```
**********************************************************
   35217 Mon Jan 11 15:36:58 2016
new/usr/src/Makefile.master
native tools must reliably use a native adjunct, even if that's inconvenient
While it is perhaps convenient for native tools to use updated versions
of certain things like libxml it is imperative that those versions are
_also_ build for the build machine.  Thus they need to be in the native
adjunct (even if that native adjunct is thus not /).
Fix the native adjunct to be rooted similarly to the adjunct proto (that
is, at /), and fix SMF to use it correctly
**********************************************************
```

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

#
# Copyright (c) 1989, 2010, Oracle and/or its affiliates. All rights reserved.
# Copyright (c) 2012 by Delphix. All rights reserved.
# Copyright 2014 Garrett D'Amore <garrett@damore.org>
# Copyright 2015, OmniTI Computer Consulting, Inc. All rights reserved.
# Copyright 2015 Gary Mills
# Copyright 2015 Igor Kozhukhov <ikozhukhov@gmail.com>
#

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

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

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

#
# RELEASE_BUILD should be cleared for final release builds.
# NOT_RELEASE_BUILD is exactly what the name implies.
#
# __GNUC toggles the building of ON components using gcc and related tools.
# Normally set to '#', set it to '' to do gcc build.
#
# The declaration POUND_SIGN is always '#'. This is needed to get around the
```



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

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

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

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

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

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

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

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

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

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

# define runtime JAVA_HOME, primarily for cmd/pools/poold
JAVA_HOME=      /usr/java
# define buildtime JAVA_ROOT
```

```
121 JAVA_ROOT=      /usr/java

123 GCC_ROOT=       /opt/gcc/4.4.4
124 GCCLIBDIR=      $(GCC_ROOT)/lib
125 GCCLIBDIR64=    $(GCC_ROOT)/lib/$(MACH64)

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

129 RPCGEN=         /usr/bin/rpcgen
130 STABS=          $(ONBLD_TOOLS)/bin/$(MACH)/stabs
131 ELFEXTRACT=     $(ONBLD_TOOLS)/bin/$(MACH)/elfextract
132 MBH_PATCH=      $(ONBLD_TOOLS)/bin/$(MACH)/mbh_patch
133 ECHO=           echo
134 INS=            install
135 TRUE=           true
136 SYMLINK=        /usr/bin/ln -s
137 LN=             /usr/bin/ln
138 CHMOD=          /usr/bin/chmod
139 MV=             /usr/bin/mv -f
140 RM=             /usr/bin/rm -f
141 CUT=            /usr/bin/cut
142 NM=             /usr/ccs/bin/nm
143 DIFF=           /usr/bin/diff
144 GREP=           /usr/bin/grep
145 EGREP=          /usr/bin/egrep
146 ELFWRAP=        /usr/bin/elfwrap
147 KSH93=          /usr/bin/ksh93
148 SED=            /usr/bin/sed
149 AWK=            /usr/bin/nawk
150 CP=             /usr/bin/cp -f
151 MCS=            /usr/ccs/bin/mcs
152 CAT=            /usr/bin/cat
153 ELFDUMP=        /usr/ccs/bin/elfdump
154 M4=             /usr/bin/m4
155 STRIP=          /usr/ccs/bin/strip
156 LEX=            /usr/ccs/bin/lex
157 FLEX=           /usr/bin/flex
158 YACC=           /usr/ccs/bin/yacc
159 CPP=            /usr/lib/cpp
160 JAVAC=          $(JAVA_ROOT)/bin/javac
161 JAVAH=          $(JAVA_ROOT)/bin/javah
162 JAVADOC=        $(JAVA_ROOT)/bin/javadoc
163 RMIC=           $(JAVA_ROOT)/bin/rmic
164 JAR=            $(JAVA_ROOT)/bin/jar
165 CTFCONVERT=     $(ONBLD_TOOLS)/bin/$(MACH)/ctfconvert
166 CTFMERGE=       $(ONBLD_TOOLS)/bin/$(MACH)/ctfmerge
167 CTFSTABS=       $(ONBLD_TOOLS)/bin/$(MACH)/ctfstabs
168 CTFSTRIP=       $(ONBLD_TOOLS)/bin/$(MACH)/ctfstrip
169 NDRGEN=         $(ONBLD_TOOLS)/bin/$(MACH)/ndrgen
170 GENOFFSETS=     $(ONBLD_TOOLS)/bin/genoffsets
171 XREF=           $(ONBLD_TOOLS)/bin/xref
172 FIND=           /usr/bin/find
173 PERL=           /usr/bin/perl
174 PERL_VERSION=   5.10.0
175 PERL_PKGVERS=   -510
176 PERL_ARCH =             i86pc-solaris-64int
177 $(SPARC_BLD)PERL_ARCH = sun4-solaris-64int
178 PYTHON_26=      /usr/bin/python2.6
179 PYTHON=         $(PYTHON_26)
180 SORT=           /usr/bin/sort
181 TOUCH=          /usr/bin/touch
182 WC=             /usr/bin/wc
183 XARGS=          /usr/bin/xargs
184 ELFEDIT=        /usr/bin/elfedit
185 ELFSIGN=        /usr/bin/elfsign
186 DTRACE=         /usr/sbin/dtrace -xnolibs
```



```
187 UNIQ=           /usr/bin/uniq
188 TAR=            /usr/bin/tar
189 ASTBINDIR=      /usr/ast/bin
190 MSGCC=          $(ASTBINDIR)/msgcc
191 MSGFMT=         /usr/bin/msgfmt -s

193 FILEMODE=       644
194 DIRMODE=        755

196 # Declare that nothing should be built in parallel.
197 # Individual Makefiles can use the .PARALLEL target to declare otherwise.
198 .NO_PARALLEL:

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

211 DOT_H_CHECK=    \
212         @$(ECHO) "checking $<"; $(CSTYLE) $< $(CSTYLE_TAIL); \
213         $(HDRCHK) $< $(HDRCHK_TAIL)

215 DOT_X_CHECK=    \
216         @$(ECHO) "checking $<"; $(RPCGEN) -C -h $< | $(CSTYLE) $(CSTYLE_TAIL); \
217         $(RPCGEN) -C -h $< | $(HDRCHK) $< $(HDRCHK_TAIL)

219 DOT_C_CHECK=    \
220         @$(ECHO) "checking $<"; $(CSTYLE) $< $(CSTYLE_TAIL)

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

229 INS.file=       $(RM) $@; $(INS) -s -m $(FILEMODE) -f $(@D) $<
230 INS.dir=        $(INS) -s -d -m $(DIRMODE) $@
231 # installs and renames at once
232 #
233 INS.rename=     $(INS.file); $(MV) $(@D)/$(<F) $@

235 # install a link
236 INSLINKTARGET=  $<
237 INS.link=       $(RM) $@; $(LN) $(INSLINKTARGET) $@
238 INS.symlink=    $(RM) $@; $(SYMLINK) $(INSLINKTARGET) $@

240 #
241 # Python bakes the mtime of the .py file into the compiled .pyc and
242 # rebuilds if the baked-in mtime != the mtime of the source file
243 # (rather than only if it's less than), thus when installing python
244 # files we must make certain to not adjust the mtime of the source
245 # (.py) file.
246 #
247 INS.pyfile=     $(INS.file); $(TOUCH) -r $< $@

249 # MACH must be set in the shell environment per uname -p on the build host
250 # More specific architecture variables should be set in lower makefiles.
251 #
252 # MACH64 is derived from MACH, and BUILD64 is set to '#' for
```

```
253 # architectures on which we do not build 64-bit versions.
254 # (There are no such architectures at the moment.)
255 #
256 # Set BUILD64=# in the environment to disable 64-bit amd64
257 # builds on i386 machines.

259 MACH64_1=       $(MACH:sparc=sparcv9)
260 MACH64=         $(MACH64_1:i386=amd64)

262 MACH32_1=       $(MACH:sparc=sparcv7)
263 MACH32=         $(MACH32_1:i386=i86)

265 sparc_BUILD64=
266 i386_BUILD64=
267 BUILD64=        $($(MACH)_BUILD64)

269 #
270 # C compiler mode. Future compilers may change the default on us,
271 # so force extended ANSI mode globally. Lower level makefiles can
272 # override this by setting CCMODE.
273 #
274 CCMODE=                 -Xa
275 CCMODE64=               -Xa

277 #
278 # C compiler verbose mode. This is so we can enable it globally,
279 # but turn it off in the lower level makefiles of things we cannot
280 # (or aren't going to) fix.
281 #
282 CCVERBOSE=              -v

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

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

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

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



```
319 # One optimization the compiler might perform is to turn this:
320 #       #pragma weak foo
321 #       extern int foo;
322 #       if (&foo)
323 #               foo = 5;
324 # into
325 #       foo = 5;
326 # Since we do some of this (foo might be referenced in common kernel code
327 # but provided only for some cpu modules or platforms), we disable this
328 # optimization.
329 #
330 sparc_CCUNBOUND = -Wd,-xsafe=unboundsym
331 i386_CCUNBOUND  =
332 CCUNBOUND       = $($(MACH)_CCUNBOUND)

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

342 # assembler '-xarch' flag.  Different from compiler '-xarch' flag.
343 sparc_AS_XARCH=         -xarch=v8plus
344 sparcv9_AS_XARCH=       -xarch=v9
345 i386_AS_XARCH=
346 amd64_AS_XARCH=         -xarch=amd64 -P -Ui386 -U__i386

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

363 SAVEARGS=               -Wu,-save_args
364 amd64_STAND_FLAGS       += $(SAVEARGS)

366 STAND_FLAGS_32 = $($(MACH)_STAND_FLAGS)
367 STAND_FLAGS_64 = $($(MACH64)_STAND_FLAGS)

369 #
370 # disable the incremental linker
371 ILDOFF=                 -xildoff
372 #
373 XDEPEND=                -xdepend
374 XFFLAG=                 -xF=%all
375 XESS=                   -xs
376 XSTRCONST=              -xstrconst

378 #
379 # turn warnings into errors (C)
380 CERRWARN = -erttags=yes -errwarn=%all
381 CERRWARN += -erroff=E_EMPTY_TRANSLATION_UNIT
382 CERRWARN += -erroff=E_STATEMENT_NOT_REACHED

