_err_generic.c
177         $(COMPILE.c) -o $@ $(KRB5ETDIR)/gssapi_err_generic.c
178         $(POST_PROCESS_O)

180 # we need this in libgss so we don't have to link against mech_krb5
181 pics/gssapi_err_krb5.o: $(KRB5ETDIR)/gssapi_err_krb5.c
182         $(COMPILE.c) -o $@ $(KRB5ETDIR)/gssapi_err_krb5.c
183         $(POST_PROCESS_O)


186 # we need this in libgss so we don't have to link against mech_krb5
187 pics/import_err.o: $(KRB5ETDIR)/import_err.c
188         $(COMPILE.c) -o $@ $(KRB5ETDIR)/import_err.c
189         $(POST_PROCESS_O)

191 # we need this in libgss so we don't have to link against mech_krb5
```



```
192 pics/kadm_err.o: $(KRB5ETDIR)/kadm_err.c
193         $(COMPILE.c) -o $@ $(KRB5ETDIR)/kadm_err.c
194         $(POST_PROCESS_O)

196 # we need this in libgss so we don't have to link against mech_krb5
197 pics/kdb5_err.o: $(KRB5ETDIR)/kdb5_err.c
198         $(COMPILE.c) -o $@ $(KRB5ETDIR)/kdb5_err.c
199         $(POST_PROCESS_O)

201 # we need this in libgss so we don't have to link against mech_krb5
202 pics/kdc5_err.o: $(KRB5ETDIR)/kdc5_err.c
203         $(COMPILE.c) -o $@ $(KRB5ETDIR)/kdc5_err.c
204         $(POST_PROCESS_O)

206 # we need this in libgss so we don't have to link against mech_krb5
207 pics/kpasswd_strings.o: $(KRB5ETDIR)/kpasswd_strings.c
208         $(COMPILE.c) -o $@ $(KRB5ETDIR)/kpasswd_strings.c
209         $(POST_PROCESS_O)

211 # we need this in libgss so we don't have to link against mech_krb5
212 pics/krb5_err.o: $(KRB5ETDIR)/krb5_err.c
213         $(COMPILE.c) -o $@ $(KRB5ETDIR)/krb5_err.c
214         $(POST_PROCESS_O)

216 # we need this in libgss so we don't have to link against mech_krb5
217 pics/kv5m_err.o: $(KRB5ETDIR)/kv5m_err.c
218         $(COMPILE.c) -o $@ $(KRB5ETDIR)/kv5m_err.c
219         $(POST_PROCESS_O)

221 # we need this in libgss so we don't have to link against mech_krb5
222 pics/prof_err.o: $(KRB5ETDIR)/prof_err.c
223         $(COMPILE.c) -o $@ $(KRB5ETDIR)/prof_err.c
224         $(POST_PROCESS_O)

226 # we need this in libgss so we don't have to link against mech_krb5
227 pics/pty_err.o: $(KRB5ETDIR)/pty_err.c
228         $(COMPILE.c) -o $@ $(KRB5ETDIR)/pty_err.c
229         $(POST_PROCESS_O)

231 # we need this in libgss so we don't have to link against mech_krb5
232 pics/ss_err.o: $(KRB5ETDIR)/ss_err.c
233         $(COMPILE.c) -o $@ $(KRB5ETDIR)/ss_err.c
234         $(POST_PROCESS_O)

236 # gen_oids.c is kept in the kernel since the OIDs declared in them are
237 # used by rpcsec module
238 pics/gen_oids.o: $(SRC)/uts/common/gssapi/gen_oids.c
239         $(COMPILE.c) -o $@ $(SRC)/uts/common/gssapi/gen_oids.c
240         $(POST_PROCESS_O)

242 # include library targets
243 include ../../Makefile.targ
```

```
**********************************************************
   85765 Sat Aug  3 15:26:17 2013
new/usr/src/tools/scripts/nightly.sh
3971 remove EXPORT_RELEASE_BUILD
**********************************************************
______unchanged_portion_omitted_

2397 #
2398 #       Decide whether to bringover to the codemgr workspace
2399 #
2400 if [ "$n_FLAG" = "n" ]; then
2401         PARENT_SCM_TYPE=$(parent_wstype)

2403         if [[ $SCM_TYPE != none && $SCM_TYPE != $PARENT_SCM_TYPE ]]; then
2404                 echo "cannot bringover from $PARENT_SCM_TYPE to $SCM_TYPE, " \
2405                     "quitting at 'date'." | tee -a $mail_msg_file >> $LOGFILE
2406                 exit 1
2407         fi

2409         run_hook PRE_BRINGOVER

2411         echo "\n==== bringover to $CODEMGR_WS at 'date' ====\n" >> $LOGFILE
2412         echo "\n==== BRINGOVER LOG ====\n" >> $mail_msg_file

