```
**********************************************************
   35320 Mon Jul 18 14:42:49 2016
new/usr/src/Makefile.master
7170 ENABLE_PERL64 multiarch builds
reviewed by: Andrew Stormont <andyjstormont@gmail.com>
reviewed by: Lauri Tirkkonen <lotheac@iki.fi>
**********************************************************
#
# 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= /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)
ENABLE_PERL64=          $(POUND_SIGN)

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

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

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

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

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

```
126 GCCLIBDIR64=    $(GCC_ROOT)/lib/$(MACH64)

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

130 RPCGEN=         /usr/bin/rpcgen
131 STABS=          $(ONBLD_TOOLS)/bin/$(MACH)/stabs
132 ELFEXTRACT=     $(ONBLD_TOOLS)/bin/$(MACH)/elfextract
133 MBH_PATCH=      $(ONBLD_TOOLS)/bin/$(MACH)/mbh_patch
134 ECHO=           echo
135 INS=            install
136 TRUE=           true
137 SYMLINK=        /usr/bin/ln -s
138 LN=             /usr/bin/ln
139 CHMOD=          /usr/bin/chmod
140 MV=             /usr/bin/mv -f
141 RM=             /usr/bin/rm -f
142 CUT=            /usr/bin/cut
143 NM=             /usr/ccs/bin/nm
144 DIFF=           /usr/bin/diff
145 GREP=           /usr/bin/grep
146 EGREP=          /usr/bin/egrep
147 ELFWRAP=        /usr/bin/elfwrap
148 KSH93=          /usr/bin/ksh93
149 SED=            /usr/bin/sed
150 AWK=            /usr/bin/nawk
151 CP=             /usr/bin/cp -f
152 MCS=            /usr/ccs/bin/mcs
153 CAT=            /usr/bin/cat
154 ELFDUMP=        /usr/ccs/bin/elfdump
155 M4=             /usr/bin/m4
156 STRIP=          /usr/ccs/bin/strip
157 LEX=            /usr/ccs/bin/lex
158 FLEX=           /usr/bin/flex
159 YACC=           /usr/ccs/bin/yacc
160 CPP=            /usr/lib/cpp
161 JAVAC=          $(JAVA_ROOT)/bin/javac
162 JAVAH=          $(JAVA_ROOT)/bin/javah
163 JAVADOC=        $(JAVA_ROOT)/bin/javadoc
164 RMIC=           $(JAVA_ROOT)/bin/rmic
165 JAR=            $(JAVA_ROOT)/bin/jar
166 CTFCONVERT=     $(ONBLD_TOOLS)/bin/$(MACH)/ctfconvert
167 CTFMERGE=       $(ONBLD_TOOLS)/bin/$(MACH)/ctfmerge
168 CTFSTABS=       $(ONBLD_TOOLS)/bin/$(MACH)/ctfstabs
169 CTFSTRIP=       $(ONBLD_TOOLS)/bin/$(MACH)/ctfstrip
170 NDRGEN=         $(ONBLD_TOOLS)/bin/$(MACH)/ndrgen
171 GENOFFSETS=     $(ONBLD_TOOLS)/bin/genoffsets
172 XREF=           $(ONBLD_TOOLS)/bin/xref
173 FIND=           /usr/bin/find
174 PERL=           /usr/bin/perl
175 PERL_VERSION=   5.10.0
176 PERL_PKGVERS=   -510
177 PERL_ARCH=      i86pc-solaris-64int
178 PERL_ARCH64=    i86pc-solaris-64
179 $(SPARC_BLD)PERL_ARCH=          sun4-solaris-64int
180 $(SPARC_BLD)PERL_ARCH64=        sun4-solaris-64
176 PERL_ARCH =             i86pc-solaris-64int
177 $(SPARC_BLD)PERL_ARCH = sun4-solaris-64int
181 PYTHON_26=      /usr/bin/python2.6
182 PYTHON=         $(PYTHON_26)
183 SORT=           /usr/bin/sort
184 TOUCH=          /usr/bin/touch
185 WC=             /usr/bin/wc
186 XARGS=          /usr/bin/xargs
187 ELFEDIT=        /usr/bin/elfedit
188 ELFSIGN=        /usr/bin/elfsign
189 DTRACE=         /usr/sbin/dtrace -xnolibs
```



```
190 UNIQ=           /usr/bin/uniq
191 TAR=            /usr/bin/tar
192 ASTBINDIR=      /usr/ast/bin
193 MSGCC=          $(ASTBINDIR)/msgcc
194 MSGFMT=         /usr/bin/msgfmt -s

196 FILEMODE=       644
197 DIRMODE=        755

199 # Declare that nothing should be built in parallel.
200 # Individual Makefiles can use the .PARALLEL target to declare otherwise.
201 .NO_PARALLEL:

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

214 DOT_H_CHECK=    \
215         @$(ECHO) "checking $<"; $(CSTYLE) $< $(CSTYLE_TAIL); \
216         $(HDRCHK) $< $(HDRCHK_TAIL)

218 DOT_X_CHECK=    \
219         @$(ECHO) "checking $<"; $(RPCGEN) -C -h $< | $(CSTYLE) $(CSTYLE_TAIL); \
220         $(RPCGEN) -C -h $< | $(HDRCHK) $< $(HDRCHK_TAIL)

222 DOT_C_CHECK=    \
223         @$(ECHO) "checking $<"; $(CSTYLE) $< $(CSTYLE_TAIL)

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

232 INS.file=       $(RM) $@; $(INS) -s -m $(FILEMODE) -f $(@D) $<
233 INS.dir=        $(INS) -s -d -m $(DIRMODE) $@
234 # installs and renames at once
235 #
236 INS.rename=     $(INS.file); $(MV) $(@D)/$(<F) $@

238 # install a link
239 INSLINKTARGET=  $<
240 INS.link=       $(RM) $@; $(LN) $(INSLINKTARGET) $@
241 INS.symlink=    $(RM) $@; $(SYMLINK) $(INSLINKTARGET) $@

243 #
244 # Python bakes the mtime of the .py file into the compiled .pyc and
245 # rebuilds if the baked-in mtime != the mtime of the source file
246 # (rather than only if it's less than), thus when installing python
247 # files we must make certain to not adjust the mtime of the source
248 # (.py) file.
249 #
250 INS.pyfile=     $(INS.file); $(TOUCH) -r $< $@

252 # MACH must be set in the shell environment per uname -p on the build host
253 # More specific architecture variables should be set in lower makefiles.
254 #
255 # MACH64 is derived from MACH, and BUILD64 is set to '#' for
```

```
256 # architectures on which we do not build 64-bit versions.
257 # (There are no such architectures at the moment.)
258 #
259 # Set BUILD64=# in the environment to disable 64-bit amd64
260 # builds on i386 machines.

262 MACH64_1=       $(MACH:sparc=sparcv9)
263 MACH64=         $(MACH64_1:i386=amd64)

265 MACH32_1=       $(MACH:sparc=sparcv7)
266 MACH32=         $(MACH32_1:i386=i86)

268 sparc_BUILD64=
269 i386_BUILD64=
270 BUILD64=        $($(MACH)_BUILD64)

272 #
273 # C compiler mode. Future compilers may change the default on us,
274 # so force extended ANSI mode globally. Lower level makefiles can
275 # override this by setting CCMODE.
276 #
277 CCMODE=                 -Xa
278 CCMODE64=               -Xa

280 #
281 # C compiler verbose mode. This is so we can enable it globally,
282 # but turn it off in the lower level makefiles of things we cannot
283 # (or aren't going to) fix.
284 #
285 CCVERBOSE=              -v

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

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

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

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



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

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

345 # assembler '-xarch' flag.  Different from compiler '-xarch' flag.
346 sparc_AS_XARCH=         -xarch=v8plus
347 sparcv9_AS_XARCH=       -xarch=v9
348 i386_AS_XARCH=
349 amd64_AS_XARCH=         -xarch=amd64 -P -Ui386 -U__i386

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

366 SAVEARGS=               -Wu,-save_args
367 amd64_STAND_FLAGS       += $(SAVEARGS)

369 STAND_FLAGS_32 = $($(MACH)_STAND_FLAGS)
370 STAND_FLAGS_64 = $($(MACH64)_STAND_FLAGS)

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

381 #
382 # turn warnings into errors (C)
383 CERRWARN = -errtags=yes -errwarn=%all
384 CERRWARN += -erroff=E_EMPTY_TRANSLATION_UNIT
385 CERRWARN += -erroff=E_STATEMENT_NOT_REACHED