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

```
385 CERRWARN += -_gcc=-Wno-sign-compare
386 CERRWARN += -_gcc=-Wno-unknown-pragmas
387 CERRWARN += -_gcc=-Wno-unused-parameter
388 CERRWARN += -_gcc=-Wno-missing-field-initializers

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

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

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

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

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

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

421 #
422 sparc_COPTFLAG=         -xO3
423 sparcv9_COPTFLAG=       -xO3
424 i386_COPTFLAG=          -O
425 amd64_COPTFLAG=         -xO3

427 COPTFLAG= $($(MACH)_COPTFLAG)
428 COPTFLAG64= $($(MACH64)_COPTFLAG)

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

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

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

446 #
447 # Default debug format for Sun Studio 11 is dwarf, so force it to
448 # generate stabs.
449 #
450 DEBUGFORMAT=    -xdebugformat=stabs
```



```
452 #
453 # Flags used to build in debug mode for ctf generation.  Bugs in the Devpro
454 # compilers currently prevent us from building with cc-emitted DWARF.
455 #
456 CTF_FLAGS_sparc = -g -Wc,-Qiselect-T1 $(C99MODE) $(CNOGLOBAL) $(CDWARFSTR)
457 CTF_FLAGS_i386  = -g $(C99MODE) $(CNOGLOBAL) $(CDWARFSTR)

459 CTF_FLAGS_sparcv9       = $(CTF_FLAGS_sparc)
460 CTF_FLAGS_amd64         = $(CTF_FLAGS_i386)

462 # Sun Studio produces broken userland code when saving arguments.
463 $(__GNUC)CTF_FLAGS_amd64 += $(SAVEARGS)

465 CTF_FLAGS_32    = $(CTF_FLAGS_$(MACH)) $(DEBUGFORMAT)
466 CTF_FLAGS_64    = $(CTF_FLAGS_$(MACH64)) $(DEBUGFORMAT)
467 CTF_FLAGS       = $(CTF_FLAGS_32)

469 #
470 # Flags used with genoffsets
471 #
472 GOFLAGS = -_noecho \
473         $(CALLSYMS) \
474         $(CDWARFSTR)

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

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

482 #
483 # tradeoff time for space (smaller is better)
484 #
485 sparc_SPACEFLAG         = -xspace -W0,-Lt
486 sparcv9_SPACEFLAG       = -xspace -W0,-Lt
487 i386_SPACEFLAG          = -xspace
488 amd64_SPACEFLAG         =

490 SPACEFLAG               = $($(MACH)_SPACEFLAG)
491 SPACEFLAG64             = $($(MACH64)_SPACEFLAG)

493 #
494 # The Sun Studio 11 compiler has changed the behaviour of integer
495 # wrap arounds and so a flag is needed to use the legacy behaviour
496 # (without this flag panics/hangs could be exposed within the source).
497 #
498 sparc_IROPTFLAG         = -W2,-xwrap_int
499 sparcv9_IROPTFLAG       = -W2,-xwrap_int
500 i386_IROPTFLAG          =
501 amd64_IROPTFLAG         =

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

506 sparc_XREGSFLAG         = -xregs=no%appl
507 sparcv9_XREGSFLAG       = -xregs=no%appl
508 i386_XREGSFLAG          =
509 amd64_XREGSFLAG         =

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

514 # dmake SOURCEDEBUG=yes ... enables source-level debugging information, and
515 # avoids stripping it.
516 SOURCEDEBUG     = $(POUND_SIGN)
```

```
517 SRCDBGBLD       = $(SOURCEDEBUG:yes=)

519 #
520 # These variables are intended ONLY for use by developers to safely pass extra
521 # flags to the compilers without unintentionally overriding Makefile-set
522 # flags.  They should NEVER be set to any value in a Makefile.
523 #
524 # They come last in the associated FLAGS variable such that they can
525 # explicitly override things if necessary, there are gaps in this, but it's
526 # the best we can manage.
527 #
528 CUSERFLAGS              =
529 CUSERFLAGS64            = $(CUSERFLAGS)
530 CCUSERFLAGS             =
531 CCUSERFLAGS64           = $(CCUSERFLAGS)

533 CSOURCEDEBUGFLAGS       =
534 CCSOURCEDEBUGFLAGS      =
535 $(SRCDBGBLD)CSOURCEDEBUGFLAGS   = -g -xs
536 $(SRCDBGBLD)CCSOURCEDEBUGFLAGS  = -g -xs

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

555 DTEXTDOM=-DTEXT_DOMAIN=\"$(TEXT_DOMAIN)\"       # For messaging.
556 DTS_ERRNO=-D_TS_ERRNO
557 CPPFLAGS.first= # Please keep empty.  Only lower makefiles should set this.
558 CPPFLAGS.master=$(DTEXTDOM) $(DTS_ERRNO) \
559         $(ENVCPPFLAGS1) $(ENVCPPFLAGS2) $(ENVCPPFLAGS3) $(ENVCPPFLAGS4) \
560         $(ADJUNCT_PROTO:%=-I%/usr/include)
561 CPPFLAGS.native=$(ENVCPPFLAGS1) $(ENVCPPFLAGS2) $(ENVCPPFLAGS3) \
562                 $(ENVCPPFLAGS4) -I$(NATIVE_ADJUNCT)/usr/include
562                 $(ENVCPPFLAGS4) -I$(NATIVE_ADJUNCT)/include
563 CPPFLAGS=       $(CPPFLAGS.first) $(CPPFLAGS.master)
564 AS_CPPFLAGS=    $(CPPFLAGS.first) $(CPPFLAGS.master)
565 JAVAFLAGS=      -source 1.6 -target 1.6 -Xlint:deprecation,-options

567 #
568 # For source message catalogue
569 #
570 .SUFFIXES: $(SUFFIXES) .i .po
571 MSGROOT= $(ROOT)/catalog
572 MSGDOMAIN= $(MSGROOT)/$(TEXT_DOMAIN)
573 MSGDOMAINPOFILE = $(MSGDOMAIN)/$(POFILE)
574 DCMSGDOMAIN= $(MSGROOT)/LC_TIME/$(TEXT_DOMAIN)
575 DCMSGDOMAINPOFILE = $(DCMSGDOMAIN)/$(DCFILE:.dc=.po)

577 CLOBBERFILES += $(POFILE) $(POFILES)
578 COMPILE.cpp= $(CC) -E -C $(CFLAGS) $(CPPFLAGS)
579 XGETTEXT= /usr/bin/xgettext
580 XGETFLAGS= -c TRANSLATION_NOTE
581 GNUXGETTEXT= /usr/gnu/bin/xgettext
```



```
582 GNUXGETFLAGS= --add-comments=TRANSLATION_NOTE --keyword=_ \
583         --strict --no-location --omit-header
584 BUILD.po= $(XGETTEXT) $(XGETFLAGS) -d $(<F) $<.i ;\
585         $(RM)   $@ ;\
586         $(SED) "/^domain/d" < $(<F).po > $@ ;\
587         $(RM) $(<F).po $<.i

589 #
590 # This is overwritten by local Makefile when PROG is a list.
591 #
592 POFILE= $(PROG).po

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

611 sparc_CCOPTFLAG=        -O
612 sparcv9_CCOPTFLAG=      -O
613 i386_CCOPTFLAG=         -O
614 amd64_CCOPTFLAG=        -O

616 CCOPTFLAG=      $($(MACH)_CCOPTFLAG)
617 CCOPTFLAG64=    $($(MACH64)_CCOPTFLAG)
618 CCFLAGS=        $(CCOPTFLAG) $($(MACH)_CCFLAGS) $(CCSOURCEDEBUGFLAGS) \
619                 $(CCUSERFLAGS)
620 CCFLAGS64=      $(CCOPTFLAG64) $($(MACH64)_CCFLAGS) $(CCSOURCEDEBUGFLAGS) \
621                 $(CCUSERFLAGS64)

623 #
624 #
625 #
626 ELFWRAP_FLAGS   =
627 ELFWRAP_FLAGS64 =       -64

629 #
630 # Various mapfiles that are used throughout the build, and delivered to
631 # /usr/lib/ld.
632 #
633 MAPFILE.NED_i386 =      $(SRC)/common/mapfiles/common/map.noexdata
634 MAPFILE.NED_sparc =
635 MAPFILE.NED =           $(MAPFILE.NED_$(MACH))
636 MAPFILE.PGA =           $(SRC)/common/mapfiles/common/map.pagealign
637 MAPFILE.NES =           $(SRC)/common/mapfiles/common/map.noexstk
638 MAPFILE.FLT =           $(SRC)/common/mapfiles/common/map.filter
639 MAPFILE.LEX =           $(SRC)/common/mapfiles/common/map.lex.yy

641 #
642 # Generated mapfiles that are compiler specific, and used throughout the
643 # build.  These mapfiles are not delivered in /usr/lib/ld.
644 #
645 MAPFILE.NGB_sparc=      $(SRC)/common/mapfiles/gen/sparc_cc_map.noexeglobs
646 $(__GNUC64)MAPFILE.NGB_sparc= \
647                         $(SRC)/common/mapfiles/gen/sparc_gcc_map.noexeglobs
```

```
MAPFILE.NGB_sparcv9=    $(SRC)/common/mapfiles/gen/sparcv9_cc_map.noexeglobs
$(__GNUC64)MAPFILE.NGB_sparcv9= \
                        $(SRC)/common/mapfiles/gen/sparcv9_gcc_map.noexeglobs
MAPFILE.NGB_i386=       $(SRC)/common/mapfiles/gen/i386_cc_map.noexeglobs
$(__GNUC64)MAPFILE.NGB_i386= \
                        $(SRC)/common/mapfiles/gen/i386_gcc_map.noexeglobs
MAPFILE.NGB_amd64=      $(SRC)/common/mapfiles/gen/amd64_cc_map.noexeglobs
$(__GNUC64)MAPFILE.NGB_amd64= \
                        $(SRC)/common/mapfiles/gen/amd64_gcc_map.noexeglobs
MAPFILE.NGB =           $(MAPFILE.NGB_$(MACH))

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

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

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

#
# Define compilation macros.
#
COMPILE.c=      $(CC) $(CFLAGS) $(CPPFLAGS) -c
COMPILE64.c=    $(CC) $(CFLAGS64) $(CPPFLAGS) -c
COMPILE.cc=     $(CCC) $(CCFLAGS) $(CPPFLAGS) -c
COMPILE64.cc=   $(CCC) $(CCFLAGS64) $(CPPFLAGS) -c
COMPILE.s=      $(AS) $(ASFLAGS) $(AS_CPPFLAGS)
COMPILE64.s=    $(AS) $(ASFLAGS) $($(MACH64)_AS_XARCH) $(AS_CPPFLAGS)
COMPILE.d=      $(DTRACE) -G -32
COMPILE64.d=    $(DTRACE) -G -64
COMPILE.b=      $(ELFWRAP) $(ELFWRAP_FLAGS$(CLASS))
COMPILE64.b=    $(ELFWRAP) $(ELFWRAP_FLAGS$(CLASS))

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

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

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



```
LINK64.cc=      $(CCC) $(CCFLAGS64) $(CPPFLAGS) $(NORUNPATH) \
                $(LDFLAGS) $(CCNEEDED)

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

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

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

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

# Define native compilation macros
#

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

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

# Specify platform compiler versions for languages
# that we use (currently only c and c++).
#
sparc_CC=               $(ONBLD_TOOLS)/bin/$(MACH)/cw -_cc
```

```
780 $(__GNUC)sparc_CC=      $(ONBLD_TOOLS)/bin/$(MACH)/cw -_gcc
781 sparc_CCC=              $(ONBLD_TOOLS)/bin/$(MACH)/cw -_CC
782 $(__GNUC)sparc_CCC=     $(ONBLD_TOOLS)/bin/$(MACH)/cw -_g++
783 sparc_CPP=              /usr/ccs/lib/cpp
784 sparc_AS=               /usr/ccs/bin/as -xregsym=no
785 sparc_LD=               /usr/ccs/bin/ld
786 sparc_LINT=             $(SPRO_ROOT)/sunstudio12.1/bin/lint

788 sparcv9_CC=            $(ONBLD_TOOLS)/bin/$(MACH)/cw -_cc
789 $(__GNUC64)sparcv9_CC= $(ONBLD_TOOLS)/bin/$(MACH)/cw -_gcc
790 sparcv9_CCC=           $(ONBLD_TOOLS)/bin/$(MACH)/cw -_CC
791 $(__GNUC64)sparcv9_CCC=$(ONBLD_TOOLS)/bin/$(MACH)/cw -_g++
792 sparcv9_CPP=           /usr/ccs/lib/cpp
793 sparcv9_AS=            /usr/ccs/bin/as -xregsym=no
794 sparcv9_LD=            /usr/ccs/bin/ld
795 sparcv9_LINT=          $(SPRO_ROOT)/sunstudio12.1/bin/lint

797 i386_CC=              $(ONBLD_TOOLS)/bin/$(MACH)/cw -_cc
798 $(__GNUC)i386_CC=     $(ONBLD_TOOLS)/bin/$(MACH)/cw -_gcc
799 i386_CCC=             $(ONBLD_TOOLS)/bin/$(MACH)/cw -_CC
800 $(__GNUC)i386_CCC=    $(ONBLD_TOOLS)/bin/$(MACH)/cw -_g++
801 i386_CPP=             /usr/ccs/lib/cpp
802 i386_AS=              /usr/ccs/bin/as
803 $(__GNUC)i386_AS=     $(ONBLD_TOOLS)/bin/$(MACH)/aw
804 i386_LD=              /usr/ccs/bin/ld
805 i386_LINT=            $(SPRO_ROOT)/sunstudio12.1/bin/lint

807 amd64_CC=             $(ONBLD_TOOLS)/bin/$(MACH)/cw -_cc
808 $(__GNUC64)amd64_CC=  $(ONBLD_TOOLS)/bin/$(MACH)/cw -_gcc
809 amd64_CCC=            $(ONBLD_TOOLS)/bin/$(MACH)/cw -_CC
810 $(__GNUC64)amd64_CCC= $(ONBLD_TOOLS)/bin/$(MACH)/cw -_g++
811 amd64_CPP=            /usr/ccs/lib/cpp
812 amd64_AS=             $(ONBLD_TOOLS)/bin/$(MACH)/aw
813 amd64_LD=             /usr/ccs/bin/ld
814 amd64_LINT=           $(SPRO_ROOT)/sunstudio12.1/bin/lint

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

823 #
824 # Makefile.master.64 overrides these settings
825 #
826 CC=                  $(NATIVECC)
827 CCC=                 $(NATIVECCC)
828 CPP=                 $(NATIVECPP)
829 AS=                  $(NATIVEAS)
830 LD=                  $(NATIVELD)
831 LINT=                $(NATIVELINT)

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

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

842 BDIRECT=        -Bdirect
843 BDYNAMIC=       -Bdynamic
844 BLOCAL=         -Blocal
845 BNODIRECT=      -Bnodirect
```



```
846 BREDUCE=         -Breduce
847 BSTATIC=         -Bstatic

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

871 GSHARED=         -G
872 CCMT=            -mt

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

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

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

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

899 AS_PICFLAGS=            $(C_PICFLAGS)
900 AS_BIGPICFLAGS=         $(C_BIGPICFLAGS)

902 #
903 # Default label for CTF sections
904 #
905 CTFCVTFLAGS=            -i -L VERSION

907 #
908 # Override to pass module-specific flags to ctfmerge.  Currently used only by
909 # krtld to turn on fuzzy matching, and source-level debugging to inhibit
910 # stripping.
911 #
```

```
912 CTFMRGFLAGS=

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

916 ELFSIGN_O=      $(TRUE)
917 ELFSIGN_CRYPTO= $(ELFSIGN_O)
918 ELFSIGN_OBJECT= $(ELFSIGN_O)

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

972 PROCESS_COMMENT=  @?${MCS} -d -a $(RELEASE_CM) -a $(DEV_CM)
973 $(RELEASE_BUILD)PROCESS_COMMENT=   @?${MCS} -d -a $(RELEASE_CM)