2414         eval "bringover_${PARENT_SCM_TYPE}" 2>&1 |
2415                 tee -a $mail_msg_file >> $LOGFILE

2417         if [ -f $TMPDIR/bringover_failed ]; then
2418                 rm -f $TMPDIR/bringover_failed
2419                 build_ok=n
2420                 echo "trouble with bringover, quitting at 'date'." |
2421                         tee -a $mail_msg_file >> $LOGFILE
2422                 exit 1
2423         fi

2425         #
2426         # It's possible that we used the bringover above to create
2427         # $CODEMGR_WS.  If so, then SCM_TYPE was previously "none,"
2428         # but should now be the same as $BRINGOVER_WS.
2429         #
2430         [[ $SCM_TYPE = none ]] && SCM_TYPE=$PARENT_SCM_TYPE

2432         run_hook POST_BRINGOVER

2434         #
2435         # Possible transition from pre-split workspace to split
2436         # workspace.  See if the bringover changed anything.
2437         #
2438         CLOSED_IS_PRESENT="$orig_closed_is_present"
2439         check_closed_tree

2441 else
2442         echo "\n==== No bringover to $CODEMGR_WS ====\n" >> $LOGFILE
2443 fi

2445 if [[ "$O_FLAG" = y && "$CLOSED_IS_PRESENT" != "yes" ]]; then
2446         build_ok=n
2447         echo "OpenSolaris binary deliverables need usr/closed." \
2448             | tee -a "$mail_msg_file" >> $LOGFILE
2449         exit 1
2450 fi

2452 # Safeguards
2453 [[ -v CODEMGR_WS ]] || fatal_error "Error: Variable CODEMGR_WS not set."
2454 [[ -d "${CODEMGR_WS}" ]] || fatal_error "Error: ${CODEMGR_WS} is not a directory
2455 [[ -f "${CODEMGR_WS}/usr/src/Makefile" ]] || fatal_error "Error: ${CODEMGR_WS}/u
```



```
2457 echo "\n==== Build environment ====\n" | tee -a $build_environ_file >> $LOGFILE

2459 # System
2460 whence uname | tee -a $build_environ_file >> $LOGFILE
2461 uname -a 2>&1 | tee -a $build_environ_file >> $LOGFILE
2462 echo | tee -a $build_environ_file >> $LOGFILE

2464 # make
2465 whence $MAKE | tee -a $build_environ_file >> $LOGFILE
2466 $MAKE -v | tee -a $build_environ_file >> $LOGFILE
2467 echo "number of concurrent jobs = $DMAKE_MAX_JOBS" |
2468     tee -a $build_environ_file >> $LOGFILE

2470 #
2471 # Report the compiler versions.
2472 #

2474 if [[ ! -f $SRC/Makefile ]]; then
2475         build_ok=n
2476         echo "\nUnable to find \"Makefile\" in $SRC." | \
2477             tee -a $build_environ_file >> $LOGFILE
2478         exit 1
2479 fi

2481 ( cd $SRC
2482   for target in cc-version cc64-version java-version; do
2483         echo
2484         #
2485         # Put statefile somewhere we know we can write to rather than trip
2486         # over a read-only $srcroot.
2487         #
2488         rm -f $TMPDIR/make-state
2489         export SRC
2490         if $MAKE -K $TMPDIR/make-state -e $target 2>/dev/null; then
2491                 continue
2492         fi
2493         touch $TMPDIR/nocompiler
2494   done
2495   echo
2496 ) | tee -a $build_environ_file >> $LOGFILE

2498 if [ -f $TMPDIR/nocompiler ]; then
2499         rm -f $TMPDIR/nocompiler
2500         build_ok=n
2501         echo "Aborting due to missing compiler." |
2502                 tee -a $build_environ_file >> $LOGFILE
2503         exit 1
2504 fi

2506 # as
2507 whence as | tee -a $build_environ_file >> $LOGFILE
2508 as -V 2>&1 | head -1 | tee -a $build_environ_file >> $LOGFILE
2509 echo | tee -a $build_environ_file >> $LOGFILE

2511 # Check that we're running a capable link-editor
2512 whence ld | tee -a $build_environ_file >> $LOGFILE
2513 LDVER='ld -V 2>&1'
2514 echo $LDVER | tee -a $build_environ_file >> $LOGFILE
2515 LDVER='echo $LDVER | sed -e "s/.*-1\.\([0-9]*\).*/\1/"'
2516 if [ 'expr $LDVER \< 422' -eq 1 ]; then
2517         echo "The link-editor needs to be at version 422 or higher to build" | \
2518             tee -a $build_environ_file >> $LOGFILE
2519         echo "the latest stuff.  Hope your build works." | \
2520             tee -a $build_environ_file >> $LOGFILE
2521 fi
```

```
2523 #
2524 # Build and use the workspace's tools if requested
2525 #
2526 if [[ "$t_FLAG" = "y" || "$O_FLAG" = y ]]; then
2527         set_non_debug_build_flags