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

```
388 CERRWARN += -_gcc=-Wno-sign-compare
389 CERRWARN += -_gcc=-Wno-unknown-pragmas
390 CERRWARN += -_gcc=-Wno-unused-parameter
391 CERRWARN += -_gcc=-Wno-missing-field-initializers

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

396 # DEBUG v. -nd make for frequent unused variables, empty conditions, etc. in
397 # -nd builds
398 $(RELEASE_BUILD)CERRWARN += -_gcc=-Wno-unused
399 $(RELEASE_BUILD)CERRWARN += -_gcc=-Wno-empty-body

401 #
402 # turn warnings into errors (C++)
403 CCERRWARN=              -xwe

405 # C99 mode
406 C99_ENABLE=     -xc99=%all
407 C99_DISABLE=    -xc99=%none
408 C99MODE=        $(C99_DISABLE)
409 C99LMODE=       $(C99MODE:-xc99%=-Xc99%)

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

419 sparc_ASFLAGS=  $(sparc_AS_XARCH)
420 sparcv9_ASFLAGS=$(sparcv9_AS_XARCH)
421 i386_ASFLAGS=   $(i386_AS_XARCH)
422 amd64_ASFLAGS=  $(amd64_AS_XARCH)

424 #
425 sparc_COPTFLAG=         -xO3
426 sparcv9_COPTFLAG=       -xO3
427 i386_COPTFLAG=          -O
428 amd64_COPTFLAG=         -xO3

430 COPTFLAG= $($(MACH)_COPTFLAG)
431 COPTFLAG64= $($(MACH64)_COPTFLAG)

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

437 # Direct the Sun Studio compiler to use a static globalization prefix based on t
438 # name of the module rather than something unique. Otherwise, objects
439 # will not build deterministically, as subsequent compilations of identical
440 # source will yeild objects that always look different.
441 #
442 # In the same spirit, this will also remove the date from the N_OPT stab.
443 CGLOBALSTATIC= -W0,-xglobalstatic

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

449 #
450 # Default debug format for Sun Studio 11 is dwarf, so force it to
451 # generate stabs.
452 #
453 DEBUGFORMAT=    -xdebugformat=stabs
```



```
455 #
456 # Flags used to build in debug mode for ctf generation.  Bugs in the Devpro
457 # compilers currently prevent us from building with cc-emitted DWARF.
458 #
459 CTF_FLAGS_sparc = -g -Wc,-Qiselect-T1 $(C99MODE) $(CNOGLOBAL) $(CDWARFSTR)
460 CTF_FLAGS_i386  = -g $(C99MODE) $(CNOGLOBAL) $(CDWARFSTR)

462 CTF_FLAGS_sparcv9       = $(CTF_FLAGS_sparc)
463 CTF_FLAGS_amd64         = $(CTF_FLAGS_i386)

465 # Sun Studio produces broken userland code when saving arguments.
466 $(__GNUC)CTF_FLAGS_amd64 += $(SAVEARGS)

468 CTF_FLAGS_32    = $(CTF_FLAGS_$(MACH)) $(DEBUGFORMAT)
469 CTF_FLAGS_64    = $(CTF_FLAGS_$(MACH64)) $(DEBUGFORMAT)
470 CTF_FLAGS       = $(CTF_FLAGS_32)

472 #
473 # Flags used with genoffsets
474 #
475 GOFLAGS = -_noecho \
476         $(CALLSYMS) \
477         $(CDWARFSTR)

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

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

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

493 SPACEFLAG               = $($(MACH)_SPACEFLAG)
494 SPACEFLAG64             = $($(MACH64)_SPACEFLAG)

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

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

509 sparc_XREGSFLAG         = -xregs=no%appl
510 sparcv9_XREGSFLAG       = -xregs=no%appl
511 i386_XREGSFLAG          =
512 amd64_XREGSFLAG         =

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

517 # dmake SOURCEDEBUG=yes ... enables source-level debugging information, and
518 # avoids stripping it.
519 SOURCEDEBUG     = $(POUND_SIGN)
```

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

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

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

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

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

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

CLOBBERFILES += $(POFILE) $(POFILES)
COMPILE.cpp= $(CC) -E -C $(CFLAGS) $(CPPFLAGS)
XGETTEXT= /usr/bin/xgettext
XGETFLAGS= -c TRANSLATION_NOTE
GNUXGETTEXT= /usr/gnu/bin/xgettext
GNUXGETFLAGS= --add-comments=TRANSLATION_NOTE --keyword=_ \
```



```
        --strict --no-location --omit-header
BUILD.po= $(XGETTEXT) $(XGETFLAGS) -d $(<F) $<.i ;\
        $(RM)   $@ ;\
        $(SED) "/^domain/d" < $(<F).po > $@ ;\
        $(RM) $(<F).po $<.i

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

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

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

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

#
#
#
ELFWRAP_FLAGS   =
ELFWRAP_FLAGS64 =       -64

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

#
# Generated mapfiles that are compiler specific, and used throughout the
# build.  These mapfiles are not delivered in /usr/lib/ld.
#
MAPFILE.NGB_sparc=      $(SRC)/common/mapfiles/gen/sparc_cc_map.noexeglobs
$(__GNUC64)MAPFILE.NGB_sparc= \
                        $(SRC)/common/mapfiles/gen/sparc_gcc_map.noexeglobs
MAPFILE.NGB_sparcv9=    $(SRC)/common/mapfiles/gen/sparcv9_cc_map.noexeglobs
```

```
652 $(__GNUC64)MAPFILE.NGB_sparcv9= \
653                     $(SRC)/common/mapfiles/gen/sparcv9_gcc_map.noexeglobs
654 MAPFILE.NGB_i386=       $(SRC)/common/mapfiles/gen/i386_cc_map.noexeglobs
655 $(__GNUC64)MAPFILE.NGB_i386= \
656                     $(SRC)/common/mapfiles/gen/i386_gcc_map.noexeglobs
657 MAPFILE.NGB_amd64=      $(SRC)/common/mapfiles/gen/amd64_cc_map.noexeglobs
658 $(__GNUC64)MAPFILE.NGB_amd64= \
659                     $(SRC)/common/mapfiles/gen/amd64_gcc_map.noexeglobs
660 MAPFILE.NGB =           $(MAPFILE.NGB_$(MACH))

662 #
663 # A generic interface mapfile name, used by various dynamic objects to define
664 # the interfaces and interposers the object must export.
665 #
666 MAPFILE.INT =           mapfile-intf

668 #
669 # LDLIBS32 and LDLIBS64 can be set in the environment to override the following
670 # assignments.
671 #
672 # These environment settings make sure that no libraries are searched outside
673 # of the local workspace proto area:
674 #       LDLIBS32=-YP,$ROOT/lib:$ROOT/usr/lib
675 #       LDLIBS64=-YP,$ROOT/lib/$MACH64:$ROOT/usr/lib/$MACH64
676 #
677 LDLIBS32 =      $(ENVLDLIBS1) $(ENVLDLIBS2) $(ENVLDLIBS3)
678 LDLIBS32 +=     $(ADJUNCT_PROTO:%=-L%/usr/lib -L%/lib)
679 LDLIBS.cmd =    $(LDLIBS32)
680 LDLIBS.lib =    $(LDLIBS32)

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

687 #
688 # Define compilation macros.
689 #
690 COMPILE.c=      $(CC) $(CFLAGS) $(CPPFLAGS) -c
691 COMPILE64.c=    $(CC) $(CFLAGS64) $(CPPFLAGS) -c
692 COMPILE.cc=     $(CCC) $(CCFLAGS) $(CPPFLAGS) -c
693 COMPILE64.cc=   $(CCC) $(CCFLAGS64) $(CPPFLAGS) -c
694 COMPILE.s=      $(AS) $(ASFLAGS) $(AS_CPPFLAGS)
695 COMPILE64.s=    $(AS) $(ASFLAGS) $($(MACH64)_AS_XARCH) $(AS_CPPFLAGS)
696 COMPILE.d=      $(DTRACE) -G -32
697 COMPILE64.d=    $(DTRACE) -G -64
698 COMPILE.b=      $(ELFWRAP) $(ELFWRAP_FLAGS$(CLASS))
699 COMPILE64.b=    $(ELFWRAP) $(ELFWRAP_FLAGS$(CLASS))

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

704 #
705 # Link time macros
706 #
707 CCNEEDED                = -lC
708 CCEXTNEEDED             = -lCrun -lCstd
709 $(__GNUC)CCNEEDED       = -L$(GCCLIBDIR) -lstdc++ -lgcc_s
710 $(__GNUC)CCEXTNEEDED    = $(CCNEEDED)

712 LINK.c=         $(CC) $(CFLAGS) $(CPPFLAGS) $(LDFLAGS)
713 LINK64.c=       $(CC) $(CFLAGS64) $(CPPFLAGS) $(LDFLAGS)
714 NORUNPATH=      -norunpath -nolib
715 LINK.cc=        $(CCC) $(CCFLAGS) $(CPPFLAGS) $(NORUNPATH) \
716                 $(LDFLAGS) $(CCNEEDED)
717 LINK64.cc=      $(CCC) $(CCFLAGS64) $(CPPFLAGS) $(NORUNPATH) \
```



```
718                 $(LDFLAGS) $(CCNEEDED)

720 #
721 # lint macros
722 #
723 # Note that the undefine of __PRAGMA_REDEFINE_EXTNAME can be removed once
724 # ON is built with a version of lint that has the fix for 4484186.
725 #
726 ALWAYS_LINT_DEFS =      -errtags=yes -s
727 ALWAYS_LINT_DEFS +=     -erroff=E_PTRDIFF_OVERFLOW
728 ALWAYS_LINT_DEFS +=     -erroff=E_ASSIGN_NARROW_CONV
729 ALWAYS_LINT_DEFS +=     -U__PRAGMA_REDEFINE_EXTNAME
730 ALWAYS_LINT_DEFS +=     $(C99LMODE)
731 ALWAYS_LINT_DEFS +=     -errsecurity=$(SECLEVEL)
732 ALWAYS_LINT_DEFS +=     -erroff=E_SEC_CREAT_WITHOUT_EXCL
733 ALWAYS_LINT_DEFS +=     -erroff=E_SEC_FORBIDDEN_WARN_CREAT
734 # XX64 -- really only needed for amd64 lint
735 ALWAYS_LINT_DEFS +=     -erroff=E_ASSIGN_INT_TO_SMALL_INT
736 ALWAYS_LINT_DEFS +=     -erroff=E_CAST_INT_CONST_TO_SMALL_INT
737 ALWAYS_LINT_DEFS +=     -erroff=E_CAST_INT_TO_SMALL_INT
738 ALWAYS_LINT_DEFS +=     -erroff=E_CAST_TO_PTR_FROM_INT
739 ALWAYS_LINT_DEFS +=     -erroff=E_COMP_INT_WITH_LARGE_INT
740 ALWAYS_LINT_DEFS +=     -erroff=E_INTEGRAL_CONST_EXP_EXPECTED
741 ALWAYS_LINT_DEFS +=     -erroff=E_PASS_INT_TO_SMALL_INT
742 ALWAYS_LINT_DEFS +=     -erroff=E_PTR_CONV_LOSES_BITS

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

748 SECLEVEL=       core
749 LINT.c=         $(LINT) $(ONLY_LINT_DEFS) $(LINTFLAGS) $(CPPFLAGS) \
750                 $(ALWAYS_LINT_DEFS)
751 LINT64.c=       $(LINT) $(ONLY_LINT_DEFS) $(LINTFLAGS64) $(CPPFLAGS) \
752                 $(ALWAYS_LINT_DEFS)
753 LINT.s=         $(LINT.c)