975 STRIP_STABS=                       $(STRIP) -x $@
976 $(SRCDBGBLD)STRIP_STABS=           :
```



```
978 POST_PROCESS_O=
979 POST_PROCESS_A=
980 POST_PROCESS_SO=        $(PROCESS_COMMENT) $@ ; $(STRIP_STABS) ; \
981                         $(ELFSIGN_OBJECT)
982 POST_PROCESS=           $(PROCESS_COMMENT) $@ ; $(STRIP_STABS) ; \
983                         $(ELFSIGN_OBJECT)

985 #
986 # chk4ubin is a tool that inspects a module for a symbol table
987 # ELF section size which can trigger an OBP bug on older platforms.
988 # This problem affects only specific sun4u bootable modules.
989 #
990 CHK4UBIN=       $(ONBLD_TOOLS)/bin/$(MACH)/chk4ubin
991 CHK4UBINFLAGS=
992 CHK4UBINARY=    $(CHK4UBIN) $(CHK4UBINFLAGS) $@

994 #
995 # PKGARCHIVE specifies the default location where packages should be
996 # placed if built.
997 #
998 $(RELEASE_BUILD)PKGARCHIVESUFFIX=       -nd
999 PKGARCHIVE=$(SRC)/../../packages/$(MACH)/nightly$(PKGARCHIVESUFFIX)

1001 #
1002 # The repositories will be created with these publisher settings.  To
1003 # update an image to the resulting repositories, this must match the
1004 # publisher name provided to "pkg set-publisher."
1005 #
1006 PKGPUBLISHER_REDIST=    on-nightly
1007 PKGPUBLISHER_NONREDIST= on-extra

1009 #       Default build rules which perform comment section post-processing.
1010 #
1011 .c:
1012         $(LINK.c) -o $@ $< $(LDLIBS)
1013         $(POST_PROCESS)
1014 .c.o:
1015         $(COMPILE.c) $(OUTPUT_OPTION) $< $(CTFCONVERT_HOOK)
1016         $(POST_PROCESS_O)
1017 .c.a:
1018         $(COMPILE.c) -o $% $<
1019         $(PROCESS_COMMENT) $%
1020         $(AR) $(ARFLAGS) $@ $%
1021         $(RM) $%
1022 .s.o:
1023         $(COMPILE.s) -o $@ $<
1024         $(POST_PROCESS_O)
1025 .s.a:
1026         $(COMPILE.s) -o $% $<
1027         $(PROCESS_COMMENT) $%
1028         $(AR) $(ARFLAGS) $@ $%
1029         $(RM) $%
1030 .cc:
1031         $(LINK.cc) -o $@ $< $(LDLIBS)
1032         $(POST_PROCESS)
1033 .cc.o:
1034         $(COMPILE.cc) $(OUTPUT_OPTION) $<
1035         $(POST_PROCESS_O)
1036 .cc.a:
1037         $(COMPILE.cc) -o $% $<
1038         $(AR) $(ARFLAGS) $@ $%
1039         $(PROCESS_COMMENT) $%
1040         $(RM) $%
1041 .y:
1042         $(YACC.y) $<
1043         $(LINK.c) -o $@ y.tab.c $(LDLIBS)
```

```
1044         $(POST_PROCESS)
1045         $(RM) y.tab.c
1046 .y.o:
1047         $(YACC.y) $<
1048         $(COMPILE.c) -o $@ y.tab.c $(CTFCONVERT_HOOK)
1049         $(POST_PROCESS_O)
1050         $(RM) y.tab.c
1051 .l:
1052         $(RM) $*.c
1053         $(LEX.l) $< > $*.c
1054         $(LINK.c) -o $@ $*.c -ll $(LDLIBS)
1055         $(POST_PROCESS)
1056         $(RM) $*.c
1057 .l.o:
1058         $(RM) $*.c
1059         $(LEX.l) $< > $*.c
1060         $(COMPILE.c) -o $@ $*.c $(CTFCONVERT_HOOK)
1061         $(POST_PROCESS_O)
1062         $(RM) $*.c

1064 .bin.o:
1065         $(COMPILE.b) -o $@ $<
1066         $(POST_PROCESS_O)

1068 .java.class:
1069         $(COMPILE.java) $<

1071 # Bourne and Korn shell script message catalog build rules.
1072 # We extract all gettext strings with sed(1) (being careful to permit
1073 # multiple gettext strings on the same line), weed out the dups, and
1074 # build the catalogue with awk(1).

1076 .sh.po .ksh.po:
1077         $(SED) -n -e ":a"                                   \
1078                   -e "h"                                        \
1079                   -e "s/.*gettext *\(\"[^\"]*\"\).*/\1/p"       \
1080                   -e "x"                                        \
1081                   -e "s/\(.*\)gettext *\"[^\"]*\"\(.*\)/\1\2/"  \
1082                   -e "t a"                                      \
1083                 $< | sort -u | $(AWK) '{ print "msgid\t" $$0 "\nmsgstr" }' > $@

1085 #
1086 # Python and Perl executable and message catalog build rules.
1087 #
1088 .SUFFIXES: .pl .pm .py .pyc

1090 .pl:
1091         $(RM) $@;
1092         $(SED) -e "s@TEXT_DOMAIN@\"$(TEXT_DOMAIN)\"@" $< > $@;
1093         $(CHMOD) +x $@

1095 .py:
1096         $(RM) $@; $(CAT) $< > $@; $(CHMOD) +x $@

1098 .py.pyc:
1099         $(RM) $@
1100         $(PYTHON) -mpy_compile $<
1101         @[ $(<)c = $@ ] || $(MV) $(<)c $@

1103 .py.po:
1104         $(GNUXGETTEXT) $(GNUXGETFLAGS) -d $(<F:%.py=%) $< ;

1106 .pl.po .pm.po:
1107         $(XGETTEXT) $(XGETFLAGS) -d $(<F) $< ;
1108         $(RM)   $@ ;
1109         $(SED) "/^domain/d" < $(<F).po > $@ ;
```



```
1110         $(RM) $(<F).po

1112 #
1113 # When using xgettext, we want messages to go to the default domain,
1114 # rather than the specified one.  This special version of the
1115 # COMPILE.cpp macro effectively prevents expansion of TEXT_DOMAIN,
1116 # causing xgettext to put all messages into the default domain.
1117 #
1118 CPPFORPO=$(COMPILE.cpp:\"$(TEXT_DOMAIN)\"=TEXT_DOMAIN)

1120 .c.i:
1121         $(CPPFORPO) $< > $@

1123 .h.i:
1124         $(CPPFORPO) $< > $@

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

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

1136 .c.po:
1137         $(CPPFORPO) $< > $<.i
1138         $(BUILD.po)

1140 .cc.po:
1141         $(CPPFORPO) $< > $<.i
1142         $(BUILD.po)

1144 .y.po:
1145         $(YACC) -d $<
1146         $(CPPFORPO) y.tab.c  > $<.i
1147         $(BUILD.po)
1148         $(RM) y.tab.c

1150 .l.po:
1151         $(LEX) $<
1152         $(CPPFORPO) lex.yy.c  > $<.i
1153         $(BUILD.po)
1154         $(RM) lex.yy.c

1156 #
1157 # Rules to perform stylistic checks
1158 #
1159 .SUFFIXES: .x .xml .check .xmlchk

1161 .h.check:
1162         $(DOT_H_CHECK)

1164 .x.check:
1165         $(DOT_X_CHECK)

1167 .xml.xmlchk:
1168         $(MANIFEST_CHECK)

1170 #
1171 # Include rules to render automated sccs get rules "safe".
1172 #
1173 include $(SRC)/Makefile.noget
```

```
**********************************************************
    2515 Mon Jan 11 15:36:58 2016
new/usr/src/cmd/hal/Makefile.hal
gag Studio warnings that only occur with an adjunct proto
Studio has no equivalent to -isystem, and no option to its -errhdr that
seems equivalent.  As such, we have to gag the warnings that are induced
from the 3rd party headers when those headers are found via the adjunct,
as they will not match Studio's seemingly hard-coded exception on
'/usr/include'
**********************************************************
#
# 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 2009 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#
# Definitions common for HAL code and consumers
#

HAL_VERSION =           0.5.8

ROOTLIB_HAL =           $(ROOTLIB)/hal
ROOTLIB_HAL_SCRIPTS =   $(ROOTLIB)/hal

ROOT_HAL_FDI =          $(ROOT)/etc/hal/fdi

HAL_USER =              daemon
HAL_GROUP =             daemon

# derived from the generated config.h
HAL_CONFIG_CPPFLAGS =   -DPACKAGE_DATA_DIR=\"/usr/lib\" \
                        -DPACKAGE_BIN_DIR=\"/usr/bin\" \
                        -DPACKAGE_LIBEXEC_DIR=\"/usr/lib/hal\" \
                        -DPACKAGE_SCRIPT_DIR=\"/usr/lib/hal\" \
                        -DPACKAGE_LOCALSTATEDIR=\"/var\" \
                        -DPACKAGE_SYSCONF_DIR=\"/etc\" \
                        -DPACKAGE_LOCALE_DIR=\"/usr/lib/locale\" \
                        -DPACKAGE_VERSION=\"$(HAL_VERSION)\" \
                        -DPACKAGE_STRING=\""hal $(HAL_VERSION)"\" \
                        -DHALD_PID_FILE=\"/var/run/hald/pid\" \
                        -DHALD_SOCKET_DIR=\"/var/run/hald\" \
                        -DHAVE_ASPRINTF \
                        -DHAVE_POLKIT \
                        -DHWDATA_DIR=\"/usr/share/hwdata\" \
                        -DHAL_USER=\"$(HAL_USER)\" \
                        -DHAL_GROUP=\"$(HAL_GROUP)\"

HAL_DBUS_CPPFLAGS =     -DDBUS_API_SUBJECT_TO_CHANGE \
```



```
                        -DDBUS_SYSTEMD_DIR=\"/etc/dbus-1/system.d\" \
                        -I$(ADJUNCT_PROTO)/usr/include/dbus-1.0 \
                        -I$(ADJUNCT_PROTO)/usr/lib/dbus-1.0/include

HAL_GLIB_CPPFLAGS =     -I$(ADJUNCT_PROTO)/usr/include/glib-2.0 \
                        -I$(ADJUNCT_PROTO)/usr/lib/glib-2.0/include

HAL_GETTEXT_PACKAGE =   $(TEXT_DOMAIN)

CERRWARN +=             -_gcc=-Wno-unused-variable
CERRWARN +=             -_gcc=-Wno-unused-label
CERRWARN +=             -_gcc=-Wno-unused-value
CERRWARN +=             -_gcc=-Wno-extra
CERRWARN +=             -_gcc=-Wno-parentheses
CERRWARN +=             -_gcc=-Wno-address
CERRWARN +=             -_gcc=-Wno-unused-function

# This is, unfortunately, from the glib headers.  Studio provides no
# equivalent to -isystem
CERRWARN +=     -_cc=-erroff=E_INTEGER_OVERFLOW_DETECTED
#endif /* ! codereview */
```

```
**********************************************************
    2231 Mon Jan 11 15:36:59 2016
new/usr/src/cmd/latencytop/Makefile.com
gag Studio warnings that only occur with an adjunct proto
Studio has no equivalent to -isystem, and no option to its -errhdr that
seems equivalent.  As such, we have to gag the warnings that are induced
from the 3rd party headers when those headers are found via the adjunct,
as they will not match Studio's seemingly hard-coded exception on
'/usr/include'
**********************************************************
```

```
#
# 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) 2008-2009, Intel Corporation.
# All Rights Reserved.
#

PROG = latencytop
OBJS = latencytop.o display.o dwrapper.o klog.o stat.o table.o util.o
SRCS = $(OBJS:%.o=../common/%.c)

include ../../Makefile.cmd

CFLAGS += $(CCVERBOSE)
CFLAGS64 += $(CCVERBOSE)

CERRWARN += -_gcc=-Wno-uninitialized

CPPFLAGS += -DEMBED_CONFIGS -I$(ADJUNCT_PROTO)/usr/include/glib-2.0 \
        -I$(ADJUNCT_PROTO)/usr/lib/glib-2.0/include
C99MODE = $(C99_ENABLE)
LDLIBS += -lcurses -ldtrace
all install     := LDLIBS += -lglib-2.0

LINTFLAGS += -erroff=E_NAME_USED_NOT_DEF2
LINTFLAGS += -erroff=E_FUNC_RET_ALWAYS_IGNOR2
LINTFLAGS += -erroff=E_STATIC_UNUSED
#endif /* ! codereview */
LINTFLAGS64 += -erroff=E_NAME_USED_NOT_DEF2
LINTFLAGS64 += -erroff=E_FUNC_RET_ALWAYS_IGNOR2
LINTFLAGS64 += -erroff=E_STATIC_UNUSED
#endif /* ! codereview */