2529         build_tools ${TOOLS_PROTO}
2530         if [[ $? != 0  && "$t_FLAG" = y ]]; then
2531                 use_tools $TOOLS_PROTO
2532         fi
2533 fi

2535 #
2536 # copy ihv proto area in addition to the build itself
2537 #
2538 if [ "$X_FLAG" = "y" ]; then
2539         copy_ihv_proto
2540 fi

2542 if [ "$i_FLAG" = "y" -a "$SH_FLAG" = "y" ]; then
2543         echo "\n==== NOT Building base OS-Net source ====\n" | \
2544             tee -a $LOGFILE >> $mail_msg_file
2545 else
2546         # timestamp the start of the normal build; the findunref tool uses it.
2547         touch $SRC/.build.tstamp

2549         normal_build
2550 fi

2552 #
2553 # Generate the THIRDPARTYLICENSE files if needed.  This is done after
2554 # the build, so that dynamically-created license files are there.
2555 # It's done before findunref to help identify license files that need
2556 # to be added to tools/opensolaris/license-list.
2557 #
2558 if [ "$O_FLAG" = y -a "$build_ok" = y ]; then
2559         echo "\n==== Generating THIRDPARTYLICENSE files ====\n" |
2560             tee -a "$mail_msg_file" >> "$LOGFILE"

2562         if [ -d $ROOT/licenses/usr ]; then
2563                 ( cd $ROOT/licenses ; \
2564                     mktpl $SRC/pkg/license-list ) >> "$LOGFILE" 2>&1
2565                 if (( $? != 0 )) ; then
2566                         echo "Couldn't create THIRDPARTYLICENSE files" |
2567                             tee -a "$mail_msg_file" >> "$LOGFILE"
2568                 fi
2569         else
2570                 echo "No licenses found under $ROOT/licenses" |
2571                     tee -a "$mail_msg_file" >> "$LOGFILE"
2572         fi
2573 fi

2575 ORIG_SRC=$SRC
2576 BINARCHIVE=${CODEMGR_WS}/bin-${MACH}.cpio.Z

2578 if [ "$SE_FLAG" = "y" -o "$SD_FLAG" = "y" -o "$SH_FLAG" = "y" ]; then
2579         save_binaries
2580 fi


2583 # EXPORT_SRC comes after CRYPT_SRC since a domestic build will need
2584 # $SRC pointing to the export_source usr/src.

2586 if [ "$SE_FLAG" = "y" -o "$SD_FLAG" = "y" -o "$SH_FLAG" = "y" ]; then
2587         if [ "$SD_FLAG" = "y" -a $build_ok = y ]; then
```



```
2588             set_up_source_build ${CODEMGR_WS} ${CRYPT_SRC} CRYPT_SRC
2589         fi

2591         if [ $build_ok = y ]; then
2592             set_up_source_build ${CODEMGR_WS} ${EXPORT_SRC} EXPORT_SRC
2593         fi
2594 fi

2596 if [ "$SD_FLAG" = "y" -a $build_ok = y ]; then
2597         # drop the crypt files in place.
2598         cd ${EXPORT_SRC}
2599         echo "\nextracting crypt_files.cpio.Z onto export_source.\n" \
2600             >> ${LOGFILE}
2601         zcat ${CODEMGR_WS}/crypt_files.cpio.Z | \
2602             cpio -idmucvB 2>/dev/null >> ${LOGFILE}
2603         if [ "$?" = "0" ]; then
2604                 echo "\n==== DOMESTIC extraction succeeded ====\n" \
2605                     >> $mail_msg_file
2606         else
2607                 echo "\n==== DOMESTIC extraction failed ====\n" \
2608                     >> $mail_msg_file
2609         fi

2611 fi

2613 if [ "$SO_FLAG" = "y" -a "$build_ok" = y ]; then
2614         #
2615         # Copy the open sources into their own tree.
2616         # If copy_source fails, it will have already generated an
2617         # error message and set build_ok=n, so we don't need to worry
2618         # about that here.
2619         #
2620         copy_source $CODEMGR_WS $OPEN_SRCDIR OPEN_SOURCE usr/src
2621 fi

2623 if [ "$SO_FLAG" = "y" -a "$build_ok" = y ]; then
2624         SRC=$OPEN_SRCDIR/usr/src
2625         export CLOSED_IS_PRESENT=no
2626 fi

2628 if is_source_build && [ $build_ok = y ] ; then
2629         # remove proto area(s) here, since we don't clobber
2630         rm -rf 'allprotos'
2631         if [ "$t_FLAG" = "y" ]; then
2632                 set_non_debug_build_flags
2633                 ORIG_TOOLS=$TOOLS
2634                 #
2635                 # SRC was set earlier to point to the source build
2636                 # source tree (e.g., $EXPORT_SRC).
2637                 #
2638                 TOOLS=${SRC}/tools
2639                 TOOLS_PROTO=${TOOLS}/${TOOLS_PROTO_REL}; export TOOLS_PROTO
2640                 build_tools ${TOOLS_PROTO}
2641                 if [[ $? != 0 ]]; then
2642                         use_tools ${TOOLS_PROTO}
2643                 fi
2644         fi