755 # For some future builds, NATIVE_MACH and MACH might be different.
756 # Therefore, NATIVE_MACH needs to be redefined in the
757 # environment as ‘uname -p‘ to override this macro.
758 #
759 # For now at least, we cross-compile amd64 on i386 machines.
760 NATIVE_MACH=    $(MACH:amd64=i386)

762 # Define native compilation macros
763 #

765 # Base directory where compilers are loaded.
766 # Defined here so it can be overridden by developer.
767 #
768 SPRO_ROOT=              $(BUILD_TOOLS)/SUNWspro
769 SPRO_VROOT=             $(SPRO_ROOT)/SS12
770 GNU_ROOT=               /usr

772 # Till SS12u1 formally becomes the NV CBE, LINT is hard
773 # coded to be picked up from the $SPRO_ROOT/sunstudio12.1/
774 # location. Impacted variables are sparc_LINT, sparcv9_LINT,
775 # i386_LINT, amd64_LINT.
776 # Reset them when SS12u1 is rolled out.
777 #

779 # Specify platform compiler versions for languages
780 # that we use (currently only c and c++).
781 #
782 sparc_CC=               $(ONBLD_TOOLS)/bin/$(MACH)/cw -_cc
783 $(__GNUC)sparc_CC=      $(ONBLD_TOOLS)/bin/$(MACH)/cw -_gcc
```

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

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

800 i386_CC=              $(ONBLD_TOOLS)/bin/$(MACH)/cw -_cc
801 $(__GNUC)i386_CC=     $(ONBLD_TOOLS)/bin/$(MACH)/cw -_gcc
802 i386_CCC=            $(ONBLD_TOOLS)/bin/$(MACH)/cw -_CC
803 $(__GNUC)i386_CCC=    $(ONBLD_TOOLS)/bin/$(MACH)/cw -_g++
804 i386_CPP=            /usr/ccs/lib/cpp
805 i386_AS=             /usr/ccs/bin/as
806 $(__GNUC)i386_AS=     $(ONBLD_TOOLS)/bin/$(MACH)/aw
807 i386_LD=             /usr/ccs/bin/ld
808 i386_LINT=           $(SPRO_ROOT)/sunstudio12.1/bin/lint

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

819 NATIVECC=            $($(NATIVE_MACH)_CC)
820 NATIVECCC=           $($(NATIVE_MACH)_CCC)
821 NATIVECPP=           $($(NATIVE_MACH)_CPP)
822 NATIVEAS=            $($(NATIVE_MACH)_AS)
823 NATIVELD=            $($(NATIVE_MACH)_LD)
824 NATIVELINT=          $($(NATIVE_MACH)_LINT)

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

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

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

845 BDIRECT=        -Bdirect
846 BDYNAMIC=       -Bdynamic
847 BLOCAL=         -Blocal
848 BNODIRECT=      -Bnodirect
849 BREDUCE=        -Breduce
```



```
850 BSTATIC=        -Bstatic

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

874 GSHARED=        -G
875 CCMT=           -mt

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

880 sparc_C_PICFLAGS =      -K pic
881 sparcv9_C_PICFLAGS =    -K pic
882 i386_C_PICFLAGS =       -K pic
883 amd64_C_PICFLAGS =      -K pic
884 C_PICFLAGS =            $($(MACH)_C_PICFLAGS)
885 C_PICFLAGS64 =          $($(MACH64)_C_PICFLAGS)

887 sparc_C_BIGPICFLAGS =   -K PIC
888 sparcv9_C_BIGPICFLAGS = -K PIC
889 i386_C_BIGPICFLAGS =    -K PIC
890 amd64_C_BIGPICFLAGS =   -K PIC
891 C_BIGPICFLAGS =         $($(MACH)_C_BIGPICFLAGS)
892 C_BIGPICFLAGS64 =       $($(MACH64)_C_BIGPICFLAGS)

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

902 AS_PICFLAGS=            $(C_PICFLAGS)
903 AS_BIGPICFLAGS=         $(C_BIGPICFLAGS)

905 #
906 # Default label for CTF sections
907 #
908 CTFCVTFLAGS=            -i -L VERSION

910 #
911 # Override to pass module-specific flags to ctfmerge.  Currently used only by
912 # krtld to turn on fuzzy matching, and source-level debugging to inhibit
913 # stripping.
914 #
915 CTFMRGFLAGS=
```

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

ELFSIGN_O=      $(TRUE)
ELFSIGN_CRYPTO= $(ELFSIGN_O)
ELFSIGN_OBJECT= $(ELFSIGN_O)

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

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

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

POST_PROCESS_O=
```



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

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

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

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

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

```
1048         $(RM) y.tab.c
1049 .y.o:
1050         $(YACC.y) $<
1051         $(COMPILE.c) -o $@ y.tab.c $(CTFCONVERT_HOOK)
1052         $(POST_PROCESS_O)
1053         $(RM) y.tab.c
1054 .l:
1055         $(RM) $*.c
1056         $(LEX.l) $< > $*.c
1057         $(LINK.c) -o $@ $*.c -ll $(LDLIBS)
1058         $(POST_PROCESS)
1059         $(RM) $*.c
1060 .l.o:
1061         $(RM) $*.c
1062         $(LEX.l) $< > $*.c
1063         $(COMPILE.c) -o $@ $*.c $(CTFCONVERT_HOOK)
1064         $(POST_PROCESS_O)
1065         $(RM) $*.c

1067 .bin.o:
1068         $(COMPILE.b) -o $@ $<
1069         $(POST_PROCESS_O)

1071 .java.class:
1072         $(COMPILE.java) $<

1074 # Bourne and Korn shell script message catalog build rules.
1075 # We extract all gettext strings with sed(1) (being careful to permit
1076 # multiple gettext strings on the same line), weed out the dups, and
1077 # build the catalogue with awk(1).

1079 .sh.po .ksh.po:
1080         $(SED) -n -e ":a"                                       \
1081                   -e "h"                                        \
1082                   -e "s/.*gettext *\(\"[^\"]*\"\).*/\1/p"       \
1083                   -e "x"                                        \
1084                   -e "s/\(.*\)gettext *\"[^\"]*\"\(.*\)/\1\2/"  \
1085                   -e "t a"                                      \
1086                $< | sort -u | $(AWK) '{ print "msgid\t" $$0 "\nmsgstr" }' > $@

1088 #
1089 # Python and Perl executable and message catalog build rules.
1090 #
1091 .SUFFIXES: .pl .pm .py .pyc

1093 .pl:
1094         $(RM) $@;
1095         $(SED) -e "s@TEXT_DOMAIN@\"$(TEXT_DOMAIN)\"@" $< > $@;
1096         $(CHMOD) +x $@

1098 .py:
1099         $(RM) $@; $(CAT) $< > $@; $(CHMOD) +x $@

1101 .py.pyc:
1102         $(RM) $@
1103         $(PYTHON) -mpy_compile $<
1104         @[ $(<)c = $@ ] || $(MV) $(<)c $@

1106 .py.po:
1107         $(GNUXGETTEXT) $(GNUXGETFLAGS) -d $(<F:%.py=%) $< ;

1109 .pl.po .pm.po:
1110         $(XGETTEXT) $(XGETFLAGS) -d $(<F) $< ;
1111         $(RM)   $@ ;
1112         $(SED) "/^domain/d" < $(<F).po > $@ ;
1113         $(RM) $(<F).po
```



```
1115 #
1116 # When using xgettext, we want messages to go to the default domain,
1117 # rather than the specified one.  This special version of the
1118 # COMPILE.cpp macro effectively prevents expansion of TEXT_DOMAIN,
1119 # causing xgettext to put all messages into the default domain.
1120 #
1121 CPPFORPO=$(COMPILE.cpp:\"$(TEXT_DOMAIN)\"=TEXT_DOMAIN)

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

1126 .h.i:
1127         $(CPPFORPO) $< > $@

1129 .y.i:
1130         $(YACC) -d $<
1131         $(CPPFORPO) y.tab.c  > $@
1132         $(RM) y.tab.c

1134 .l.i:
1135         $(LEX) $<
1136         $(CPPFORPO) lex.yy.c  > $@
1137         $(RM) lex.yy.c

1139 .c.po:
1140         $(CPPFORPO) $< > $<.i
1141         $(BUILD.po)

1143 .cc.po:
1144         $(CPPFORPO) $< > $<.i
1145         $(BUILD.po)

1147 .y.po:
1148         $(YACC) -d $<
1149         $(CPPFORPO) y.tab.c  > $<.i
1150         $(BUILD.po)
1151         $(RM) y.tab.c

1153 .l.po:
1154         $(LEX) $<
1155         $(CPPFORPO) lex.yy.c  > $<.i
1156         $(BUILD.po)
1157         $(RM) lex.yy.c

1159 #
1160 # Rules to perform stylistic checks
1161 #
1162 .SUFFIXES: .x .xml .check .xmlchk

1164 .h.check:
1165         $(DOT_H_CHECK)

1167 .x.check:
1168         $(DOT_X_CHECK)

1170 .xml.xmlchk:
1171         $(MANIFEST_CHECK)

1173 #
1174 # Include rules to render automated sccs get rules "safe".
1175 #
1176 include $(SRC)/Makefile.noget
```

****
1791 Mon Jul 18 14:42:49 2016
new/usr/src/cmd/perl/Makefile.perl
7170 ENABLE_PERL64 multiarch builds
reviewed by: Andrew Stormont <andyjstormont@gmail.com>
reviewed by: Lauri Tirkkonen <lotheac@iki.fi>
****

```
#
# This file and its contents are supplied under the terms of the
# Common Development and Distribution License ("CDDL"), version 1.0.
# You may only use this file in accordance with the terms of version
# 1.0 of the CDDL.
#
# A full copy of the text of the CDDL should have accompanied this
# source.  A copy of the CDDL is also available via the Internet at
# http://www.illumos.org/license/CDDL.
#
#
# Copyright (c) 2014 Racktop Systems.
# Copyright 2015, OmniTI Computer Consulting, Inc. All rights reserved.
#

include $(SRC)/lib/Makefile.lib

# PERL_VERSION and PERL_ARCH used to be set here,
# but as they were also needed in usr/src/pkg/Makefile,
# the definition was moved to usr/src/Makefile.master