FILEMODE = 0555

ELFWRAP = elfwrap
WRAPOBJ = latencytop_wrap.o
```



```
CLEANFILES += $(OBJS) $(WRAPOBJ) ./latencytop_d ./latencytop_trans

.KEEP_STATE:

all: $(PROG)

install:        $(SUBDIRS)
        -$(RM) $(ROOTPROG)
        -$(LN) $(ISAEXEC) $(ROOTPROG)

$(PROG): $(OBJS) $(WRAPOBJ)
        $(LINK.c) -o $@ $(OBJS) $(WRAPOBJ) $(LDLIBS)
        $(POST_PROCESS)

$(WRAPOBJ): latencytop_d latencytop_trans
        $(ELFWRAP) $(WRAPOPT) -o $(WRAPOBJ) latencytop_d latencytop_trans

latencytop_d:
        cp ../common/latencytop.d ./latencytop_d

latencytop_trans:
        cp ../common/latencytop.trans ./latencytop_trans

clean:
        $(RM) $(CLEANFILES)

lint:   lint_SRCS

%.o: ../common/%.c
        $(COMPILE.c) $<

include ../../Makefile.targ
```

```
**********************************************************
    1478 Mon Jan 11 15:36:59 2016
new/usr/src/cmd/mdb/intel/ia32/libpython2.6/Makefile
gag Studio warnings that only occur with an adjunct proto
Studio has no equivalent to -isystem, and no option to its -errhdr that
seems equivalent.  As such, we have to gag the warnings that are induced
from the 3rd party headers when those headers are found via the adjunct,
as they will not match Studio's seemingly hard-coded exception on
'/usr/include'
**********************************************************
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
#
# Copyright 2010 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#

MODULE = libpython2.6.so
MDBTGT = proc

MODSRCS_DIR = ../../../common/modules/libpython2.6

MODSRCS = libpython26.c

include ../../../../Makefile.cmd
include ../../Makefile.ia32
include ../../../Makefile.module

%.o := CPPFLAGS +=      -_gcc=-isystem -_gcc=$(ADJUNCT_PROTO)/usr/include

# Studio provides no equivalent to -isystem, so we have to be heavy handed
CERRWARN +=     -_cc=-erroff=E_MACRO_REDEFINED
LINTFLAGS +=    -erroff=E_MACRO_REDEFINED

#endif /* ! codereview */
dmod/$(MODULE) := LDLIBS += -lproc

%.o: $(MODSRCS_DIR)/%.c
        $(COMPILE.c) $<
        $(CTFCONVERT_O)

%.ln: $(MODSRCS_DIR)/%.c
        $(LINT.c) -c $<
```

```
**********************************************************
    2402 Mon Jan 11 15:36:59 2016
new/usr/src/cmd/rmmount/Makefile
gag Studio warnings that only occur with an adjunct proto
Studio has no equivalent to -isystem, and no option to its -errhdr that
seems equivalent.  As such, we have to gag the warnings that are induced
from the 3rd party headers when those headers are found via the adjunct,
as they will not match Studio's seemingly hard-coded exception on
'/usr/include'
**********************************************************
```

```
#
# 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.
#

PROG =          rmmount
LOCAL_OBJS =    rmmount.o
RMVOLMGR_OBJS = rmm_common.o vold.o
OBJS =          $(LOCAL_OBJS) $(RMVOLMGR_OBJS)
LOCAL_SRCS =    $(LOCAL_OBJS:%.o=%.c)
RMVOLGMR_SRCS = $(RMVOLMGR_OBJS:%.o=$(SRC)/cmd/rmvolmgr/%.c)
SRCS =          $(LOCAL_SRCS) $(RMVOLMGR_SRCS)

include $(SRC)/cmd/Makefile.cmd
include $(SRC)/cmd/hal/Makefile.hal

LDLIBS +=       -ldbus-1 -ldbus-glib-1 -lglib-2.0 -lhal -lhal-storage -lcontract

CPPFLAGS +=     $(HAL_DBUS_CPPFLAGS) $(HAL_GLIB_CPPFLAGS)
CPPFLAGS +=     -I$(ROOT)/usr/include/hal
CPPFLAGS +=     -I$(SRC)/cmd/rmvolmgr
C99MODE =       $(C99_ENABLE)

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

# This is, unfortunately, from the glib headers.  Studio provides no
# equivalent to -isystem
CERRWARN +=     -_cc=-erroff=E_INTEGER_OVERFLOW_DETECTED

#endif /* ! codereview */
.KEEP_STATE:
```



```
all: $(PROG)

$(PROG): $(OBJS)
        $(LINK.c) -o $(PROG) $(OBJS) $(LDLIBS)
        $(POST_PROCESS)

install: all $(ROOTPROG)
        -$(RM) $(ROOT)/usr/bin/rmumount
        -$(RM) $(ROOT)/usr/sbin/rmmount
        -$(SYMLINK) ./rmmount $(ROOT)/usr/bin/rmumount
        -$(SYMLINK) ../bin/rmmount $(ROOT)/usr/sbin/rmmount

rmm_common.o: $(SRC)/cmd/rmvolmgr/rmm_common.c $(SRC)/cmd/rmvolmgr/rmm_common.h
        $(COMPILE.c) -o $@ $(SRC)/cmd/rmvolmgr/rmm_common.c
        $(POST_PROCESS_O)

vold.o: $(SRC)/cmd/rmvolmgr/vold.c $(SRC)/cmd/rmvolmgr/vold.h
        $(COMPILE.c) -o $@ $(SRC)/cmd/rmvolmgr/vold.c
        $(POST_PROCESS_O)

clean:
        $(RM) $(OBJS)

include ../Makefile.targ
```

```
**********************************************************
    2121 Mon Jan 11 15:37:00 2016
new/usr/src/cmd/rmvolmgr/Makefile
gag Studio warnings that only occur with an adjunct proto
Studio has no equivalent to -isystem, and no option to its -errhdr that
seems equivalent.  As such, we have to gag the warnings that are induced
from the 3rd party headers when those headers are found via the adjunct,
as they will not match Studio's seemingly hard-coded exception on
'/usr/include'
**********************************************************
#
# 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.
#

PROG =          rmvolmgr
OBJS =          rmm_common.o rmvolmgr.o vold.o
SRCS =          $(OBJS:%.o=%.c)

MANIFEST =      rmvolmgr.xml
SVCMETHOD =     svc-rmvolmgr

include ../Makefile.cmd
include ../hal/Makefile.hal

POFILE=rmvolmgr_all.po
POFILES=$(OBJS:%.o=%.po)

LDLIBS +=       -ldbus-1 -ldbus-glib-1 -lglib-2.0 -lhal -lhal-storage -lcontract

CPPFLAGS +=     $(HAL_DBUS_CPPFLAGS) $(HAL_GLIB_CPPFLAGS)
CPPFLAGS +=     -I$(ROOT)/usr/include/hal
C99MODE =       $(C99_ENABLE)

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

# This is, unfortunately, from the glib headers.  Studio provides no
# equivalent to -isystem
CERRWARN +=     -_cc=-erroff=E_INTEGER_OVERFLOW_DETECTED

#endif /* ! codereview */
ROOTCMDDIR =            $(ROOTLIB)
```



```
ROOTMANIFESTDIR =       $(ROOTSVCSYSTEMFILESYSTEM)
$(ROOTMANIFEST) :=      FILEMODE = 444
$(ROOTLIBSVCMETHOD)/svc-rmvolmgr:= FILEMODE = 555

.KEEP_STATE:

all: $(PROG)

$(PROG): $(OBJS)
        $(LINK.c) -o $(PROG) $(OBJS) $(LDLIBS)
        $(POST_PROCESS)

install: all $(ROOTCMD) $(ROOTMANIFEST) $(ROOTSVCMETHOD)

clean:
        $(RM) $(OBJS)

check: $(CHKMANIFEST)

$(POFILE): $(POFILES)
        $(RM) $@
        $(CAT) $(POFILES) > $@

include ../Makefile.targ
```

```
**********************************************************
    3228 Mon Jan 11 15:37:00 2016
new/usr/src/cmd/svc/configd/Makefile
native tools must reliably use a native adjunct, even if that's inconvenient
While it is perhaps convenient for native tools to use updated versions
of certain things like libxml it is imperative that those versions are
_also_ build for the build machine.  Thus they need to be in the native
adjunct (even if that native adjunct is thus not /).
Fix the native adjunct to be rooted similarly to the adjunct proto (that
is, at /), and fix SMF to use it correctly
**********************************************************
```

```
#
# 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 2009 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#

MYPROG = svc.configd
MYOBJS = \
        backend.o \
        configd.o \
        client.o \
        file_object.o \
        maindoor.o \
        object.o \
        rc_node.o \
        snapshot.o

PROG = $(MYPROG)
OBJS = $(MYOBJS)

SRCS = $(MYOBJS:%.o=%.c)

include ../../Makefile.cmd
include ../../Makefile.ctf

NATIVE_BUILD=$(POUND_SIGN)
$(NATIVE_BUILD)PROG = $(MYPROG:%=%-native)
$(NATIVE_BUILD)OBJS = $(MYOBJS:%.o=%-native.o)

ROOTCMDDIR=     $(ROOT)/lib/svc/bin

MYCPPFLAGS = -I. -I../common -I../../../common/svc \
        -I$(ROOT)/usr/include/sqlite-sys -D_REENTRANT
CPPFLAGS += $(MYCPPFLAGS)
CFLAGS  += $(CCVERBOSE)
CERRWARN += -_gcc=-Wno-parentheses
```



```
CERRWARN += -_gcc=-Wno-type-limits
CERRWARN += -_gcc=-Wno-unused-label
CERRWARN += -_gcc=-Wno-unused-variable
CERRWARN += -_gcc=-Wno-unused-function
CERRWARN += -_gcc=-Wno-uninitialized
MYLDLIBS = -lumem -luutil -lbsm
LDLIBS  += -lsecdb $(MYLDLIBS)
LINTFLAGS += -errtags -erroff=E_BAD_FORMAT_ARG_TYPE2 -erroff=E_NAME_DEF_NOT_USED

CLOBBERFILES += $(MYPROG:%=%-native)

LIBUUTIL        = $(SRC)/lib/libuutil
LIBSCF          = $(SRC)/lib/libscf

SCRIPTFILE      = restore_repository
ROOTSCRIPTFILE  = $(ROOTCMDDIR)/$(SCRIPTFILE)

#
# Native variant (used in ../seed)
#
$(NATIVE_BUILD)CC =     $(NATIVECC)
$(NATIVE_BUILD)LD =     $(NATIVELD)
$(NATIVE_BUILD)CFLAGS = $(NATIVE_CFLAGS)
$(NATIVE_BUILD)CPPFLAGS = $(MYCPPFLAGS) -I$(LIBUUTIL)/common -I$(LIBSCF)/inc
$(NATIVE_BUILD)CPPFLAGS += -DNATIVE_BUILD
$(NATIVE_BUILD)LDFLAGS =
$(NATIVE_BUILD)LDLIBS = -L$(NATIVE_ADJUNCT)/usr/lib \
                        -R$(NATIVE_ADJUNCT)/usr/lib \
$(NATIVE_BUILD)LDLIBS = -L$(ADJUNCT_PROTO)/usr/lib -R$(ADJUNCT_PROTO)/usr/lib \
                        -L$(LIBUUTIL)/native -R $(LIBUUTIL)/native $(MYLDLIBS)

DIRMODE = 0755
FILEMODE = 0555

OBJSQLITE =
LIBSQLITE = -lsqlite-sys
$(NATIVE_BUILD)OBJSQLITE = $(ROOT)/lib/libsqlite-native.o
$(NATIVE_BUILD)LIBSQLITE =

OBJS += $(OBJSQLITE)
LDLIBS += $(LIBSQLITE)

install := TARGET = install
clobber := TARGET = clobber

.KEEP_STATE:
.PARALLEL: $(MYOBJS) $(MYOBJS:%.o=%-native.o)

all: $(PROG)

native: FRC
        @cd $(LIBUUTIL)/native; pwd; $(MAKE) $(MFLAGS) install
        @NATIVE_BUILD= $(MAKE) $(MFLAGS) all

$(PROG): $(OBJS)
        $(LINK.c) -o $@ $(OBJS) $(LDLIBS)
        $(POST_PROCESS)

%-native.o: %.c
        $(COMPILE.c) -o $@ $<
        $(POST_PROCESS_O)

$(ROOTCMDDIR)/%: %.sh
        $(INS.rename)

install: all $(ROOTCMD) $(ROOTVARSADMFILE) $(ROOTSCRIPTFILE)
```

```
122 clean: FRC
123         $(RM) $(MYOBJS) $(MYOBJS:%.o=%-native.o)

125 clobber:

127 lint:   lint_SRCS

129 lint_SRCS:

131 include ../../Makefile.targ

133 FRC:
```

```
**********************************************************
    4408 Mon Jan 11 15:37:01 2016
new/usr/src/cmd/svc/svccfg/Makefile
native tools must reliably use a native adjunct, even if that's inconvenient
While it is perhaps convenient for native tools to use updated versions
of certain things like libxml it is imperative that those versions are
_also_ build for the build machine.  Thus they need to be in the native
adjunct (even if that native adjunct is thus not /).
Fix the native adjunct to be rooted similarly to the adjunct proto (that
is, at /), and fix SMF to use it correctly
**********************************************************
#
# 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) 2004, 2010, Oracle and/or its affiliates. All rights reserved.
#

MYPROG =        svccfg
PROG =          $(MYPROG)

SRCS  =         svccfg_main.c \
                svccfg_engine.c \
                svccfg_internal.c \
                svccfg_libscf.c \
                svccfg_tmpl.c \
                svccfg_xml.c \
                svccfg_help.c

LNTS =          $(SRCS:%.c=%.ln) \
                manifest_find.ln \
                manifest_hash.ln

MYOBJS =        $(SRCS:%.c=%.o) \
                svccfg_grammar.o \
                svccfg_lex.o \
                manifest_find.o \
                manifest_hash.o \
                notify_params.o
OBJS =          $(MYOBJS)

POFILES =       $(SRCS:%.c=%.po) \
                svccfg_grammar.po \
                svccfg_lex.po \
                ../common/manifest_find.po \
                ../common/manifest_hash.po

include ../../Makefile.cmd
```



```
include ../../Makefile.ctf

POFILE =        $(PROG)_all.po

NATIVE_BUILD=$(POUND_SIGN)
$(NATIVE_BUILD)NOT_NATIVE=$(POUND_SIGN)

$(NATIVE_BUILD)PROG = $(MYPROG:%=%-native)
$(NATIVE_BUILD)OBJS = $(MYOBJS:%.o=%-native.o)

# svccfg has a name clash with main() and libl.so.1.  However, svccfg must
# still export a number of "yy*" (libl) interfaces.  Reduce all other symbols
# to local scope.
MAPFILES +=     $(MAPFILE.LEX) $(MAPFILE.NGB)
MAPOPTS =       $(MAPFILES:%=-M%)

MYCPPFLAGS =    -I ../common -I$(ADJUNCT_PROTO)/usr/include/libxml2
$(NATIVE_BUILD)MYCPPFLAGS =     -I ../common \
                -I$(NATIVE_ADJUNCT)/usr/include/libxml2
#endif /* ! codereview */
CPPFLAGS +=     $(MYCPPFLAGS)
LDFLAGS +=      $(MAPOPTS)

CERRWARN +=     -_gcc=-Wno-unused-label
CERRWARN +=     -_gcc=-Wno-implicit-function-declaration
CERRWARN +=     -_gcc=-Wno-switch
CERRWARN +=     -_gcc=-Wno-uninitialized
CERRWARN +=     -_gcc=-Wno-unused-variable
CERRWARN +=     -_gcc=-Wno-parentheses

LFLAGS = -t
YFLAGS = -d

CLOBBERFILES += svccfg_lex.c svccfg_grammar.c svccfg_grammar.h \
    $(MYPROG:%=%-native)

SVCCFG_EXTRA_LIBS = -lxml2 -lscf -ll -luutil -lumem -lmd5 -lnvpair
$(NOT_NATIVE)SVCCFG_EXTRA_LIBS += -ltecla

LIBSCF          = $(SRC)/lib/libscf
LIBTECLA        = $(SRC)/lib/libtecla           # just for the header
LIBUUTIL        = $(SRC)/lib/libuutil

debug := COPTFLAG = -g

lint := LINTFLAGS = -mux
lint := SVCCFG_EXTRA_LIBS = -lscf -ll -luutil -lumem -lmd5 -lnvpair

LDLIBS += $(SVCCFG_EXTRA_LIBS)

$(NATIVE_BUILD)CC =     $(NATIVECC)
$(NATIVE_BUILD)LD =     $(NATIVELD)
$(NATIVE_BUILD)CFLAGS = $(NATIVE_CFLAGS)
$(NATIVE_BUILD)CPPFLAGS = \
        -DNATIVE_BUILD \
        $(MYCPPFLAGS) \
        -I$(LIBSCF)/inc \
        -I$(LIBTECLA) \
        -I$(LIBUUTIL)/common
$(NATIVE_BUILD)LDFLAGS =
$(NATIVE_BUILD)LDLIBS = \
        -L$(LIBUUTIL)/native -R $(LIBUUTIL)/native \
        -L$(LIBSCF)/native -R $(LIBSCF)/native \
        -L$(NATIVE_ADJUNCT)/usr/lib -R$(NATIVE_ADJUNCT)/usr/lib \
        -L$(ADJUNCT_PROTO)/usr/lib -R$(ADJUNCT_PROTO)/usr/lib \
        $(SVCCFG_EXTRA_LIBS)
```

```
122 svccfg_lex.o svccfg_grammar.o := CCVERBOSE =

124 svccfg_help.po := XGETFLAGS =   -a

126 .KEEP_STATE:
127 .PARALLEL: $(OBJS) $(LNTS)

129 all debug: $(PROG)

131 native: FRC
132         @cd $(LIBUUTIL)/native; pwd; $(MAKE) $(MFLAGS) install
133         @cd $(LIBSCF)/native; pwd; $(MAKE) $(MFLAGS) install
134         @NATIVE_BUILD= $(MAKE) $(MFLAGS) all

136 $(PROG): $(OBJS) $(MAPFILES)
137         $(LINK.c) -o $@ $(OBJS) $(LDLIBS)
138         $(POST_PROCESS)

140 $(POFILE): $(POFILES)
141         cat $(POFILES) > $(POFILE)

143 install: all $(ROOTUSRSBINPROG)

145 svccfg_lex.c: svccfg.l svccfg_grammar.h
146         $(LEX) $(LFLAGS) svccfg.l > $@

148 svccfg_help.o: svccfg_grammar.h
149 svccfg_help-native.o: svccfg_grammar.h

151 svccfg_grammar.h svccfg_grammar.c: svccfg.y
152         $(YACC) $(YFLAGS) svccfg.y
153         @$(MV) y.tab.h svccfg_grammar.h
154         @$(MV) y.tab.c svccfg_grammar.c

156 clean: FRC
157         $(RM) $(MYOBJS) $(MYOBJS:%.o=%-native.o) $(LNTS)

159 lint: $(LNTS)
160         $(LINT.c) $(LINTFLAGS) $(LNTS) $(LDLIBS)

162 %-native.o: %.c
163         $(COMPILE.c) -o $@ $<
164         $(POST_PROCESS_O)

166 %-native.o: ../common/%.c
167         $(COMPILE.c) -o $@ $<
168         $(POST_PROCESS_O)

170 %.o: ../common/%.c
171         $(COMPILE.c) $(OUTPUT_OPTION) $<
172         $(POST_PROCESS_O)

174 %.ln: ../common/%.c
175         $(LINT.c) $(OUTPUT_OPTION) -c $<

177 include ../../Makefile.targ

179 FRC:
```

```
**********************************************************
    2241 Mon Jan 11 15:37:01 2016
new/usr/src/cmd/volrmmount/Makefile
gag Studio warnings that only occur with an adjunct proto
Studio has no equivalent to -isystem, and no option to its -errhdr that
seems equivalent.  As such, we have to gag the warnings that are induced
from the 3rd party headers when those headers are found via the adjunct,
as they will not match Studio's seemingly hard-coded exception on
'/usr/include'
**********************************************************
#
# 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.
#

PROG =          volrmmount
LOCAL_OBJS =    volrmmount.o
RMVOLMGR_OBJS = rmm_common.o vold.o
OBJS =          $(LOCAL_OBJS) $(RMVOLMGR_OBJS)
LOCAL_SRCS =    $(LOCAL_OBJS:%.o=%.c)
RMVOLGMR_SRCS = $(RMVOLMGR_OBJS:%.o=$(SRC)/cmd/rmvolmgr/%.c)
SRCS =          $(LOCAL_SRCS) $(RMVOLMGR_SRCS)

include $(SRC)/cmd/Makefile.cmd
include $(SRC)/cmd/hal/Makefile.hal

LDLIBS +=       -ldbus-1 -ldbus-glib-1 -lglib-2.0 -lhal -lhal-storage -lcontract

CPPFLAGS +=     $(HAL_DBUS_CPPFLAGS) $(HAL_GLIB_CPPFLAGS)
CPPFLAGS +=     -I$(ROOT)/usr/include/hal
CPPFLAGS +=     -I$(SRC)/cmd/rmvolmgr
C99MODE =       $(C99_ENABLE)

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

# This is, unfortunately, from the glib headers.  Studio provides no
# equivalent to -isystem
CERRWARN +=     -_cc=-erroff=E_INTEGER_OVERFLOW_DETECTED

#endif /* ! codereview */
.KEEP_STATE:
```



```
all: $(PROG)

$(PROG): $(OBJS)
        $(LINK.c) -o $(PROG) $(OBJS) $(LDLIBS)
        $(POST_PROCESS)

install: all $(ROOTPROG)

rmm_common.o: $(SRC)/cmd/rmvolmgr/rmm_common.c $(SRC)/cmd/rmvolmgr/rmm_common.h
        $(COMPILE.c) -o $@ $(SRC)/cmd/rmvolmgr/rmm_common.c
        $(POST_PROCESS_O)

vold.o: $(SRC)/cmd/rmvolmgr/vold.c $(SRC)/cmd/rmvolmgr/vold.h
        $(COMPILE.c) -o $@ $(SRC)/cmd/rmvolmgr/vold.c
        $(POST_PROCESS_O)

clean:
        $(RM) $(OBJS)

include ../Makefile.targ
```

```
**********************************************************
    2285 Mon Jan 11 15:37:02 2016
new/usr/src/lib/libscf/Makefile.com
native tools must reliably use a native adjunct, even if that's inconvenient
While it is perhaps convenient for native tools to use updated versions
of certain things like libxml it is imperative that those versions are
_also_ build for the build machine.  Thus they need to be in the native
adjunct (even if that native adjunct is thus not /).
Fix the native adjunct to be rooted similarly to the adjunct proto (that
is, at /), and fix SMF to use it correctly
**********************************************************
#
# 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) 2004, 2010, Oracle and/or its affiliates. All rights reserved.
#

LIBRARY =       libscf.a
VERS =          .1

OBJECTS = \
        error.o                 \
        lowlevel.o              \
        midlevel.o              \
        notify_params.o         \
        highlevel.o             \
        scf_tmpl.o              \
        scf_type.o

include ../../Makefile.lib
include ../../Makefile.rootfs

LIBS =          $(DYNLIB) $(LINTLIB)

$(NOT_NATIVE)NATIVE_BUILD = $(POUND_SIGN)
$(NATIVE_BUILD)VERS =
$(NATIVE_BUILD)LIBS = $(DYNLIB)

LDLIBS_i386 += -lsmbios
LDLIBS +=       -luutil -lc -lgen -lnsl -lnvpair
LDLIBS +=       $(LDLIBS_$(MACH))

SRCDIR =        ../common
$(LINTLIB) :=   SRCS = $(SRCDIR)/$(LINTSRC)

COMDIR =        ../../../common/svc

CFLAGS +=       $(CCVERBOSE) -Wp,-xc99=%all
```



```
CPPFLAGS +=     -I../inc -I../../common/inc -I$(COMDIR) -I$(ROOTHDRDIR)
$(NOT_RELEASE_BUILD) CPPFLAGS += -DFASTREBOOT_DEBUG

CERRWARN +=     -_gcc=-Wno-switch
CERRWARN +=     -_gcc=-Wno-char-subscripts
CERRWARN +=     -_gcc=-Wno-unused-label
CERRWARN +=     -_gcc=-Wno-parentheses
CERRWARN +=     -_gcc=-Wno-uninitialized

#
# For native builds, we compile and link against the native version
# of libuutil.
#
LIBUUTIL =      $(SRC)/lib/libuutil
MY_NATIVE_CPPFLAGS =\
                -DNATIVE_BUILD $(DTEXTDOM) \
                -I../inc -I$(COMDIR) -I$(LIBUUTIL)/common -I$(ROOTHDRDIR)
MY_NATIVE_LDLIBS = -L$(LIBUUTIL)/native -R$(LIBUUTIL)/native \
                   -L$(NATIVE_ADJUNCT)/usr/lib -R$(NATIVE_ADJUNCT)/usr/lib \
                   -luutil -lc -lgen -lnsl -lnvpair
MY_NATIVE_LDLIBS = -L$(LIBUUTIL)/native -R$(LIBUUTIL)/native -luutil -lc -lgen \
                   -lnsl -lnvpair
MY_NATIVE_LDLIBS_i386 = -lsmbios
MY_NATIVE_LDLIBS += $(MY_NATIVE_LDLIBS_$(MACH))