2646         export EXPORT_RELEASE_BUILD ; EXPORT_RELEASE_BUILD=#
2646         normal_build
2647 fi

2649 #
2650 # There are several checks that need to look at the proto area, but
2651 # they only need to look at one, and they don't care whether it's
2652 # DEBUG or non-DEBUG.
```

```
#
if [[ "$MULTI_PROTO" = yes && "$D_FLAG" = n ]]; then
        checkroot=$ROOT-nd
else
        checkroot=$ROOT
fi

if [ "$build_ok" = "y" ]; then
        echo "\n==== Creating protolist system file at ‘date‘ ====" \
                >> $LOGFILE
        protolist $checkroot > $ATLOG/proto_list_${MACH}
        echo "==== protolist system file created at ‘date‘ ====\n" \
                >> $LOGFILE

        if [ "$N_FLAG" != "y" ]; then

                E1=
                f1=
                if [ -d "$SRC/pkgdefs" ]; then
                        f1="$SRC/pkgdefs/etc/exception_list_$MACH"
                        if [ "$X_FLAG" = "y" ]; then
                                f1="$f1 $IA32_IHV_WS/usr/src/pkgdefs/etc/excepti
                        fi
                fi

                for f in $f1; do
                        if [ -f "$f" ]; then
                                E1="$E1 -e $f"
                        fi
                done

                E2=
                f2=
                if [ -d "$SRC/pkg" ]; then
                        f2="$f2 exceptions/packaging"
                fi

                for f in $f2; do
                        if [ -f "$f" ]; then
                                E2="$E2 -e $f"
                        fi
                done

                if [ -f "$REF_PROTO_LIST" ]; then
                        #
                        # For builds that copy the IHV proto area (-X), add the
                        # IHV proto list to the reference list if the reference
                        # was built without -X.
                        #
                        # For builds that don’t copy the IHV proto area, add the
                        # IHV proto list to the build’s proto list if the
                        # reference was built with -X.
                        #
                        # Use the presence of the first file entry of the cached
                        # IHV proto list in the reference list to determine
                        # whether it was built with -X or not.
                        #
                        IHV_REF_PROTO_LIST=$SRC/pkg/proto_list_ihv_$MACH
                        grepfor=$(nawk ’$1 == "f" { print $2; exit }’ \
                                $IHV_REF_PROTO_LIST 2> /dev/null)
                        if [ $? = 0 -a -n "$grepfor" ]; then
                                if [ "$X_FLAG" = "y" ]; then
                                        grep -w "$grepfor" \
                                                $REF_PROTO_LIST > /dev/null
                                        if [ ! "$?" = "0" ]; then
                                                REF_IHV_PROTO="-d $IHV_REF_PROTO
```



```
                                        fi
                                else
                                        grep -w "$grepfor" \
                                                $REF_PROTO_LIST > /dev/null
                                        if [ "$?" = "0" ]; then
                                                IHV_PROTO_LIST="$IHV_REF_PROTO_L
                                        fi
                                fi
                        fi
                fi
        fi

        if [ "$N_FLAG" != "y" -a -f $SRC/pkgdefs/Makefile ]; then
                echo "\n==== Impact on SVr4 packages ====\n" >> $mail_msg_file
                #
                # Compare the build’s proto list with current package
                # definitions to audit the quality of package
                # definitions and makefile install targets. Use the
                # current exception list.
                #
                PKGDEFS_LIST=""
                for d in $abssrcdirs; do
                        if [ -d $d/pkgdefs ]; then
                                PKGDEFS_LIST="$PKGDEFS_LIST -d $d/pkgdefs"
                        fi
                done
                if [ "$X_FLAG" = "y" -a \
                    -d $IA32_IHV_WS/usr/src/pkgdefs ]; then
                        PKGDEFS_LIST="$PKGDEFS_LIST -d $IA32_IHV_WS/usr/src/pkgd
                fi
                $PROTOCMPTERSE \
                    "Files missing from the proto area:" \
                    "Files missing from packages:" \
                    "Inconsistencies between pkgdefs and proto area:" \
                    ${E1} \
                    ${PKGDEFS_LIST} \
                    $ATLOG/proto_list_${MACH} \
                    >> $mail_msg_file
        fi

        if [ "$N_FLAG" != "y" -a -d $SRC/pkg ]; then
                echo "\n==== Validating manifests against proto area ====\n" \
                    >> $mail_msg_file
                ( cd $SRC/pkg ; $MAKE -e protocmp ROOT="$checkroot" ) \
                    >> $mail_msg_file