PERLDIR = $(ADJUNCT_PROTO)/usr/perl5/$(PERL_VERSION)
PERLLIBDIR = $(PERLDIR)/lib/$(PERL_ARCH)
PERLLIBDIR64 = $(PERLDIR)/lib/$(PERL_ARCH64)
PERLINCDIR = $(PERLLIBDIR)/CORE
PERLINCDIR64 = $(PERLLIBDIR64)/CORE

PERLMOD = $(MODULE).pm
PERLEXT = $(MACH)/$(MODULE).so
PERLEXT64 = $(MACH64)/$(MODULE).so

ROOTPERLDIR = $(ROOT)/usr/perl5/$(PERL_VERSION)
ROOTPERLLIBDIR = $(ROOTPERLDIR)/lib/$(PERL_ARCH)
ROOTPERLMODDIR = $(ROOTPERLLIBDIR)/Sun/Solaris
ROOTPERLEXTDIR = $(ROOTPERLLIBDIR)/auto/Sun/Solaris/$(MODULE)
ROOTPERLLIBDIR64 = $(ROOTPERLDIR)/lib/$(PERL_ARCH64)
ROOTPERLMODDIR64 = $(ROOTPERLLIBDIR64)/Sun/Solaris
ROOTPERLEXTDIR64 = $(ROOTPERLLIBDIR64)/auto/Sun/Solaris/$(MODULE)

ROOTPERLMOD = $(ROOTPERLMODDIR)/$(MODULE).pm
ROOTPERLEXT = $(ROOTPERLEXTDIR)/$(MODULE).so
ROOTPERLMOD64 = $(ROOTPERLMODDIR64)/$(MODULE).pm
ROOTPERLEXT64 = $(ROOTPERLEXTDIR64)/$(MODULE).so

XSUBPP = $(PERL) $(PERLDIR)/lib/ExtUtils/xsubpp \
        -typemap $(PERLDIR)/lib/ExtUtils/typemap

C99MODE = $(C99_ENABLE)

CERRWARN += -_cc=-erroff=E_ATTRIBUTE_UNKNOWN
CERRWARN += -_cc=-erroff=E_ATTRIBUTE_PARAM_UNDEFINED
```

```
**********************************************************
    2633 Mon Jul 18 14:42:49 2016
new/usr/src/cmd/perl/Makefile.targ
7170 ENABLE_PERL64 multiarch builds
reviewed by: Andrew Stormont <andyjstormont@gmail.com>
reviewed by: Lauri Tirkkonen <lotheac@iki.fi>
**********************************************************
#
# This file and its contents are supplied under the terms of the
# Common Development and Distribution License ("CDDL"), version 1.0.
# You may only use this file in accordance with the terms of version
# 1.0 of the CDDL.
#
# A full copy of the text of the CDDL should have accompanied this
# source.  A copy of the CDDL is also available via the Internet at
# http://www.illumos.org/license/CDDL.
#
#
# Copyright (c) 2014 Racktop Systems.
# Copyright 2015, OmniTI Computer Consulting, Inc. All rights reserved.
#

# Link against libc as per solaris specs
# Link against libc as perl solaris specs
$(PERLEXT):= LDLIBS += -lc
$(PERLEXT64):= LDLIBS += -lc

# Allow for undefined symbols satisfied by perl
$(PERLEXT):= ZDEFS =
$(PERLEXT64):= ZDEFS =

$(ROOTPERLEXT) := FILEMODE = 0555
$(ROOTPERLMOD) := FILEMODE = 0444
$(ROOTPERLEXT64) := FILEMODE = 0555
$(ROOTPERLMOD64) := FILEMODE = 0444

# CFLAGS for perl, specifically.
PCFLAGS= -DPERL_EUPXS_ALWAYS_EXPORT -D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64 \
        -DPERL_USE_SAFE_PUTENV -D_TS_ERRNO
PCFLAGS64= -m64 -DPERL_EUPXS_ALWAYS_EXPORT -D_LARGEFILE64_SOURCE \
        -DPERL_USE_SAFE_PUTENV -D_TS_ERRNO

$(MACH) $(MACH64):
$(MACH):
        $(INS.dir)

# Sorry about this...
# BUILD.SO doesn't have distinct versions like COMPILE.c/COMPILE64.c does.
# To that end - we're defining BUILD64.SO here.
BUILD64.SO=       $(CC) $(CFLAGS64) -o $@ $(GSHARED) $(DYNFLAGS) \
                $(PICS) $(EXTPICS) -L $(ROOTLIBDIR64) $(LDLIBS)

$(PERLEXT): $(MACH)/$(MODULE).o
        $(BUILD.SO) $(MACH)/$(MODULE).o

$(PERLEXT64): $(MACH64)/$(MODULE).o
        $(BUILD64.SO) $(MACH64)/$(MODULE).o

# NOTE: With later version of Perl, we need to define PERL_EUPXS_ALWAYS_EXPORT
# which is a backward-compatibility definition for the assumes-5.10 stuff here.
$(MACH)/$(MODULE).o: $(MACH)/$(MODULE).c
        $(COMPILE.c) $(PCFLAGS) $(C_PICFLAGS) -I$(PERLINCDIR) $< -o $@

$(MACH64)/$(MODULE).o: $(MACH64)/$(MODULE).c
        $(COMPILE64.c) $(PCFLAGS64) $(C_PICFLAGS) -I$(PERLINCDIR64) $< -o $@
```



```
$(MACH)/$(MODULE).c: $(MACH) $(MODULE).xs
        ISALIST=$(MACH) $(XSUBPP) $(XSUBPPFLAGS) $(MODULE).xs >$@
        $(PERLDIR)/bin/xsubpp $(XSUBPPFLAGS) $(MODULE).xs >$@

$(MACH64)/$(MODULE).c: $(MACH64) $(MODULE).xs
        ISALIST=$(MACH64) $(XSUBPP) $(XSUBPPFLAGS) $(MODULE).xs >$@

$(ROOTPERLMODDIR) $(ROOTPERLMODDIR64):
$(ROOTPERLMODDIR):
        $(INS.dir)

$(ROOTPERLMOD): $(ROOTPERLMODDIR) $(MODULE).pm
        $(RM) $@; $(INS) -s -m $(FILEMODE) -f $^

$(ROOTPERLMOD64): $(ROOTPERLMODDIR64) $(MODULE).pm
        $(RM) $@; $(INS) -s -m $(FILEMODE) -f $^

$(ROOTPERLEXTDIR) $(ROOTPERLEXTDIR64):
$(ROOTPERLEXTDIR):
        $(INS.dir)

$(ROOTPERLEXT): $(ROOTPERLEXTDIR) $(MACH)/$(MODULE).so
        $(RM) $@; $(INS) -s -m $(FILEMODE) -f $^

$(ROOTPERLEXT64): $(ROOTPERLEXTDIR64) $(MACH64)/$(MODULE).so
        $(RM) $@; $(INS) -s -m $(FILEMODE) -f $^
```

```
**********************************************************
     816 Mon Jul 18 14:42:50 2016
new/usr/src/cmd/perl/contrib/Sun/Solaris/BSM/Makefile
7170 ENABLE_PERL64 multiarch builds
reviewed by: Andrew Stormont <andyjstormont@gmail.com>
reviewed by: Lauri Tirkkonen <lotheac@iki.fi>
**********************************************************
```

```
#
# This file and its contents are supplied under the terms of the
# Common Development and Distribution License ("CDDL"), version 1.0.
# You may only use this file in accordance with the terms of version
# 1.0 of the CDDL.
#
# A full copy of the text of the CDDL should have accompanied this
# source.  A copy of the CDDL is also available via the Internet at
# http://www.illumos.org/license/CDDL.
#
#
# Copyright (c) 2014 Racktop Systems.
# Copyright 2014, OmniTI Computer Consulting, Inc. All rights reserved.
#

MODULE = _BSMparse

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

# Install module into arch independent directory
# Install module into arch independant directory
ROOTPERLMODDIR = $(ROOTPERLDIR)/lib/Sun/Solaris/BSM

include $(SRC)/cmd/perl/Makefile.targ

.KEEP_STATE:

all: $(PERLMOD)

install: $(ROOTPERLMOD)

clean clobber:
        $(RM) -r $(MACH) $(MACH64)
        $(RM) -r $(MACH)
```

```
**********************************************************
     894 Mon Jul 18 14:42:50 2016
new/usr/src/cmd/perl/contrib/Sun/Solaris/Intrs/Makefile
7170 ENABLE_PERL64 multiarch builds
reviewed by: Andrew Stormont <andyjstormont@gmail.com>
reviewed by: Lauri Tirkkonen <lotheac@iki.fi>
**********************************************************
```

```
#
# This file and its contents are supplied under the terms of the
# Common Development and Distribution License ("CDDL"), version 1.0.
# You may only use this file in accordance with the terms of version
# 1.0 of the CDDL.
#
# A full copy of the text of the CDDL should have accompanied this
# source.  A copy of the CDDL is also available via the Internet at
# http://www.illumos.org/license/CDDL.
#
#
# Copyright (c) 2014 Racktop Systems.
# Copyright 2014, OmniTI Computer Consulting, Inc. All rights reserved.
#

MODULE = Intrs

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

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

MAPFILES = mapfile-vers

include $(SRC)/cmd/perl/Makefile.targ

.KEEP_STATE:

all: $(PERLEXT) $(PERLMOD)
$(ENABLE_PERL64)all: $(PERLEXT64)

install: $(ROOTPERLEXT) $(ROOTPERLMOD)
$(ENABLE_PERL64)install: $(ROOTPERLEXT64) $(ROOTPERLMOD64)

clean clobber:
        $(RM) -r $(MACH) $(MACH64)
        $(RM) -r $(MACH)
```

```
**********************************************************
    1152 Mon Jul 18 14:42:50 2016
new/usr/src/cmd/perl/contrib/Sun/Solaris/Kstat/Makefile
7170 ENABLE_PERL64 multiarch builds
reviewed by: Andrew Stormont <andyjstormont@gmail.com>
reviewed by: Lauri Tirkkonen <lotheac@iki.fi>
**********************************************************
```