.KEEP_STATE:

all:

lint: lintcheck

include ../../Makefile.targ
```

```
**********************************************************
    1675 Mon Jan 11 15:37:02 2016
new/usr/src/lib/policykit/Makefile.com
gag Studio warnings that only occur with an adjunct proto
Studio has no equivalent to -isystem, and no option to its -errhdr that
seems equivalent.  As such, we have to gag the warnings that are induced
from the 3rd party headers when those headers are found via the adjunct,
as they will not match Studio's seemingly hard-coded exception on
'/usr/include'
**********************************************************
#
# 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.
#

include $(SRC)/lib/Makefile.lib
include $(SRC)/lib/policykit/Makefile.policykit

CPPFLAGS =      $(POLICYKIT_DBUS_CPPFLAGS) $(POLICYKIT_GLIB_CPPFLAGS) $(CPPFLAGS

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

# This is, unfortunately, from the glib headers.  Studio provides no
# equivalent to -isystem
CERRWARN +=     -_cc=-erroff=E_INTEGER_OVERFLOW_DETECTED

#endif /* ! codereview */
C99MODE =       $(C99_ENABLE)

ROOTLIBPCDIR =  $(ROOT)/usr/lib/pkgconfig
ROOTLIBPC =     $(LIBPCSRC:%=$(ROOTLIBPCDIR)/%)

CLOBBERFILES += $(LIBPCSRC)

#
# Ensure 'all' is the default target.
#
all:

$(ROOTLIBPCDIR):
        $(INS.dir)

$(ROOTLIBPC): $(ROOTLIBPCDIR) $(LIBPCSRC)
        $(INS.file) $(LIBPCSRC)

$(LIBPCSRC): ../common/$(LIBPCSRC).in
```



```
        $(SED)  -e "s@__VERSION__@$(POLICYKIT_VERSION)@" \
                < ../common/$(LIBPCSRC).in > $(LIBPCSRC)
```

```
**********************************************************
   25412 Mon Jan 11 15:37:02 2016
new/usr/src/pkg/Makefile
pkg: If the adjunct proto is a package image, use it for dependency resolution
**********************************************************
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#

#
# Copyright (c) 2010, Oracle and/or its affiliates. All rights reserved.
# Copyright 2015, OmniTI Computer Consulting, Inc. All rights reserved.
# Copyright 2015 Igor Kozhukhov <ikozhukhov@gmail.com>
#

include $(SRC)/Makefile.master
include $(SRC)/Makefile.buildnum

#
# Make sure we're getting a consistent execution environment for the
# embedded scripts.
#
SHELL= /usr/bin/ksh93

#
# To suppress package dependency generation on any system, regardless
# of how it was installed, set SUPPRESSPKGDEP=true in the build
# environment.
#
SUPPRESSPKGDEP= false

#
# Comment this line out or set "PKGDEBUG=" in your build environment
# to get more verbose output from the make processes in usr/src/pkg
#
PKGDEBUG= @

#
# Cross platform packaging notes
#
# By default, we package the proto area from the same architecture as
# the packaging build.  In other words, if you're running nightly or
# bldenv on an x86 platform, it will take objects from the x86 proto
# area and use them to create x86 repositories.
#
# If you want to create repositories for an architecture that's
# different from $(uname -p), you do so by setting PKGMACH in your
# build environment.
#
```



```
# For this to work correctly, the following must all happen:
#
#   1. You need the desired proto area, which you can get either by
#      doing a gatekeeper-style build with the -U option to
#      nightly(1), or by using rsync.  If you don't do this, you will
#      get packaging failures building all packages, because pkgsend
#      is unable to find the required binaries.
#   2. You need the desired tools proto area, which you can get in the
#      same ways as the normal proto area.  If you don't do this, you
#      will get packaging failures building onbld, because pkgsend is
#      unable to find the tools binaries.
#   3. The remainder of this Makefile should never refer directly to
#      $(MACH).  Instead, $(PKGMACH) should be used whenever an
#      architecture-specific path or token is needed.  If this is done
#      incorrectly, then packaging will fail, and you will see the
#      value of $(uname -p) instead of the value of $(PKGMACH) in the
#      commands that fail.
#   4. Each time a rule in this Makefile invokes $(MAKE), it should
#      pass PKGMACH=$(PKGMACH) explicitly on the command line.  If
#      this is done incorrectly, then packaging will fail, and you
#      will see the value of $(uname -p) instead of the value of
#      $(PKGMACH) in the commands that fail.
#
# Refer also to the convenience targets defined later in this
# Makefile.
#
PKGMACH=        $(MACH)

#
# ROOT, TOOLS_PROTO, and PKGARCHIVE should be set by nightly or
# bldenv.  These macros translate them into terms of $PKGMACH, instead
# of $ARCH.
#
PKGROOT.cmd=    print $(ROOT) | sed -e s:/root_$(MACH):/root_$(PKGMACH):
PKGROOT=        $(PKGROOT.cmd:sh)
TOOLSROOT.cmd=  print $(TOOLS_PROTO) | sed -e s:/root_$(MACH):/root_$(PKGMACH):
TOOLSROOT=      $(TOOLSROOT.cmd:sh)
PKGDEST.cmd=    print $(PKGARCHIVE) | sed -e s:/$(MACH)/:/$(PKGMACH)/:
PKGDEST=        $(PKGDEST.cmd:sh)

EXCEPTIONS= packaging

PKGMOGRIFY= pkgmogrify

#
# Always build the redistributable repository, but only build the
# nonredistributable bits if we have access to closed source.
#
# Some objects that result from the closed build are still
# redistributable, and should be packaged as part of an open-only
# build.  Access to those objects is provided via the closed-bins
# tarball.  See usr/src/tools/scripts/bindrop.sh for details.
#
REPOS= redist

#
# The packages directory will contain the processed manifests as
# direct build targets and subdirectories for package metadata extracted
# incidentally during manifest processing.
#
# Nothing underneath $(PDIR) should ever be managed by SCM.
#
PDIR= packages.$(PKGMACH)

#
# The tools proto must be specified for dependency generation.
```

```
128 # Publication from the tools proto area is managed in the
129 # publication rule.
130 #
131 $(PDIR)/developer-build-onbld.dep:= PKGROOT= $(TOOLSROOT)

133 PKGPUBLISHER= $(PKGPUBLISHER_REDIST)

135 #
136 # To get these defaults, manifests should simply refer to $(PKGVERS).
137 #
138 PKGVERS_COMPONENT= 0.$(RELEASE)
139 PKGVERS_BUILTON= $(RELEASE)
140 PKGVERS_BRANCH= 0.$(ONNV_BUILDNUM)
141 PKGVERS= $(PKGVERS_COMPONENT),$(PKGVERS_BUILTON)-$(PKGVERS_BRANCH)

143 #
144 # The ARCH32 and ARCH64 macros are used in the manifests to express
145 # architecture-specific subdirectories in the installation paths
146 # for isaexec'd commands.
147 #
148 # We can't simply use $(MACH32) and $(MACH64) here, because they're
149 # only defined for the build architecture.  To do cross-platform
150 # packaging, we need both values.
151 #
152 i386_ARCH32= i86
153 sparc_ARCH32= sparcv7
154 i386_ARCH64= amd64
155 sparc_ARCH64= sparcv9

157 #
158 # macros and transforms needed by pkgmogrify
159 #
160 # If you append to this list using target-specific assignments (:=),
161 # be very careful that the targets are of the form $(PDIR)/pkgname.  If
162 # you use a higher level target, or a package list, you'll trigger a
163 # complete reprocessing of all manifests because they'll fail command
164 # dependency checking.
165 #
166 PM_TRANSFORMS= common_actions publish restart_fmri facets defaults \
167         extract_metadata
168 PM_INC= transforms manifests

170 PKGMOG_DEFINES= \
171         i386_ONLY=$(POUND_SIGN) \
172         sparc_ONLY=$(POUND_SIGN) \
173         $(PKGMACH)_ONLY= \
174         ARCH=$(PKGMACH) \
175         ARCH32=$($(PKGMACH)_ARCH32) \
176         ARCH64=$($(PKGMACH)_ARCH64) \
177         PKGVERS_COMPONENT=$(PKGVERS_COMPONENT) \
178         PKGVERS_BUILTON=$(PKGVERS_BUILTON) \
179         PKGVERS_BRANCH=$(PKGVERS_BRANCH) \
180         PKGVERS=$(PKGVERS) \
181         PERL_ARCH=$(PERL_ARCH) \
182         PERL_VERSION=$(PERL_VERSION) \
183         PERL_PKGVERS=$(PERL_PKGVERS)

185 PKGDEP_TOKENS_i386= \
186         'PLATFORM=i86hvm' \
187         'PLATFORM=i86pc' \
188         'PLATFORM=i86xpv' \
189         'ISALIST=amd64' \
190         'ISALIST=i386'
191 PKGDEP_TOKENS_sparc= \
192         'PLATFORM=sun4u' \
193         'PLATFORM=sun4v' \
```



```
194         'ISALIST=sparcv9' \
195         'ISALIST=sparc'
196 PKGDEP_TOKENS= $(PKGDEP_TOKENS_$(PKGMACH))

198 #
199 # The package lists are generated with $(PKGDEP_TYPE) as their
200 # dependency types, so that they can be included by either an
201 # incorporation or a group package.
202 #
203 $(PDIR)/osnet-redist.mog := PKGDEP_TYPE= require
204 $(PDIR)/osnet-incorporation.mog:= PKGDEP_TYPE= incorporate

206 PKGDEP_INCORP= \
207         depend fmri=consolidation/osnet/osnet-incorporation type=require

209 #
210 # All packaging build products should go into $(PDIR), so they don't
211 # need to be included separately in CLOBBERFILES.
212 #
213 CLOBBERFILES= $(PDIR) proto_list_$(PKGMACH) install-$(PKGMACH).out \
214         license-list

216 #
217 # By default, PKGS will list all manifests.  To build and/or publish a
218 # subset of packages, override this on the command line or in the
219 # build environment and then reference (implicitly or explicitly) the all
220 # or install targets.  Using ls -1 (that's a one) or print or echo to
221 # get the list of manifests is a little hackish, but avoids having a
222 # 900+ line file to explicitly list them all.
223 #
224 # We want some manifests to optionally built based on environment
225 # options, so those are excluded and optionally added back in.
226 # We also want a relatively easy way to add files to the list of
227 # manifests given special treatment.  Add any other special ones
228 # to the SPECIAL_MANIFESTS vaiable.  It can contain wildcards in
229 # regexp form, i.e. SUNW.* as one useful example.
230 #
231 SPECIAL_MANIFESTS = print-lp-ipp-ipp-listener.mf
232 LIST_MANIFESTS_CMD = (cd manifests ; /usr/bin/ls -1 *.mf |\
233         $(SED) $(SPECIAL_MANIFESTS:%=-e '/^%$$/d') )
234 MANIFESTS = $(LIST_MANIFESTS_CMD:sh)

236 # Conditionally add back lp-ipp
237 $(ENABLE_IPP_PRINTING) MANIFESTS += print-lp-ipp-ipp-listener.mf

239 PKGS= $(MANIFESTS:%.mf=%)
240 DEP_PKGS= $(PKGS:%=$(PDIR)/%.dep)
241 PROC_PKGS= $(PKGS:%=$(PDIR)/%.mog)

243 #
244 # Track the synthetic manifests separately so we can properly express
245 # build rules and dependencies.  The synthetic and real packages use
246 # different sets of transforms and macros for pkgmogrify.
247 #
248 SYNTH_PKGS= osnet-incorporation osnet-redist
249 DEP_SYNTH_PKGS= $(SYNTH_PKGS:%=$(PDIR)/%.dep)
250 PROC_SYNTH_PKGS= $(SYNTH_PKGS:%=$(PDIR)/%.mog)

252 #
253 # Root of pkg image to use for dependency resolution Normally / on the machine
254 # used to build the binaries, or the ADJUNCT_PROTO
253 # Root of pkg image to use for dependency resolution
254 # Normally / on the machine used to build the binaries
255 #
256 PKG_ADJUNCT.cmd= if [[ -d $${ADJUNCT_PROTO}/var/pkg ]]; then \
257                 echo $$ADJUNCT_PROTO; \
```

```
258         else    \
259                 echo /; \
260         fi

262 PKGDEP_RESOLVE_IMAGE = $(PKG_ADJUNCT.cmd:sh)
256 PKGDEP_RESOLVE_IMAGE = /

264 #
265 # For each package, we determine the target repository based on
266 # manifest-embedded metadata.  Because we make that determination on
267 # the fly, the publication target cannot be expressed as a
268 # subdirectory inside the unknown-by-the-makefile target repository.
269 #
270 # In order to limit the target set to real files in known locations,
271 # we use a ".pub" file in $(PDIR) for each processed manifest, regardless
272 # of content or target repository.
273 #
274 PUB_PKGS= $(SYNTH_PKGS:%=$(PDIR)/%.pub) $(PKGS:%=$(PDIR)/%.pub)