        fi

        if [ "$N_FLAG" != "y" -a -f "$REF_PROTO_LIST" ]; then
                echo "\n==== Impact on proto area ====\n" >> $mail_msg_file
                if [ -n "$E2" ]; then
                        ELIST=$E2
                else
                        ELIST=$E1
                fi
                $PROTOCMPTERSE \
                        "Files in yesterday’s proto area, but not today’s:" \
                        "Files in today’s proto area, but not yesterday’s:" \
                        "Files that changed between yesterday and today:" \
                        ${ELIST} \
                        -d $REF_PROTO_LIST \
                        $REF_IHV_PROTO \
                        $ATLOG/proto_list_${MACH} \
                        $IHV_PROTO_LIST \
                        >> $mail_msg_file
        fi
```

```
2785 fi

2787 if [ "$u_FLAG" = "y"  -a "$build_ok" = "y" ]; then
2788         staffer cp $ATLOG/proto_list_${MACH} \
2789                 $PARENT_WS/usr/src/proto_list_${MACH}
2790 fi

2792 # Update parent proto area if necessary. This is done now
2793 # so that the proto area has either DEBUG or non-DEBUG kernels.
2794 # Note that this clears out the lock file, so we can dispense with
2795 # the variable now.
2796 if [ "$U_FLAG" = "y" -a "$build_ok" = "y" ]; then
2797         echo "\n==== Copying proto area to $NIGHTLY_PARENT_ROOT ====\n" | \
2798             tee -a $LOGFILE >> $mail_msg_file
2799         rm -rf $NIGHTLY_PARENT_ROOT/*
2800         unset Ulockfile
2801         mkdir -p $NIGHTLY_PARENT_ROOT
2802         if [[ "$MULTI_PROTO" = no || "$D_FLAG" = y ]]; then
2803                 ( cd $ROOT; tar cf - . |
2804                     ( cd $NIGHTLY_PARENT_ROOT;  umask 0; tar xpf - ) ) 2>&1 |
2805                     tee -a $mail_msg_file >> $LOGFILE
2806         fi
2807         if [[ "$MULTI_PROTO" = yes && "$F_FLAG" = n ]]; then
2808                 rm -rf $NIGHTLY_PARENT_ROOT-nd/*
2809                 mkdir -p $NIGHTLY_PARENT_ROOT-nd
2810                 cd $ROOT-nd
2811                 ( tar cf - . |
2812                     ( cd $NIGHTLY_PARENT_ROOT-nd; umask 0; tar xpf - ) ) 2>&1 |
2813                     tee -a $mail_msg_file >> $LOGFILE
2814         fi
2815         if [ -n "${NIGHTLY_PARENT_TOOLS_ROOT}" ]; then
2816                 echo "\n==== Copying tools proto area to $NIGHTLY_PARENT_TOOLS_R
2817                     tee -a $LOGFILE >> $mail_msg_file
2818                 rm -rf $NIGHTLY_PARENT_TOOLS_ROOT/*
2819                 mkdir -p $NIGHTLY_PARENT_TOOLS_ROOT
2820                 if [[ "$MULTI_PROTO" = no || "$D_FLAG" = y ]]; then
2821                         ( cd $TOOLS_PROTO; tar cf - . |
2822                             ( cd $NIGHTLY_PARENT_TOOLS_ROOT;
2823                             umask 0; tar xpf - ) ) 2>&1 |
2824                             tee -a $mail_msg_file >> $LOGFILE
2825                 fi
2826         fi
2827 fi

2829 #
2830 # ELF verification: ABI (-A) and runtime (-r) checks
2831 #
2832 if [[ ($build_ok = y) && ( ($A_FLAG = y) || ($r_FLAG = y) ) ]]; then
2833         # Directory ELF-data.$MACH holds the files produced by these tests.
2834         elf_ddir=$SRC/ELF-data.$MACH

2836         # If there is a previous ELF-data backup directory, remove it. Then,
2837         # rotate current ELF-data directory into its place and create a new
2838         # empty directory
2839         rm -rf $elf_ddir.ref
2840         if [[ -d $elf_ddir ]]; then
2841                 mv $elf_ddir $elf_ddir.ref
2842         fi
2843         mkdir -p $elf_ddir

2845         # Call find_elf to produce a list of the ELF objects in the proto area.
2846         # This list is passed to check_rtime and interface_check, preventing
2847         # them from separately calling find_elf to do the same work twice.
2848         find_elf -fr $checkroot > $elf_ddir/object_list

2850         if [[ $A_FLAG = y ]]; then
```



```
2851                 echo "\n==== Check versioning and ABI information ====\n"  | \
2852                     tee -a $LOGFILE >> $mail_msg_file