```
   1 #
   2 # This file and its contents are supplied under the terms of the
   3 # Common Development and Distribution License ("CDDL"), version 1.0.
   4 # You may only use this file in accordance with the terms of version
   5 # 1.0 of the CDDL.
   6 #
   7 # A full copy of the text of the CDDL should have accompanied this
   8 # source.  A copy of the CDDL is also available via the Internet at
   9 # http://www.illumos.org/license/CDDL.
  10 #
  11 # Copyright 2014 Gary Mills
  12 #
  13 # Copyright (c) 2014 Racktop Systems.
  14 # Copyright 2014, OmniTI Computer Consulting, Inc. All rights reserved.
  15 #

  17 MODULE = Kstat

  19 include $(SRC)/cmd/perl/Makefile.perl

  21 LDLIBS += -lkstat

  23 CERRWARN += -_gcc=-Wno-unused-value
  24 CERRWARN += -_gcc=-Wno-unused-variable

  26 # Additional include directories only needed for the SPARC platform

  28 $(SPARC_BLD) CPPFLAGS += -I$(ROOT)/usr/platform/sun4u/include \
  29                          -I$(ROOT)/usr/platform/sun4v/include

  31 MAPFILES = mapfile-vers

  33 include $(SRC)/cmd/perl/Makefile.targ

  35 .KEEP_STATE:

  37 all: $(PERLEXT) $(PERLMOD)
  38 $(ENABLE_PERL64)all: $(PERLEXT64)

  40 install: $(ROOTPERLEXT) $(ROOTPERLMOD)
  41 $(ENABLE_PERL64)install: $(ROOTPERLEXT64) $(ROOTPERLMOD64)

  43 clean clobber:
  44         $(RM) -r $(MACH) $(MACH64)
  41         $(RM) -r $(MACH)
```

```
**********************************************************
     934 Mon Jul 18 14:42:50 2016
new/usr/src/cmd/perl/contrib/Sun/Solaris/Lgrp/Makefile
7170 ENABLE_PERL64 multiarch builds
reviewed by: Andrew Stormont <andyjstormont@gmail.com>
reviewed by: Lauri Tirkkonen <lotheac@iki.fi>
**********************************************************
```

```
#
# This file and its contents are supplied under the terms of the
# Common Development and Distribution License ("CDDL"), version 1.0.
# You may only use this file in accordance with the terms of version
# 1.0 of the CDDL.
#
# A full copy of the text of the CDDL should have accompanied this
# source.  A copy of the CDDL is also available via the Internet at
# http://www.illumos.org/license/CDDL.
#
#
# Copyright (c) 2014 Racktop Systems.
# Copyright 2014, OmniTI Computer Consulting. All rights reserved.
#

MODULE = Lgrp

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

LDLIBS += -llgrp

CERRWARN += -_gcc=-Wno-type-limits

XSUBPPFLAGS = -typemap typemap

MAPFILES = mapfile-vers

include $(SRC)/cmd/perl/Makefile.targ

.KEEP_STATE:

all: $(PERLEXT) $(PERLMOD)
$(ENABLE_PERL64)all: $(PERLEXT64)

install: $(ROOTPERLEXT) $(ROOTPERLMOD)
$(ENABLE_PERL64)install: $(ROOTPERLEXT64) $(ROOTPERLMOD64)

clean clobber:
        $(RM) -r $(MACH) $(MACH64)
        $(RM) -r $(MACH)
```

****
805 Mon Jul 18 14:42:50 2016
new/usr/src/cmd/perl/contrib/Sun/Solaris/Pg/Makefile
7170 ENABLE_PERL64 multiarch builds
reviewed by: Andrew Stormont <andyjstormont@gmail.com>
reviewed by: Lauri Tirkkonen <lotheac@iki.fi>
****

```
#
# This file and its contents are supplied under the terms of the
# Common Development and Distribution License ("CDDL"), version 1.0.
# You may only use this file in accordance with the terms of version
# 1.0 of the CDDL.
#
# A full copy of the text of the CDDL should have accompanied this
# source.  A copy of the CDDL is also available via the Internet at
# http://www.illumos.org/license/CDDL.
#
#
# Copyright (c) 2014 Racktop Systems.
# Copyright 2014, OmniTI Computer Consulting, Inc. All rights reserved.
#

MODULE = Pg

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

# Install module into arch independent directory
# Install module into arch independant directory
ROOTPERLMODDIR = $(ROOTPERLDIR)/lib/Sun/Solaris

include $(SRC)/cmd/perl/Makefile.targ

.KEEP_STATE:

all: $(PERLMOD)

install: $(ROOTPERLMOD)

clean clobber:
        $(RM) -r $(MACH) $(MACH64)
        $(RM) -r $(MACH)
```

```
**********************************************************
     955 Mon Jul 18 14:42:51 2016
new/usr/src/cmd/perl/contrib/Sun/Solaris/Project/Makefile
7170 ENABLE_PERL64 multiarch builds
reviewed by: Andrew Stormont <andyjstormont@gmail.com>
reviewed by: Lauri Tirkkonen <lotheac@iki.fi>
**********************************************************
```

```
   1 #
   2 # This file and its contents are supplied under the terms of the
   3 # Common Development and Distribution License ("CDDL"), version 1.0.
   4 # You may only use this file in accordance with the terms of version
   5 # 1.0 of the CDDL.
   6 #
   7 # A full copy of the text of the CDDL should have accompanied this
   8 # source.  A copy of the CDDL is also available via the Internet at
   9 # http://www.illumos.org/license/CDDL.
  10 #
  11 #
  12 # Copyright (c) 2014 Racktop Systems.
  13 # Copyright 2014, OmniTI Computer Consulting, Inc. All rights reserved.
  14 #

  16 MODULE = Project

  18 include $(SRC)/cmd/perl/Makefile.perl

  20 LDLIBS += -lproject -lgen

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

  24 XSUBPPFLAGS = -typemap typemap

  26 MAPFILES = mapfile-vers

  28 include $(SRC)/cmd/perl/Makefile.targ

  30 .KEEP_STATE:

  32 all: $(PERLEXT) $(PERLMOD)
  33 $(ENABLE_PERL64)all: $(PERLEXT64)

  35 install: $(ROOTPERLEXT) $(ROOTPERLMOD)
  36 $(ENABLE_PERL64)install: $(ROOTPERLEXT64) $(ROOTPERLMOD64)

  38 clean clobber:
  39         $(RM) -r $(MACH) $(MACH64)
  36         $(RM) -r $(MACH)
```

```
**********************************************************
     885 Mon Jul 18 14:42:51 2016
new/usr/src/cmd/perl/contrib/Sun/Solaris/Task/Makefile
7170 ENABLE_PERL64 multiarch builds
reviewed by: Andrew Stormont <andyjstormont@gmail.com>
reviewed by: Lauri Tirkkonen <lotheac@iki.fi>
**********************************************************
```

```
#
# This file and its contents are supplied under the terms of the
# Common Development and Distribution License ("CDDL"), version 1.0.
# You may only use this file in accordance with the terms of version
# 1.0 of the CDDL.
#
# A full copy of the text of the CDDL should have accompanied this
# source.  A copy of the CDDL is also available via the Internet at
# http://www.illumos.org/license/CDDL.
#
#
# Copyright (c) 2014 Racktop Systems.
# Copyright 2014, OmniTI Computer Consulting, Inc. All rights reserved.
#

MODULE = Task

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

XSUBPPFLAGS = -typemap typemap

MAPFILES = mapfile-vers

include $(SRC)/cmd/perl/Makefile.targ

.KEEP_STATE:

all: $(PERLEXT) $(PERLMOD)
$(ENABLE_PERL64)all: $(PERLEXT64)

install: $(ROOTPERLEXT) $(ROOTPERLMOD)
$(ENABLE_PERL64)install: $(ROOTPERLEXT64) $(ROOTPERLMOD64)

clean clobber:
        $(RM) -r $(MACH) $(MACH64)
        $(RM) -r $(MACH)
```

```
**********************************************************
     878 Mon Jul 18 14:42:51 2016
new/usr/src/cmd/perl/contrib/Sun/Solaris/Utils/Makefile
7170 ENABLE_PERL64 multiarch builds
reviewed by: Andrew Stormont <andyjstormont@gmail.com>
reviewed by: Lauri Tirkkonen <lotheac@iki.fi>
**********************************************************
```

```
   1 #
   2 # This file and its contents are supplied under the terms of the
   3 # Common Development and Distribution License ("CDDL"), version 1.0.
   4 # You may only use this file in accordance with the terms of version
   5 # 1.0 of the CDDL.
   6 #
   7 # A full copy of the text of the CDDL should have accompanied this
   8 # source.  A copy of the CDDL is also available via the Internet at
   9 # http://www.illumos.org/license/CDDL.
  10 #
  11 #
  12 # Copyright (c) 2014 Racktop Systems.
  13 # Copyright 2014, OmniTI Computer Consulting, Inc. All rights reserved.
  14 #

  16 MODULE = Utils

  18 include $(SRC)/cmd/perl/Makefile.perl

  20 LDLIBS += -lgen -lintl

  22 MAPFILES = mapfile-vers

  24 include $(SRC)/cmd/perl/Makefile.targ

  26 .KEEP_STATE:

  28 all: $(PERLEXT) $(PERLMOD)
  29 $(ENABLE_PERL64)all: $(PERLEXT64)

  31 install: $(ROOTPERLEXT) $(ROOTPERLMOD)
  32 $(ENABLE_PERL64)install: $(ROOTPERLEXT64) $(ROOTPERLMOD64)