276 #
277 # Any given repository- and status-specific package list may be empty,
278 # but we can only determine that dynamically, so we always generate all
279 # lists for each repository we're building.
280 #
281 # The meanings of each package status are as follows:
282 #
283 #       PKGSTAT         meaning
284 #       ----------      ------------------------------------------------------
285 #       noincorp        Do not include in incorporation or group package
286 #       obsolete        Include in incorporation, but not group package
287 #       renamed         Include in incorporation, but not group package
288 #       current         Include in incorporation and group package
289 #
290 # Since the semantics of the "noincorp" package status dictate that
291 # such packages are not included in the incorporation or group packages,
292 # there is no need to build noincorp package lists.
293 #
294 PKGLISTS= \
295         $(REPOS:%=$(PDIR)/packages.%.current) \
296         $(REPOS:%=$(PDIR)/packages.%.renamed) \
297         $(REPOS:%=$(PDIR)/packages.%.obsolete)

299 .KEEP_STATE:

301 .PARALLEL: $(PKGS) $(PROC_PKGS) $(DEP_PKGS) \
302         $(PROC_SYNTH_PKGS) $(DEP_SYNTH_PKGS) $(PUB_PKGS)

304 #
305 # For a single manifest, the dependency chain looks like this:
306 #
307 #       raw manifest (mypkg.mf)
308 #               |
309 #               | use pkgmogrify to process raw manifest
310 #               |
311 #       processed manifest (mypkg.mog)
312 #               |
313 #          *    | use pkgdepend generate to generate dependencies
314 #               |
315 #       manifest with TBD dependencies (mypkg.dep)
316 #               |
317 #          %    | use pkgdepend resolve to resolve dependencies
318 #               |
319 #       manifest with dependencies resolved (mypkg.res)
320 #               |
321 #               | use pkgsend to publish the package
322 #               |
```



```
323 #       placeholder to indicate successful publication (mypkg.pub)
324 #
325 # * This may be suppressed via SUPPRESSPKGDEP.  The resulting
326 #   packages will install correctly, but care must be taken to
327 #   install all dependencies, because pkg will not have the input
328 #   it needs to determine this automatically.
329 #
330 # % This is included in this diagram to make the picture complete, but
331 #   this is a point of synchronization in the build process.
332 #   Dependency resolution is actually done once on the entire set of
333 #   manifests, not on a per-package basis.
334 #
335 # The full dependency chain for generating everything that needs to be
336 # published, without actually publishing it, looks like this:
337 #
338 #       processed synthetic packages
339 #                |               |
340 #       package lists       synthetic package manifests
341 #                |
342 #       processed real packages
343 #           |            |
344 #       package dir     real package manifests
345 #
346 # Here, each item is a set of real or synthetic packages.  For this
347 # portion of the build, no reference is made to the proto area.  It is
348 # therefore suitable for the "all" target, as opposed to "install."
349 #
350 # Since each of these steps is expressed explicitly, "all" need only
351 # depend on the head of the chain.
352 #
353 # From the end of manifest processing, the publication dependency
354 # chain looks like this:
355 #
356 #               repository metadata (catalogs and search indices)
357 #                       |
358 #                       | pkgrepo refresh
359 #                       |
360 #               published packages
361 #                 |               |
362 #                 |               | pkgsend publish
363 #                 |               |
364 #          repositories      resolved dependencies
365 #                 |                       |
366 # pkgsend         |                       | pkgdepend resolve
367 # create-repository                       |
368 #                 |               generated dependencies
369 #        repo directories                 |
370 #                                         | pkgdepend
371 #                                         |
372 #                                 processed manifests
373 #

375 ALL_TARGETS= $(PROC_SYNTH_PKGS) proto_list_$(PKGMACH)

377 all: $(ALL_TARGETS)

379 #
380 # This will build the directory to contain the processed manifests
381 # and the metadata symlinks.
382 #
383 $(PDIR):
384         @print "Creating $(@)"
385         $(PKGDEBUG)$(INS.dir)

387 #
388 # This rule resolves dependencies across all published manifests.
```

```
389 #
390 # We shouldn't have to ignore the error from pkgdepend, but until
391 # 16012 and its dependencies are resolved, pkgdepend will always exit
392 # with an error.
393 #
394 $(PDIR)/gendeps: $(DEP_SYNTH_PKGS) $(DEP_PKGS)
395         -$(PKGDEBUG)if [ "$(SUPPRESSPKGDEP)" = "true" ]; then \
396                 print "Suppressing dependency resolution"; \
397                 for p in $(DEP_PKGS:%.dep=%); do \
398                         $(CP) $$p.dep $$p.res; \
399                 done; \
400         else \
401                 print "Resolving dependencies"; \
402                 pkgdepend -R $(PKGDEP_RESOLVE_IMAGE) resolve \
403                     -m $(DEP_SYNTH_PKGS) $(DEP_PKGS); \
404                 for p in $(DEP_SYNTH_PKGS:%.dep=%) $(DEP_PKGS:%.dep=%); do \
405                         if [ "$$(print $$p.metadata.*)" = \
406                             "$$(print $$p.metadata.noincorp.*)" ]; \
407                         then \
408                                 print "Removing dependency versions from $$p"; \
409                                 $(PKGMOGRIFY) $(PKGMOG_VERBOSE) \
410                                     -O $$p.res -I transforms \
411                                     strip_versions $$p.dep.res; \
412                                 $(RM) $$p.dep.res; \
413                         else \
414                                 $(MV) $$p.dep.res $$p.res; \
415                         fi; \
416                 done; \
417         fi
418         $(PKGDEBUG)$(TOUCH) $(@)

420 install: $(ALL_TARGETS) repository-metadata

422 repository-metadata: publish_pkgs
423         $(PKGDEBUG)for r in $(REPOS); do \
424                 pkgrepo refresh -s $(PKGDEST)/repo.$$r; \
425         done

427 #
428 # Since we create zero-length processed manifests for a graceful abort
429 # from pkgmogrify, we need to detect that here and make no effort to
430 # publish the package.
431 #
432 # For all other packages, we publish them regardless of status.  We
433 # derive the target repository as a component of the metadata-derived
434 # symlink for each package.
435 #
436 publish_pkgs: $(REPOS:%=$(PKGDEST)/repo.%) $(PDIR)/gendeps .WAIT $(PUB_PKGS)

438 #
439 # Before publishing, we want to pull the license files from $CODEMGR_WS
440 # into the proto area.  This allows us to NOT pass $SRC (or
441 # $CODEMGR_WS) as a basedir for publication.
442 #
443 $(PUB_PKGS): stage-licenses

445 #
446 # Initialize the empty on-disk repositories
447 #
448 $(REPOS:%=$(PKGDEST)/repo.%):
449         @print "Initializing $(@F)"
450         $(PKGDEBUG)$(INS.dir)
451         $(PKGDEBUG)pkgsend -s file://$(@) create-repository \
452                 --set-property publisher.prefix=$(PKGPUBLISHER)

454 #
```



```
455 # rule to process real manifests
456 #
457 # To allow redistributability and package status to change, we must
458 # remove not only the actual build target (the processed manifest), but
459 # also the incidental ones (the metadata-derived symlinks).
460 #
461 # If pkgmogrify exits cleanly but fails to create the specified output
462 # file, it means that it encountered an abort directive.  That means
463 # that this package should not be published for this particular build
464 # environment.  Since we can't prune such packages from $(PKGS)
465 # retroactively, we need to create an empty target file to keep make
466 # from trying to rebuild it every time.  For these empty targets, we
467 # do not create metadata symlinks.
468 #
469 # Automatic dependency resolution to files is also done at this phase of
470 # processing.  The skipped packages are skipped due to existing bugs
471 # in pkgdepend.
472 #
473 # The incorporation dependency is tricky: it needs to go into all
474 # current and renamed manifests (ie all incorporated packages), but we
475 # don't know which those are until after we run pkgmogrify.  So
476 # instead of expressing it as a transform, we tack it on ex post facto.
477 #
478 # Implementation notes:
479 #
480 # - The first $(RM) must not match other manifests, or we'll run into
481 #   race conditions with parallel manifest processing.
482 #
483 # - The make macros [ie $(MACRO)] are evaluated when the makefile is
484 #   read in, and will result in a fixed, macro-expanded rule for each
485 #   target enumerated in $(PROC_PKGS).
486 #
487 # - The shell variables (ie $$VAR) are assigned on the fly, as the rule
488 #   is executed.  The results may only be referenced in the shell in
489 #   which they are assigned, so from the perspective of make, all code
490 #   that needs these variables needs to be part of the same line of
491 #   code.  Hence the use of command separators and line continuation
492 #   characters.
493 #
494 # - The extract_metadata transforms are designed to spit out shell
495 #   variable assignments to stdout.  Those are published to the
496 #   .vars temporary files, and then used as input to the eval
497 #   statement.  This is done in stages specifically so that pkgmogrify
498 #   can signal failure if the manifest has a syntactic or other error.
499 #   The eval statement should begin with the default values, and the
500 #   output from pkgmogrify (if any) should be in the form of a
501 #   variable assignment to override those defaults.
502 #
503 # - When this rule completes execution, it must leave an updated
504 #   target file ($@) in place, or make will reprocess the package
505 #   every time it encounters it as a dependency.  Hence the "touch"
506 #   statement to ensure that the target is created, even when
507 #   pkgmogrify encounters an abort in the publish transforms.
508 #

510 .SUFFIXES: .mf .mog .dep .res .pub

512 $(PDIR)/%.mog: manifests/%.mf
513         @print "Processing manifest $(<F)"
514         @env PKGFMT_OUTPUT=v1 pkgfmt -c $<
515         $(PKGDEBUG)$(RM) $(@) $(@:%.mog=%) $(@:%.mog=%.nodepend) \
516                 $(@:%.mog=%.lics) $(PDIR)/$(@F:%.mog=%).metadata.* $(@).vars
517         $(PKGDEBUG)$(PKGMOGRIFY) $(PKGMOG_VERBOSE) $(PM_INC:%= -I %) \
518                 $(PKGMOG_DEFINES:%=-D %) -P $(@).vars -O $(@) \
519                 $(<) $(PM_TRANSFORMS)
520         $(PKGDEBUG)eval REPO=redist PKGSTAT=current NODEPEND=$(SUPPRESSPKGDEP) \
```

```
521                 '$(CAT) -s $(@).vars'; \
522         if [ -f $(@) ]; then \
523                 if [ "$$NODEPEND" != "false" ]; then \
524                         $(TOUCH) $(@:%.mog=%.nodepend); \
525                 fi; \
526                 $(LN) -s $(@F) \
527                         $(PDIR)/$(@F:%.mog=%).metadata.$$PKGSTAT.$$REPO; \
528                 if [ \( "$$PKGSTAT" = "current" \) -o \
529                      \( "$$PKGSTAT" = "renamed" \) ]; \
530                         then print $(PKGDEP_INCORP) >> $(@); \
531                 fi; \
532                 print $$LICS > $(@:%.mog=%.lics); \
533         else \
534                 $(TOUCH) $(@) $(@:%.mog=%.lics); \
535         fi
536         $(PKGDEBUG)$(RM) $(@).vars

538 $(PDIR)/%.dep: $(PDIR)/%.mog
539         @print "Generating dependencies for $(<F)"
540         $(PKGDEBUG)$(RM) $(@)
541         $(PKGDEBUG)if [ ! -f $(@:%.dep=%.nodepend) ]; then \
542                 pkgdepend generate -m $(PKGDEP_TOKENS:%=-D %) $(<) \
543                         $(PKGROOT) > $(@); \
544         else \
545                 $(CP) $(<) $(@); \
546         fi

548 #
549 # The full chain implies that there should be a .dep.res suffix rule,
550 # but dependency generation is done on a set of manifests, rather than
551 # on a per-manifest basis.  Instead, see the gendeps rule above.
552 #

554 $(PDIR)/%.pub: $(PDIR)/%.res
555         $(PKGDEBUG)m=$$(basename $(@:%.pub=%).metadata.*); \
556         r=$${m#$(@F:%.pub=%.metadata.)+(?).}; \
557         if [ -s $(<) ]; then \
558                 print "Publishing $(@F:%.pub=%) to $$r repository"; \
559                 pkgsend -s file://$(PKGDEST)/repo.$$r publish \
560                     -d $(PKGROOT) -d $(TOOLSROOT) \
561                     -d license_files -d $(PKGROOT)/licenses \
562                     --fmri-in-manifest --no-index --no-catalog $(<) \
563                     > /dev/null; \
564         fi; \
565         $(TOUCH) $(@);

567 #
568 # rule to build the synthetic manifests
569 #
570 # This rule necessarily has PKGDEP_TYPE that changes according to
571 # the specific synthetic manifest.  Rather than escape command
572 # dependency checking for the real manifest processing, or failing to
573 # express the (indirect) dependency of synthetic manifests on real
574 # manifests, we simply split this rule out from the one above.
575 #
576 # The implementation notes from the previous rule are applicable
577 # here, too.
578 #
579 $(PROC_SYNTH_PKGS): $(PKGLISTS) $$(@F:%.mog=%.mf)
580         @print "Processing synthetic manifest $(@F:%.mog=%.mf)"
581         $(PKGDEBUG)$(RM) $(@) $(PDIR)/$(@F:%.mog=%).metadata.* $(@).vars
582         $(PKGDEBUG)$(PKGMOGRIFY) $(PKGMOG_VERBOSE) -I transforms -I $(PDIR) \
583                 $(PKGMOG_DEFINES:%=-D %) -D PKGDEP_TYPE=$(PKGDEP_TYPE) \
584                 -P $(@).vars -O $(@) $(@F:%.mog=%.mf) \
585                 $(PM_TRANSFORMS) synthetic
586         $(PKGDEBUG)eval REPO=redist PKGSTAT=current '$(CAT) -s $(@).vars'; \
```