2854                 # Produce interface description for the proto. Report errors.
2855                 interface_check -o -w $elf_ddir -f object_list \
2856                         -i interface -E interface.err
2857                 if [[ -s $elf_ddir/interface.err ]]; then
2858                         tee -a $LOGFILE < $elf_ddir/interface.err \
2859                                 >> $mail_msg_file
2860                 fi

2862                 # If ELF_DATA_BASELINE_DIR is defined, compare the new interface
2863                 # description file to that from the baseline gate. Issue a
2864                 # warning if the baseline is not present, and keep going.
2865                 if [[ "$ELF_DATA_BASELINE_DIR" != '' ]]; then
2866                         base_ifile="$ELF_DATA_BASELINE_DIR/interface"

2868                         echo "\n==== Compare versioning and ABI information" \
2869                             "to baseline ====\n"  | \
2870                             tee -a $LOGFILE >> $mail_msg_file
2871                         echo "Baseline:  $base_ifile\n" >> $LOGFILE

2873                         if [[ -f $base_ifile ]]; then
2874                                 interface_cmp -d -o $base_ifile \
2875                                     $elf_ddir/interface > $elf_ddir/interface.cm
2876                                 if [[ -s $elf_ddir/interface.cmp ]]; then
2877                                         echo | tee -a $LOGFILE >> $mail_msg_file
2878                                         tee -a $LOGFILE < \
2879                                             $elf_ddir/interface.cmp \
2880                                             >> $mail_msg_file
2881                                 fi
2882                         else
2883                                 echo "baseline not available. comparison" \
2884                                     "skipped" | \
2885                                     tee -a $LOGFILE >> $mail_msg_file
2886                         fi

2888                 fi
2889         fi

2891         if [[ $r_FLAG = y ]]; then
2892                 echo "\n==== Check ELF runtime attributes ====\n" | \
2893                     tee -a $LOGFILE >> $mail_msg_file

2895                 # If we're doing a DEBUG build the proto area will be left
2896                 # with debuggable objects, thus don't assert -s.
2897                 if [[ $D_FLAG = y ]]; then
2898                         rtime_sflag=""
2899                 else
2900                         rtime_sflag="-s"
2901                 fi
2902                 check_rtime -i -m -v $rtime_sflag -o -w $elf_ddir \
2903                         -D object_list  -f object_list -E runtime.err \
2904                         -I runtime.attr.raw

2906                 # check_rtime -I output needs to be sorted in order to
2907                 # compare it to that from previous builds.
2908                 sort $elf_ddir/runtime.attr.raw > $elf_ddir/runtime.attr
2909                 rm $elf_ddir/runtime.attr.raw

2911                 # Report errors
2912                 if [[ -s $elf_ddir/runtime.err ]]; then
2913                         tee -a $LOGFILE < $elf_ddir/runtime.err \
2914                                 >> $mail_msg_file
2915                 fi
```

```
2917                 # If there is an ELF-data directory from a previous build,
2918                 # then diff the attr files. These files contain information
2919                 # about dependencies, versioning, and runpaths. There is some
2920                 # overlap with the ABI checking done above, but this also
2921                 # flushes out non-ABI interface differences along with the
2922                 # other information.
2923                 echo "\n==== Diff ELF runtime attributes" \
2924                     "(since last build) ====\n" | \
2925                     tee -a $LOGFILE >> $mail_msg_file >> $mail_msg_file

2927                 if [[ -f $elf_ddir.ref/runtime.attr ]]; then
2928                         diff $elf_ddir.ref/runtime.attr \
2929                                 $elf_ddir/runtime.attr \
2930                                 >> $mail_msg_file
2931                 fi
2932         fi

2934         # If -u set, copy contents of ELF-data.$MACH to the parent workspace.
2935         if [[ "$u_FLAG" = "y" ]]; then
2936                 p_elf_ddir=$PARENT_WS/usr/src/ELF-data.$MACH

2938                 # If parent lacks the ELF-data.$MACH directory, create it
2939                 if [[ ! -d $p_elf_ddir ]]; then
2940                         staffer mkdir -p $p_elf_ddir
2941                 fi

2943                 # These files are used asynchronously by other builds for ABI
2944                 # verification, as above for the -A option. As such, we require
2945                 # the file replacement to be atomic. Copy the data to a temp
2946                 # file in the same filesystem and then rename into place.
2947                 (
2948                         cd $elf_ddir
2949                         for elf_dfile in *; do
2950                                 staffer cp $elf_dfile \
2951                                     ${p_elf_ddir}/${elf_dfile}.new
2952                                 staffer mv -f ${p_elf_ddir}/${elf_dfile}.new \
2953                                     ${p_elf_ddir}/${elf_dfile}
2954                         done
2955                 )
2956         fi
2957 fi