  34 clean clobber:
  35         $(RM) -r $(MACH) $(MACH64)
  33         $(RM) -r $(MACH)
```

```
**********************************************************
   25320 Mon Jul 18 14:42:51 2016
new/usr/src/pkg/Makefile
7170 ENABLE_PERL64 multiarch builds
reviewed by: Andrew Stormont <andyjstormont@gmail.com>
reviewed by: Lauri Tirkkonen <lotheac@iki.fi>
**********************************************************
```

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

  22 #
  23 # Copyright (c) 2010, Oracle and/or its affiliates. All rights reserved.
  24 # Copyright 2015, OmniTI Computer Consulting, Inc. All rights reserved.
  25 # Copyright 2015 Igor Kozhukhov <ikozhukhov@gmail.com>
  26 #

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

  31 #
  32 # Make sure we're getting a consistent execution environment for the
  33 # embedded scripts.
  34 #
  35 SHELL= /usr/bin/ksh93

  37 #
  38 # To suppress package dependency generation on any system, regardless
  39 # of how it was installed, set SUPPRESSPKGDEP=true in the build
  40 # environment.
  41 #
  42 SUPPRESSPKGDEP= false

  44 #
  45 # Comment this line out or set "PKGDEBUG=" in your build environment
  46 # to get more verbose output from the make processes in usr/src/pkg
  47 #
  48 PKGDEBUG= @

  50 #
  51 # Cross platform packaging notes
  52 #
  53 # By default, we package the proto area from the same architecture as
  54 # the packaging build.  In other words, if you're running nightly or
  55 # bldenv on an x86 platform, it will take objects from the x86 proto
  56 # area and use them to create x86 repositories.
  57 #
  58 # If you want to create repositories for an architecture that's
  59 # different from $(uname -p), you do so by setting PKGMACH in your
```



```
  60 # build environment.
  61 #
  62 # For this to work correctly, the following must all happen:
  63 #
  64 #   1. You need the desired proto area, which you can get either by
  65 #      doing a gatekeeper-style build with the -U option to
  66 #      nightly(1), or by using rsync.  If you don't do this, you will
  67 #      get packaging failures building all packages, because pkgsend
  68 #      is unable to find the required binaries.
  69 #   2. You need the desired tools proto area, which you can get in the
  70 #      same ways as the normal proto area.  If you don't do this, you
  71 #      will get packaging failures building onbld, because pkgsend is
  72 #      unable to find the tools binaries.
  73 #   3. The remainder of this Makefile should never refer directly to
  74 #      $(MACH).  Instead, $(PKGMACH) should be used whenever an
  75 #      architecture-specific path or token is needed.  If this is done
  76 #      incorrectly, then packaging will fail, and you will see the
  77 #      value of $(uname -p) instead of the value of $(PKGMACH) in the
  78 #      commands that fail.
  79 #   4. Each time a rule in this Makefile invokes $(MAKE), it should
  80 #      pass PKGMACH=$(PKGMACH) explicitly on the command line.  If
  81 #      this is done incorrectly, then packaging will fail, and you
  82 #      will see the value of $(uname -p) instead of the value of
  83 #      $(PKGMACH) in the commands that fail.
  84 #
  85 # Refer also to the convenience targets defined later in this
  86 # Makefile.
  87 #
  88 PKGMACH=        $(MACH)

  90 #
  91 # ROOT, TOOLS_PROTO, and PKGARCHIVE should be set by nightly or
  92 # bldenv.  These macros translate them into terms of $PKGMACH, instead
  93 # of $ARCH.
  94 #
  95 PKGROOT.cmd=    print $(ROOT) | sed -e s:/root_$(MACH):/root_$(PKGMACH):
  96 PKGROOT=        $(PKGROOT.cmd:sh)
  97 TOOLSROOT.cmd=  print $(TOOLS_PROTO) | sed -e s:/root_$(MACH):/root_$(PKGMACH):
  98 TOOLSROOT=      $(TOOLSROOT.cmd:sh)
  99 PKGDEST.cmd=    print $(PKGARCHIVE) | sed -e s:/$(MACH)/:/$(PKGMACH)/:
 100 PKGDEST=        $(PKGDEST.cmd:sh)

 102 EXCEPTIONS= packaging

 104 PKGMOGRIFY= pkgmogrify

 106 #
 107 # Always build the redistributable repository, but only build the
 108 # nonredistributable bits if we have access to closed source.
 109 #
 110 # Some objects that result from the closed build are still
 111 # redistributable, and should be packaged as part of an open-only
 112 # build.  Access to those objects is provided via the closed-bins
 113 # tarball.  See usr/src/tools/scripts/bindrop.sh for details.
 114 #
 115 REPOS= redist

 117 #
 118 # The packages directory will contain the processed manifests as
 119 # direct build targets and subdirectories for package metadata extracted
 120 # incidentally during manifest processing.
 121 #
 122 # Nothing underneath $(PDIR) should ever be managed by SCM.
 123 #
 124 PDIR= packages.$(PKGMACH)
```

```
126 #
127 # 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         ENABLE_PERL64=$(ENABLE_PERL64) \
182         PERL_ARCH=$(PERL_ARCH) \
183         PERL_ARCH64=$(PERL_ARCH64) \
184         PERL_VERSION=$(PERL_VERSION) \
185         PERL_PKGVERS=$(PERL_PKGVERS)

187 PKGDEP_TOKENS_i386= \
188         'PLATFORM=i86hvm' \
189         'PLATFORM=i86pc' \
190         'PLATFORM=i86xpv' \
191         'ISALIST=amd64' \
```



```
192         'ISALIST=i386'
193 PKGDEP_TOKENS_sparc= \
194         'PLATFORM=sun4u' \
195         'PLATFORM=sun4v' \
196         'ISALIST=sparcv9' \
197         'ISALIST=sparc'
198 PKGDEP_TOKENS= $(PKGDEP_TOKENS_$(PKGMACH))

200 #
201 # The package lists are generated with $(PKGDEP_TYPE) as their
202 # dependency types, so that they can be included by either an
203 # incorporation or a group package.
204 #
205 $(PDIR)/osnet-redist.mog := PKGDEP_TYPE= require
206 $(PDIR)/osnet-incorporation.mog:= PKGDEP_TYPE= incorporate

208 PKGDEP_INCORP= \
209         depend fmri=consolidation/osnet/osnet-incorporation type=require

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

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

238 # Conditionally add back lp-ipp
239 $(ENABLE_IPP_PRINTING) MANIFESTS += print-lp-ipp-ipp-listener.mf

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

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

254 #
255 # Root of pkg image to use for dependency resolution
256 # Normally / on the machine used to build the binaries
257 #
```

```
258 PKGDEP_RESOLVE_IMAGE = /

260 #
261 # For each package, we determine the target repository based on
262 # manifest-embedded metadata.  Because we make that determination on
263 # the fly, the publication target cannot be expressed as a
264 # subdirectory inside the unknown-by-the-makefile target repository.
265 #
266 # In order to limit the target set to real files in known locations,
267 # we use a ".pub" file in $(PDIR) for each processed manifest, regardless
268 # of content or target repository.
269 #
270 PUB_PKGS= $(SYNTH_PKGS:%=$(PDIR)/%.pub) $(PKGS:%=$(PDIR)/%.pub)

272 #
273 # Any given repository- and status-specific package list may be empty,
274 # but we can only determine that dynamically, so we always generate all
275 # lists for each repository we're building.
276 #
277 # The meanings of each package status are as follows:
278 #
279 #       PKGSTAT         meaning
280 #       ----------      ------------------------------------------------------
281 #       noincorp        Do not include in incorporation or group package
282 #       obsolete        Include in incorporation, but not group package
283 #       renamed         Include in incorporation, but not group package
284 #       current         Include in incorporation and group package
285 #
286 # Since the semantics of the "noincorp" package status dictate that
287 # such packages are not included in the incorporation or group packages,
288 # there is no need to build noincorp package lists.
289 #
290 PKGLISTS= \
291         $(REPOS:%=$(PDIR)/packages.%.current) \
292         $(REPOS:%=$(PDIR)/packages.%.renamed) \
293         $(REPOS:%=$(PDIR)/packages.%.obsolete)

295 .KEEP_STATE:

297 .PARALLEL: $(PKGS) $(PROC_PKGS) $(DEP_PKGS) \
298         $(PROC_SYNTH_PKGS) $(DEP_SYNTH_PKGS) $(PUB_PKGS)

300 #
301 # For a single manifest, the dependency chain looks like this:
302 #
303 #       raw manifest (mypkg.mf)
304 #               |
305 #               | use pkgmogrify to process raw manifest
306 #               |
307 #       processed manifest (mypkg.mog)
308 #               |
309 #          *    | use pkgdepend generate to generate dependencies
310 #               |
311 #       manifest with TBD dependencies (mypkg.dep)
312 #               |
313 #          %    | use pkgdepend resolve to resolve dependencies
314 #               |
315 #       manifest with dependencies resolved (mypkg.res)
316 #               |
317 #               | use pkgsend to publish the package
318 #               |
319 #       placeholder to indicate successful publication (mypkg.pub)
320 #
321 # * This may be suppressed via SUPPRESSPKGDEP.  The resulting
322 #   packages will install correctly, but care must be taken to
323 #   install all dependencies, because pkg will not have the input
```



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

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

373 all: $(ALL_TARGETS)

375 #
376 # This will build the directory to contain the processed manifests
377 # and the metadata symlinks.
378 #
379 $(PDIR):
380         @print "Creating $(@)"
381         $(PKGDEBUG)$(INS.dir)

383 #
384 # This rule resolves dependencies across all published manifests.
385 #
386 # We shouldn't have to ignore the error from pkgdepend, but until
387 # 16012 and its dependencies are resolved, pkgdepend will always exit
388 # with an error.
389 #
```