```
587         if [ -f $(@) ]; then \
588                 $(LN) -s $(@F) \
589                         $(PDIR)/$(@F:%.mog=%).metadata.$$PKGSTAT.$$REPO; \
590         else \
591                 $(TOUCH) $(@); \
592         fi
593         $(PKGDEBUG)$(RM) $(@).vars

595 $(DEP_SYNTH_PKGS): $$(@:%.dep=%.mog)
596         @print "Skipping dependency generation for $(@F:%.dep=%)"
597         $(PKGDEBUG)$(CP) $(@:%.dep=%.mog) $(@)

599 clean:

601 clobber: clean
602         $(RM) -r $(CLOBBERFILES)

604 #
605 # This rule assumes that all links in the $PKGSTAT directories
606 # point to valid manifests, and will fail the make run if one
607 # does not contain an fmri.
608 #
609 # We do this in the BEGIN action instead of using pattern matching
610 # because we expect the fmri to be at or near the first line of each input
611 # file, and this way lets us avoid reading the rest of the file after we
612 # find what we need.
613 #
614 # We keep track of a failure to locate an fmri, so we can fail the
615 # make run, but we still attempt to process each package in the
616 # repo/pkgstat-specific subdir, in hopes of maybe giving some
617 # additional useful info.
618 #
619 # The protolist is used for bfu archive creation, which may be invoked
620 # interactively by the user.  Both protolist and PKGLISTS targets
621 # depend on $(PROC_PKGS), but protolist builds them recursively.
622 # To avoid collisions, we insert protolist into the dependency chain
623 # here.  This has two somewhat subtle benefits: it allows bfu archive
624 # creation to work correctly, even when -a was not part of NIGHTLY_OPTIONS,
625 # and it ensures that a protolist file here will always correspond to the
626 # contents of the processed manifests, which can vary depending on build
627 # environment.
628 #
629 $(PKGLISTS): $(PROC_PKGS)
630         $(PKGDEBUG)sdotr=$(@F:packages.%=%); \
631         r=$${sdotr%.+(?)}; s=$${sdotr#+(?).}; \
632         print "Generating $$r $$s package list"; \
633         $(RM) $(@); $(TOUCH) $(@); \
634         $(AWK) 'BEGIN { \
635                 if (ARGC < 2) { \
636                         exit; \
637                 } \
638                 retcode = 0; \
639                 for (i = 1; i < ARGC; i++) { \
640                         do { \
641                                 e = getline f < ARGV[i]; \
642                         } while ((e == 1) && (f !~ /name=pkg.fmri/)); \
643                         close(ARGV[i]); \
644                         if (e == 1) { \
645                                 l = split(f, a, "="); \
646                                 print "depend fmri=" a[l], \
647                                         "type=$$(PKGDEP_TYPE)"; \
648                         } else { \
649                                 print "no fmri in " ARGV[i] >> "/dev/stderr"; \
650                                 retcode = 2; \
651                         } \
652                 } \
```

```
653                 exit retcode; \
654         }' `find $(PDIR) -type l -a \( $(PKGS:%=-name %.metadata.$$s.$$r -o) \
655                 -name NOSUCHFILE \)` >> $(@)

657 #
658 # rules to validate proto area against manifests, check for safe
659 # file permission modes, and generate a faux proto list
660 #
661 # For the check targets, the dependencies on $(PROC_PKGS) is specified
662 # as a subordinate make process in order to suppress output.
663 #
664 makesilent:
665         @$(MAKE) -e $(PROC_PKGS) PKGMACH=$(PKGMACH) \
666                 SUPPRESSPKGDEP=$(SUPPRESSPKGDEP) > /dev/null

668 #
669 # The .lics files were created during pkgmogrification, and list the
670 # set of licenses to pull from $SRC for each package.  Because
671 # licenses may be duplicated between packages, we uniquify them as
672 # well as aggregating them here.
673 #
674 license-list: makesilent
675         $(PKGDEBUG)( for l in `cat $(PROC_PKGS:%.mog=%.lics)`; \
676                 do print $$l; done ) | sort -u > $@

678 #
679 # Staging the license and description files in the proto area allows
680 # us to do proper unreferenced file checking of both license and
681 # description files without blanket exceptions, and to pull license
682 # content without reference to $CODEMGR_WS during publication.
683 #
684 stage-licenses: license-list FRC
685         $(PKGDEBUG)$(MAKE) -e -f Makefile.lic \
686                 PKGDEBUG=$(PKGDEBUG) LICROOT=$(PKGROOT)/licenses \
687                 `$(AWK) '{ \
688                         print "$(PKGROOT)/licenses/" $$0; \
689                         print "$(PKGROOT)/licenses/" $$0 ".descrip"; \
690                 }' license-list` > /dev/null;

692 protocmp: makesilent
693         @validate_pkg -a $(PKGMACH) -v \
694                 $(EXCEPTIONS:%=-e $(CODEMGR_WS)/exception_lists/%) \
695                 -m $(PDIR) -p $(PKGROOT) -p $(TOOLSROOT)

697 pmodes: makesilent
698         @validate_pkg -a $(PKGMACH) -M -m $(PDIR) \
699                 -e $(CODEMGR_WS)/exception_lists/pmodes

701 check: protocmp pmodes

703 protolist: proto_list_$(PKGMACH)

705 proto_list_$(PKGMACH): $(PROC_PKGS)
706         @validate_pkg -a $(PKGMACH) -L -m $(PDIR) > $(@)

708 $(PROC_PKGS): $(PDIR)

710 #
711 # This is a convenience target to allow package names to function as
712 # build targets.  Generally, using it is only useful when iterating on
713 # development of a manifest.
714 #
715 # When processing a manifest, use the basename (without extension) of
716 # the package.  When publishing, use the basename with a ".pub"
717 # extension.
718 #
```



```
719 # Other than during manifest development, the preferred usage is to
720 # avoid these targets and override PKGS on the make command line and
721 # use the provided all and install targets.
722 #
723 $(PKGS) $(SYNTH_PKGS): $(PDIR)/$$(@:%=%.mog)

725 $(PKGS:%=%.pub) $(SYNTH_PKGS:%=%.pub): $(PDIR)/$$(@)

727 #
728 # This is a convenience target to resolve dependencies without publishing
729 # packages.
730 #
731 gendeps: $(PDIR)/gendeps

733 #
734 # These are convenience targets for cross-platform packaging.  If you
735 # want to build any of "the normal" targets for a different
736 # architecture, simply use "arch/target" as your build target.
737 #
738 # Since the most common use case for this is "install," the architecture
739 # specific install targets have been further abbreviated to elide "/install."
740 #
741 i386/% sparc/%:
742         $(MAKE) -e $(@F) PKGMACH=$(@D) SUPPRESSPKGDEP=$(SUPPRESSPKGDEP)

744 i386 sparc: $$(@)/install

746 FRC:
```

****
1578 Mon Jan 11 15:37:03 2016
new/usr/src/tools/ctf/Makefile.ctf
native tools must reliably use a native adjunct, even if that's inconvenient
While it is perhaps convenient for native tools to use updated versions
of certain things like libxml it is imperative that those versions are
also build for the build machine. Thus they need to be in the native
adjunct (even if that native adjunct is thus not /).
Fix the native adjunct to be rooted similarly to the adjunct proto (that
is, at /), and fix SMF to use it correctly
****

```
#
# 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.
#

include ../../../Makefile.tools

#
# A 'make install' from the tools directory needs to work, even if the rest of
# the tree hasn't been built.  As such, we need to tell the ctf builds how
# to find the ctf specific headers located outside the tools subtree. We also
# want to tell them how to get to the common tools source files.
#
# For additonal details on the ordering of includes via -I, see the comments
# in $(SRC)/tools/ctf/common/ctf_headers.h.
#

HDRDIRS= \
        -_gcc=-nostdinc \
        -I../../common \
        -I$(SRC) \
        -I/usr/include \
        -I$(SRC)/uts/common \
        -I$(NATIVE_ADJUNCT)/usr/include
        -I$(NATIVE_ADJUNCT)/include

CPPFLAGS += $(HDRDIRS)
CFLAGS += $(CCVERBOSE)
CERRWARN += -_gcc=-Wno-parentheses
```

```
**********************************************************
    1984 Mon Jan 11 15:37:03 2016
new/usr/src/tools/ctf/cvt/Makefile.com
native tools must reliably use a native adjunct, even if that's inconvenient
While it is perhaps convenient for native tools to use updated versions
of certain things like libxml it is imperative that those versions are
_also_ build for the build machine.  Thus they need to be in the native
adjunct (even if that native adjunct is thus not /).
Fix the native adjunct to be rooted similarly to the adjunct proto (that
is, at /), and fix SMF to use it correctly
**********************************************************
#
# 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.
#

include ../../Makefile.ctf

.KEEP_STATE:
.PARALLEL:

PROG=ctfconvert ctfmerge

GENSRCS= \
        alist.c \
        barrier.c \
        ctf.c \
        fifo.c \
        hash.c \
        iidesc.c \
        input.c \
        list.c \
        memory.c \
        merge.c \
        output.c \
        stack.c \
        strtab.c \
        symbol.c \
        tdata.c \
        traverse.c \
        util.c

CVTSRCS=$(GENSRCS) \
        ctfconvert.c \
        dwarf.c \
        stabs.c \
```



```
        fixup_tdescs.c \
        st_parse.c
CVTOBJS=$(CVTSRCS:%.c=%.o)
CVTLINTFILES = $(CVTSRCS:%.c=%.ln)

MRGSRCS=$(GENSRCS) \
        ctfmerge.c
MRGOBJS=$(MRGSRCS:%.c=%.o)
MRGLINTFILES = $(MRGSRCS:%.c=%.ln)

SRCS=$(CVTSRCS) $(MRGSRCS) $(CMPSRCS)
OBJS=$(SRCS:%.c=%.o)
LINTFILES=$(SRCS:%.c=%.ln)

DWARFLDFLAGS    = \
        -L$(ROOTONBLDLIBMACH) \
        '-R$$ORIGIN/../../lib/$(MACH)' \
        -ldwarf
DWARFCPPFLAGS   = -I../../dwarf/common

LDFLAGS         += -L$(NATIVE_ADJUNCT)/usr/lib
LDFLAGS         += -L$(NATIVE_ADJUNCT)/lib
LDLIBS          += -lz -lelf
CPPFLAGS        += -D_REENTRANT
CFLAGS          += $(CTF_FLAGS)
LINTFLAGS       += -mnux

CERRWARN        += -_gcc=-Wno-unused
CERRWARN        += -_gcc=-Wno-uninitialized
CERRWARN        += -_gcc=-Wno-switch

C99MODE         = $(C99_ENABLE)

ctfconvert      := LDFLAGS += $(DWARFLDFLAGS)

dwarf.o dwarf.ln        := CPPFLAGS += $(DWARFCPPFLAGS)
```

```
**********************************************************
    1539 Mon Jan 11 15:37:04 2016
new/usr/src/tools/ctf/dump/Makefile.com
native tools must reliably use a native adjunct, even if that's inconvenient
While it is perhaps convenient for native tools to use updated versions
of certain things like libxml it is imperative that those versions are
_also_ build for the build machine.  Thus they need to be in the native
adjunct (even if that native adjunct is thus not /).
Fix the native adjunct to be rooted similarly to the adjunct proto (that
is, at /), and fix SMF to use it correctly
**********************************************************
```

```
#
# 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 2009 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#

.KEEP_STATE:
.SUFFIXES:

PROG = ctfdump
SRCS = dump.c utils.c symbol.c

include ../../Makefile.ctf

LDFLAGS += -L$(NATIVE_ADJUNCT)/usr/lib
LDFLAGS += -L$(NATIVE_ADJUNCT)/lib
LDLIBS += -lelf -lz

OBJS = $(SRCS:%.c=%.o)
LINTFILES = $(SRCS:%.c=%.ln)

CERRWARN += -_gcc=-Wno-uninitialized

.NO_PARALLEL:
.PARALLEL: $(OBJS) $(LINTFILES)

all: $(PROG)

$(PROG): $(OBJS)
        $(LINK.c) $(OBJS) -o $@ $(LDLIBS)
        $(POST_PROCESS)

%.o: ../%.c
        $(COMPILE.c) $<

$(ROOTONBLDMACHPROG): $(PROG)
```



```
install: $(ROOTONBLDMACHPROG)

clean:
        $(RM) $(OBJS) $(LINTFILES)

%.ln: ../%.c
        $(LINT.c) -c $<

lint: $(LINTFILES)
        $(LINT) $(LINTFLAGS) $(LINTFILES) $(LDLIBS)

include ../../Makefile.ctf.targ
```