2959 # DEBUG lint of kernel begins

2961 if [ "$i_CMD_LINE_FLAG" = "n" -a "$l_FLAG" = "y" ]; then
2962         if [ "$LINTDIRS" = "" ]; then
2963                 # LINTDIRS="$SRC/uts y $SRC/stand y $SRC/psm y"
2964                 LINTDIRS="$SRC y"
2965         fi
2966         set $LINTDIRS
2967         while [ $# -gt 0 ]; do
2968                 dolint $1 $2; shift; shift
2969         done
2970 else
2971         echo "\n==== No '$MAKE lint' ====\n" >> $LOGFILE
2972 fi

2974 # "make check" begins

2976 if [ "$i_CMD_LINE_FLAG" = "n" -a "$C_FLAG" = "y" ]; then
2977         # remove old check.out
2978         rm -f $SRC/check.out

2980         rm -f $SRC/check-${MACH}.out
2981         cd $SRC
2982         $MAKE -ek check ROOT="$checkroot" 2>&1 | tee -a $SRC/check-${MACH}.out \
```



```
2983             >> $LOGFILE
2984         echo "\n==== cstyle/hdrchk errors ====\n" >> $mail_msg_file

2986         grep ":" $SRC/check-${MACH}.out |
2987                 egrep -v "Ignoring unknown host" | \
2988                 sort | uniq >> $mail_msg_file
2989 else
2990         echo "\n==== No '$MAKE check' ====\n" >> $LOGFILE
2991 fi

2993 echo "\n==== Find core files ====\n" | \
2994     tee -a $LOGFILE >> $mail_msg_file

2996 find $abssrcdirs -name core -a -type f -exec file {} \; | \
2997         tee -a $LOGFILE >> $mail_msg_file

2999 if [ "$f_FLAG" = "y" -a "$build_ok" = "y" ]; then
3000         echo "\n==== Diff unreferenced files (since last build) ====\n" \
3001             | tee -a $LOGFILE >>$mail_msg_file
3002         rm -f $SRC/unref-${MACH}.ref
3003         if [ -f $SRC/unref-${MACH}.out ]; then
3004                 mv $SRC/unref-${MACH}.out $SRC/unref-${MACH}.ref
3005         fi

3007         findunref -S $SCM_TYPE -t $SRC/.build.tstamp -s usr $CODEMGR_WS \
3008             ${TOOLS}/findunref/exception_list 2>> $mail_msg_file | \
3009             sort > $SRC/unref-${MACH}.out

3011         if [ ! -f $SRC/unref-${MACH}.ref ]; then
3012                 cp $SRC/unref-${MACH}.out $SRC/unref-${MACH}.ref
3013         fi

3015         diff $SRC/unref-${MACH}.ref $SRC/unref-${MACH}.out >>$mail_msg_file
3016 fi

3018 #
3019 # Generate the OpenSolaris deliverables if requested.  Some of these
3020 # steps need to come after findunref and are commented below.
3021 #

3023 # If we are doing an OpenSolaris _source_ build (-S O) then we do
3024 # not have usr/closed available to us to generate closedbins from,
3025 # so skip this part.
3026 if [ "$SO_FLAG" = n -a "$O_FLAG" = y -a "$build_ok" = y ]; then
3027         echo "\n==== Generating OpenSolaris tarballs ====\n" | \
3028             tee -a $mail_msg_file >> $LOGFILE

3030         cd $CODEMGR_WS

3032         #
3033         # This step grovels through the package manifests, so it
3034         # must come after findunref.
3035         #
3036         # We assume no DEBUG vs non-DEBUG package content variation
3037         # here; if that changes, then the "make all" in $SRC/pkg will
3038         # need to be moved into the conditionals and repeated for each
3039         # different build.
3040         #
3041         echo "Generating closed binaries tarball(s)..." >> $LOGFILE
3042         closed_basename=on-closed-bins
3043         if [ "$D_FLAG" = y ]; then
3044                 bindrop "$closed_basename" >>"$LOGFILE" 2>&1
3045                 if (( $? != 0 )) ; then
3046                         echo "Couldn't create DEBUG closed binaries." |
3047                             tee -a $mail_msg_file >> $LOGFILE
3048                         build_ok=n
```

```
3049                 fi
3050         fi
3051         if [ "$F_FLAG" = n ]; then
3052                 bindrop -n "$closed_basename-nd" >>"$LOGFILE" 2>&1
3053                 if (( $? != 0 )) ; then
3054                         echo "Couldn't create non-DEBUG closed binaries." |
3055                             tee -a $mail_msg_file >> $LOGFILE
3056                         build_ok=n
3057                 fi
3058         fi

3060         echo "Generating README.opensolaris..." >> $LOGFILE
3061         cat $SRC/tools/opensolaris/README.opensolaris.tmpl | \
3062             mkreadme_osol $CODEMGR_WS/README.opensolaris >> $LOGFILE 2>&1
3063         if (( $? != 0 )) ; then
3064                 echo "Couldn't create README.opensolaris." |
3065                     tee -a $mail_msg_file >> $LOGFILE
3066                 build_ok=n
3067         fi
3068 fi