```
390 $(PDIR)/gendeps: $(DEP_SYNTH_PKGS) $(DEP_PKGS)
391         -$(PKGDEBUG)if [ "$(SUPPRESSPKGDEP)" = "true" ]; then \
392                 print "Suppressing dependency resolution"; \
393                 for p in $(DEP_PKGS:%.dep=%); do \
394                         $(CP) $$p.dep $$p.res; \
395                 done; \
396         else \
397                 print "Resolving dependencies"; \
398                 pkgdepend -R $(PKGDEP_RESOLVE_IMAGE) resolve \
399                     -m $(DEP_SYNTH_PKGS) $(DEP_PKGS); \
400                 for p in $(DEP_SYNTH_PKGS:%.dep=%) $(DEP_PKGS:%.dep=%); do \
401                         if [ "$$(print $$p.metadata.*)" = \
402                              "$$(print $$p.metadata.noincorp.*)" ]; \
403                         then \
404                                 print "Removing dependency versions from $$p"; \
405                                 $(PKGMOGRIFY) $(PKGMOG_VERBOSE) \
406                                     -O $$p.res -I transforms \
407                                     strip_versions $$p.dep.res; \
408                                 $(RM) $$p.dep.res; \
409                         else \
410                                 $(MV) $$p.dep.res $$p.res; \
411                         fi; \
412                 done; \
413         fi
414         $(PKGDEBUG)$(TOUCH) $(@)

416 install: $(ALL_TARGETS) repository-metadata

418 repository-metadata: publish_pkgs
419         $(PKGDEBUG)for r in $(REPOS); do \
420                 pkgrepo refresh -s $(PKGDEST)/repo.$$r; \
421         done

423 #
424 # Since we create zero-length processed manifests for a graceful abort
425 # from pkgmogrify, we need to detect that here and make no effort to
426 # publish the package.
427 #
428 # For all other packages, we publish them regardless of status.  We
429 # derive the target repository as a component of the metadata-derived
430 # symlink for each package.
431 #
432 publish_pkgs: $(REPOS:%=$(PKGDEST)/repo.%) $(PDIR)/gendeps .WAIT $(PUB_PKGS)

434 #
435 # Before publishing, we want to pull the license files from $CODEMGR_WS
436 # into the proto area.  This allows us to NOT pass $SRC (or
437 # $CODEMGR_WS) as a basedir for publication.
438 #
439 $(PUB_PKGS): stage-licenses

441 #
442 # Initialize the empty on-disk repositories
443 #
444 $(REPOS:%=$(PKGDEST)/repo.%):
445         @print "Initializing $(@F)"
446         $(PKGDEBUG)$(INS.dir)
447         $(PKGDEBUG)pkgsend -s file://$(@) create-repository \
448                 --set-property publisher.prefix=$(PKGPUBLISHER)

450 #
451 # rule to process real manifests
452 #
453 # To allow redistributability and package status to change, we must
454 # remove not only the actual build target (the processed manifest), but
455 # also the incidental ones (the metadata-derived symlinks).
```



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

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

508 $(PDIR)/%.mog: manifests/%.mf
509         @print "Processing manifest $(<F)"
510         @env PKGFMT_OUTPUT=v1 pkgfmt -c $<
511         $(PKGDEBUG)$(RM) $(@) $(@:%.mog=%) $(@:%.mog=%.nodepend) \
512                 $(@:%.mog=%.lics) $(PDIR)/$(@F:%.mog=%).metadata.* $(@).vars
513         $(PKGDEBUG)$(PKGMOGRIFY) $(PKGMOG_VERBOSE) $(PM_INC:%= -I %) \
514                 $(PKGMOG_DEFINES:%=-D %) -P $(@).vars -O $(@) \
515                 $(<) $(PM_TRANSFORMS)
516         $(PKGDEBUG)eval REPO=redist PKGSTAT=current NODEPEND=$(SUPPRESSPKGDEP) \
517                 '$(CAT) -s $(@).vars'; \
518         if [ -f $(@) ]; then \
519                 if [ "$$NODEPEND" != "false" ]; then \
520                         $(TOUCH) $(@:%.mog=%.nodepend); \
521                 fi; \
```

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

534 $(PDIR)/%.dep: $(PDIR)/%.mog
535         @print "Generating dependencies for $(<F)"
536         $(PKGDEBUG)$(RM) $(@)
537         $(PKGDEBUG)if [ ! -f $(@:%.dep=%.nodepend) ]; then \
538                 pkgdepend generate -m $(PKGDEP_TOKENS:%=-D %) $(<) \
539                         $(PKGROOT) > $(@); \
540         else \
541                 $(CP) $(<) $(@); \
542         fi

544 #
545 # The full chain implies that there should be a .dep.res suffix rule,
546 # but dependency generation is done on a set of manifests, rather than
547 # on a per-manifest basis.  Instead, see the gendeps rule above.
548 #

550 $(PDIR)/%.pub: $(PDIR)/%.res
551         $(PKGDEBUG)m=$$(basename $(@:%.pub=%).metadata.*); \
552         r=$${m#$(@F:%.pub=%.metadata.)+(?).}; \
553         if [ -s $(<) ]; then \
554                 print "Publishing $(@F:%.pub=%) to $$r repository"; \
555                 pkgsend -s file://$(PKGDEST)/repo.$$r publish \
556                     -d $(PKGROOT) -d $(TOOLSROOT) \
557                     -d license_files -d $(PKGROOT)/licenses \
558                     --fmri-in-manifest --no-index --no-catalog $(<) \
559                     > /dev/null; \
560         fi; \
561         $(TOUCH) $(@);

563 #
564 # rule to build the synthetic manifests
565 #
566 # This rule necessarily has PKGDEP_TYPE that changes according to
567 # the specific synthetic manifest.  Rather than escape command
568 # dependency checking for the real manifest processing, or failing to
569 # express the (indirect) dependency of synthetic manifests on real
570 # manifests, we simply split this rule out from the one above.
571 #
572 # The implementation notes from the previous rule are applicable
573 # here, too.
574 #
575 $(PROC_SYNTH_PKGS): $(PKGLISTS) $$(@F:%.mog=%.mf)
576         @print "Processing synthetic manifest $(@F:%.mog=%.mf)"
577         $(PKGDEBUG)$(RM) $(@) $(PDIR)/$(@F:%.mog=%).metadata.* $(@).vars
578         $(PKGDEBUG)$(PKGMOGRIFY) $(PKGMOG_VERBOSE) -I transforms -I $(PDIR) \
579                 $(PKGMOG_DEFINES:%=-D %) -D PKGDEP_TYPE=$(PKGDEP_TYPE) \
580                 -P $(@).vars -O $(@) $(@F:%.mog=%.mf) \
581                 $(PM_TRANSFORMS) synthetic
582         $(PKGDEBUG)eval REPO=redist PKGSTAT=current '$(CAT) -s $(@).vars'; \
583         if [ -f $(@) ]; then \
584                 $(LN) -s $(@F) \
585                         $(PDIR)/$(@F:%.mog=%).metadata.$$PKGSTAT.$$REPO; \
586         else \
587                 $(TOUCH) $(@); \
```



```
588         fi
589         $(PKGDEBUG)$(RM) $(@).vars

591 $(DEP_SYNTH_PKGS): $$(@:%.dep=%.mog)
592         @print "Skipping dependency generation for $(@F:%.dep=%)"
593         $(PKGDEBUG)$(CP) $(@:%.dep=%.mog) $(@)

595 clean:

597 clobber: clean
598         $(RM) -r $(CLOBBERFILES)

600 #
601 # This rule assumes that all links in the $PKGSTAT directories
602 # point to valid manifests, and will fail the make run if one
603 # does not contain an fmri.
604 #
605 # We do this in the BEGIN action instead of using pattern matching
606 # because we expect the fmri to be at or near the first line of each input
607 # file, and this way lets us avoid reading the rest of the file after we
608 # find what we need.
609 #
610 # We keep track of a failure to locate an fmri, so we can fail the
611 # make run, but we still attempt to process each package in the
612 # repo/pkgstat-specific subdir, in hopes of maybe giving some
613 # additional useful info.
614 #
615 # The protolist is used for bfu archive creation, which may be invoked
616 # interactively by the user.  Both protolist and PKGLISTS targets
617 # depend on $(PROC_PKGS), but protolist builds them recursively.
618 # To avoid collisions, we insert protolist into the dependency chain
619 # here.  This has two somewhat subtle benefits: it allows bfu archive
620 # creation to work correctly, even when -a was not part of NIGHTLY_OPTIONS,
621 # and it ensures that a protolist file here will always correspond to the
622 # contents of the processed manifests, which can vary depending on build
623 # environment.
624 #
625 $(PKGLISTS): $(PROC_PKGS)
626         $(PKGDEBUG)sdotr=$(@F:packages.%=%); \
627         r=$${sdotr%.+(?)}; s=$${sdotr#+(?).}; \
628         print "Generating $$r $$s package list"; \
629         $(RM) $(@); $(TOUCH) $(@); \
630         $(AWK) 'BEGIN { \
631                 if (ARGC < 2) { \
632                         exit; \
633                 } \
634                 retcode = 0; \
635                 for (i = 1; i < ARGC; i++) { \
636                         do { \
637                                 e = getline f < ARGV[i]; \
638                         } while ((e == 1) && (f !~ /name=pkg.fmri/)); \
639                         close(ARGV[i]); \
640                         if (e == 1) { \
641                                 l = split(f, a, "="); \
642                                 print "depend fmri=" a[l], \
643                                         "type=$$(PKGDEP_TYPE)"; \
644                         } else { \
645                                 print "no fmri in " ARGV[i] >> "/dev/stderr"; \
646                                 retcode = 2; \
647                         } \
648                 } \
649                 exit retcode; \
650         }' 'find $(PDIR) -type l -a \( $(PKGS:%=-name %.metadata.$$s.$$r -o) \
651                 -name NOSUCHFILE \)' >> $(@)

653 #
```

```
654 # rules to validate proto area against manifests, check for safe
655 # file permission modes, and generate a faux proto list
656 #
657 # For the check targets, the dependencies on $(PROC_PKGS) is specified
658 # as a subordinate make process in order to suppress output.
659 #
660 makesilent:
661         @$(MAKE) -e $(PROC_PKGS) PKGMACH=$(PKGMACH) \
662                 SUPPRESSPKGDEP=$(SUPPRESSPKGDEP) > /dev/null