3070 # Verify that the usual lists of files, such as exception lists,
3071 # contain only valid references to files.  If the build has failed,
3072 # then don't check the proto area.
3073 CHECK_PATHS=${CHECK_PATHS:-y}
3074 if [ "$CHECK_PATHS" = y -a "$N_FLAG" != y ]; then
3075         echo "\n==== Check lists of files ====\n" | tee -a $LOGFILE \
3076                 >>$mail_msg_file
3077         arg=-b
3078         [ "$build_ok" = y ] && arg=
3079         checkpaths $arg $checkroot 2>&1 | tee -a $LOGFILE >>$mail_msg_file
3080 fi

3082 if [ "$M_FLAG" != "y" -a "$build_ok" = y ]; then
3083         echo "\n==== Impact on file permissions ====\n" \
3084                 >> $mail_msg_file

3086         abspkgdefs=
3087         abspkg=
3088         for d in $abssrcdirs; do
3089                 if [ -d "$d/pkgdefs" ]; then
3090                         abspkgdefs="$abspkgdefs $d"
3091                 fi
3092                 if [ -d "$d/pkg" ]; then
3093                         abspkg="$abspkg $d"
3094                 fi
3095         done

3097         if [ -n "$abspkgdefs" ]; then
3098                 pmodes -qvdP \
3099                     'find $abspkgdefs -name pkginfo.tmpl -print -o \
3100                     -name .del\* -prune | sed -e 's:/pkginfo.tmpl$::' | \
3101                     sort -u' >> $mail_msg_file
3102         fi

3104         if [ -n "$abspkg" ]; then
3105                 for d in "$abspkg"; do
3106                         ( cd $d/pkg ; $MAKE -e pmodes ) >> $mail_msg_file
3107                 done
3108         fi
3109 fi

3111 if [ "$w_FLAG" = "y" -a "$build_ok" = "y" ]; then
3112         if [[ "$MULTI_PROTO" = no || "$D_FLAG" = y ]]; then
3113                 do_wsdiff DEBUG $ROOT.prev $ROOT
3114         fi
```



```
3116         if [[ "$MULTI_PROTO" = yes && "$F_FLAG" = n ]]; then
3117                 do_wsdiff non-DEBUG $ROOT-nd.prev $ROOT-nd
3118         fi
3119 fi

3121 END_DATE='date'
3122 echo "==== Nightly $maketype build completed: $END_DATE ====" | \
3123     tee -a $LOGFILE >> $build_time_file

3125 typeset -i10 hours
3126 typeset -Z2 minutes
3127 typeset -Z2 seconds

3129 elapsed_time=$SECONDS
3130 ((hours = elapsed_time / 3600 ))
3131 ((minutes = elapsed_time / 60  % 60))
3132 ((seconds = elapsed_time % 60))

3134 echo "\n==== Total build time ====" | \
3135     tee -a $LOGFILE >> $build_time_file
3136 echo "\nreal    ${hours}:${minutes}:${seconds}" | \
3137     tee -a $LOGFILE >> $build_time_file

3139 if [ "$u_FLAG" = "y" -a "$f_FLAG" = "y" -a "$build_ok" = "y" ]; then
3140         staffer cp ${SRC}/unref-${MACH}.out $PARENT_WS/usr/src/

3142         #
3143         # Produce a master list of unreferenced files -- ideally, we'd
3144         # generate the master just once after all of the nightlies
3145         # have finished, but there's no simple way to know when that
3146         # will be.  Instead, we assume that we're the last nightly to
3147         # finish and merge all of the unref-${MACH}.out files in
3148         # $PARENT_WS/usr/src/.  If we are in fact the final ${MACH} to
3149         # finish, then this file will be the authoritative master
3150         # list.  Otherwise, another ${MACH}'s nightly will eventually
3151         # overwrite ours with its own master, but in the meantime our
3152         # temporary "master" will be no worse than any older master
3153         # which was already on the parent.
3154         #

3156         set -- $PARENT_WS/usr/src/unref-*.out
3157         cp "$1" ${TMPDIR}/unref.merge
3158         shift

3160         for unreffile; do
3161                 comm -12 ${TMPDIR}/unref.merge "$unreffile" > ${TMPDIR}/unref.$$
3162                 mv ${TMPDIR}/unref.$$ ${TMPDIR}/unref.merge
3163         done

3165         staffer cp ${TMPDIR}/unref.merge $PARENT_WS/usr/src/unrefmaster.out
3166 fi

3168 #
3169 # All done save for the sweeping up.
3170 # (whichever exit we hit here will trigger the "cleanup" trap which
3171 # optionally sends mail on completion).
3172 #
3173 if [ "$build_ok" = "y" ]; then
3174         exit 0
3175 fi
3176 exit 1
```