664 #
665 # The .lics files were created during pkgmogrification, and list the
666 # set of licenses to pull from $SRC for each package.  Because
667 # licenses may be duplicated between packages, we uniquify them as
668 # well as aggregating them here.
669 #
670 license-list: makesilent
671         $(PKGDEBUG)( for l in 'cat $(PROC_PKGS:%.mog=%.lics)'; \
672                 do print $$l; done ) | sort -u > $@

674 #
675 # Staging the license and description files in the proto area allows
676 # us to do proper unreferenced file checking of both license and
677 # description files without blanket exceptions, and to pull license
678 # content without reference to $CODEMGR_WS during publication.
679 #
680 stage-licenses: license-list FRC
681         $(PKGDEBUG)$(MAKE) -e -f Makefile.lic \
682                 PKGDEBUG=$(PKGDEBUG) LICROOT=$(PKGROOT)/licenses \
683                 '$(AWK) '{ \
684                         print "$(PKGROOT)/licenses/" $$0; \
685                         print "$(PKGROOT)/licenses/" $$0 ".descrip"; \
686                 }' license-list' > /dev/null;

688 protocmp: makesilent
689         @validate_pkg -a $(PKGMACH) -v \
690                 $(EXCEPTIONS:%=-e $(CODEMGR_WS)/exception_lists/%) \
691                 -m $(PDIR) -p $(PKGROOT) -p $(TOOLSROOT)

693 pmodes: makesilent
694         @validate_pkg -a $(PKGMACH) -M -m $(PDIR) \
695                 -e $(CODEMGR_WS)/exception_lists/pmodes

697 check: protocmp pmodes

699 protolist: proto_list_$(PKGMACH)

701 proto_list_$(PKGMACH): $(PROC_PKGS)
702         @validate_pkg -a $(PKGMACH) -L -m $(PDIR) > $(@)

704 $(PROC_PKGS): $(PDIR)

706 #
707 # This is a convenience target to allow package names to function as
708 # build targets.  Generally, using it is only useful when iterating on
709 # development of a manifest.
710 #
711 # When processing a manifest, use the basename (without extension) of
712 # the package.  When publishing, use the basename with a ".pub"
713 # extension.
714 #
715 # Other than during manifest development, the preferred usage is to
716 # avoid these targets and override PKGS on the make command line and
717 # use the provided all and install targets.
718 #
719 $(PKGS) $(SYNTH_PKGS): $(PDIR)/$$(@:%=%.mog)
```



```
721 $(PKGS:%=%.pub) $(SYNTH_PKGS:%=%.pub): $(PDIR)/$$(@)

723 #
724 # This is a convenience target to resolve dependencies without publishing
725 # packages.
726 #
727 gendeps: $(PDIR)/gendeps

729 #
730 # These are convenience targets for cross-platform packaging.  If you
731 # want to build any of "the normal" targets for a different
732 # architecture, simply use "arch/target" as your build target.
733 #
734 # Since the most common use case for this is "install," the architecture
735 # specific install targets have been further abbreviated to elide "/install."
736 #
737 i386/% sparc/%:
738         $(MAKE) -e $(@F) PKGMACH=$(@D) SUPPRESSPKGDEP=$(SUPPRESSPKGDEP)

740 i386 sparc: $$(@)/install

742 FRC:
```

```
**********************************************************
    6277 Mon Jul 18 14:42:52 2016
new/usr/src/pkg/manifests/runtime-perl-module-sun-solaris.mf
7170 ENABLE_PERL64 multiarch builds
reviewed by: Andrew Stormont <andyjstormont@gmail.com>
reviewed by: Lauri Tirkkonen <lotheac@iki.fi>
**********************************************************
```

```
#
# 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 (c) 2014 Racktop Systems.
# Copyright 2015, OmniTI Computer Consulting, Inc. All rights reserved.
#

<transform file path=.*\.(pm|bs) -> default mode 0444>
<transform file path=.*\.so -> default mode 0555>
set name=pkg.fmri \
    value=pkg:/runtime/perl$(PERL_PKGVERS)/module/sun-solaris@0.5.11,$(PKGVERS_B
set name=pkg.summary value="Perl $(PERL_VERSION) Sun::Solaris Modules"
set name=info.classification \
    value=org.opensolaris.category.2008:Development/Perl
set name=variant.arch value=$(ARCH)
dir path=usr group=sys
dir path=usr/perl5
dir path=usr/perl5/$(PERL_VERSION)
dir path=usr/perl5/$(PERL_VERSION)/lib
dir path=usr/perl5/$(PERL_VERSION)/lib/$(PERL_ARCH)
dir path=usr/perl5/$(PERL_VERSION)/lib/$(PERL_ARCH)/Sun
dir path=usr/perl5/$(PERL_VERSION)/lib/$(PERL_ARCH)/Sun/Solaris
dir path=usr/perl5/$(PERL_VERSION)/lib/$(PERL_ARCH)/auto
dir path=usr/perl5/$(PERL_VERSION)/lib/$(PERL_ARCH)/auto/Sun
dir path=usr/perl5/$(PERL_VERSION)/lib/$(PERL_ARCH)/auto/Sun/Solaris
dir path=usr/perl5/$(PERL_VERSION)/lib/$(PERL_ARCH)/auto/Sun/Solaris/Intrs
dir path=usr/perl5/$(PERL_VERSION)/lib/$(PERL_ARCH)/auto/Sun/Solaris/Kstat
dir path=usr/perl5/$(PERL_VERSION)/lib/$(PERL_ARCH)/auto/Sun/Solaris/Lgrp
dir path=usr/perl5/$(PERL_VERSION)/lib/$(PERL_ARCH)/auto/Sun/Solaris/Project
dir path=usr/perl5/$(PERL_VERSION)/lib/$(PERL_ARCH)/auto/Sun/Solaris/Task
dir path=usr/perl5/$(PERL_VERSION)/lib/$(PERL_ARCH)/auto/Sun/Solaris/Utils
$(ENABLE_PERL64)dir path=usr/perl5/$(PERL_VERSION)/lib/$(PERL_ARCH64)
$(ENABLE_PERL64)dir path=usr/perl5/$(PERL_VERSION)/lib/$(PERL_ARCH64)/Sun
$(ENABLE_PERL64)dir \
    path=usr/perl5/$(PERL_VERSION)/lib/$(PERL_ARCH64)/Sun/Solaris
$(ENABLE_PERL64)dir path=usr/perl5/$(PERL_VERSION)/lib/$(PERL_ARCH64)/auto
$(ENABLE_PERL64)dir path=usr/perl5/$(PERL_VERSION)/lib/$(PERL_ARCH64)/auto/Sun
$(ENABLE_PERL64)dir \
    path=usr/perl5/$(PERL_VERSION)/lib/$(PERL_ARCH64)/auto/Sun/Solaris
```



```
$(ENABLE_PERL64)dir \
    path=usr/perl5/$(PERL_VERSION)/lib/$(PERL_ARCH64)/auto/Sun/Solaris/Intrs
$(ENABLE_PERL64)dir \
    path=usr/perl5/$(PERL_VERSION)/lib/$(PERL_ARCH64)/auto/Sun/Solaris/Kstat
$(ENABLE_PERL64)dir \
    path=usr/perl5/$(PERL_VERSION)/lib/$(PERL_ARCH64)/auto/Sun/Solaris/Lgrp
$(ENABLE_PERL64)dir \
    path=usr/perl5/$(PERL_VERSION)/lib/$(PERL_ARCH64)/auto/Sun/Solaris/Project
$(ENABLE_PERL64)dir \
    path=usr/perl5/$(PERL_VERSION)/lib/$(PERL_ARCH64)/auto/Sun/Solaris/Task
$(ENABLE_PERL64)dir \
    path=usr/perl5/$(PERL_VERSION)/lib/$(PERL_ARCH64)/auto/Sun/Solaris/Utils
dir path=usr/perl5/$(PERL_VERSION)/lib/Sun
dir path=usr/perl5/$(PERL_VERSION)/lib/Sun/Solaris
dir path=usr/perl5/$(PERL_VERSION)/lib/Sun/Solaris/BSM
dir path=usr/share/man
dir path=usr/share/man/man3perl
file path=usr/perl5/$(PERL_VERSION)/lib/$(PERL_ARCH)/Sun/Solaris/Intrs.pm
file path=usr/perl5/$(PERL_VERSION)/lib/$(PERL_ARCH)/Sun/Solaris/Kstat.pm
file path=usr/perl5/$(PERL_VERSION)/lib/$(PERL_ARCH)/Sun/Solaris/Lgrp.pm
file path=usr/perl5/$(PERL_VERSION)/lib/$(PERL_ARCH)/Sun/Solaris/Project.pm
file path=usr/perl5/$(PERL_VERSION)/lib/$(PERL_ARCH)/Sun/Solaris/Task.pm
file path=usr/perl5/$(PERL_VERSION)/lib/$(PERL_ARCH)/Sun/Solaris/Utils.pm
file \
    path=usr/perl5/$(PERL_VERSION)/lib/$(PERL_ARCH)/auto/Sun/Solaris/Intrs/Intrs
file \
    path=usr/perl5/$(PERL_VERSION)/lib/$(PERL_ARCH)/auto/Sun/Solaris/Kstat/Kstat
file \
    path=usr/perl5/$(PERL_VERSION)/lib/$(PERL_ARCH)/auto/Sun/Solaris/Lgrp/Lgrp.s
file \
    path=usr/perl5/$(PERL_VERSION)/lib/$(PERL_ARCH)/auto/Sun/Solaris/Project/Pro
file \
    path=usr/perl5/$(PERL_VERSION)/lib/$(PERL_ARCH)/auto/Sun/Solaris/Task/Task.s
file \
    path=usr/perl5/$(PERL_VERSION)/lib/$(PERL_ARCH)/auto/Sun/Solaris/Utils/Utils
$(ENABLE_PERL64)file \
    path=usr/perl5/$(PERL_VERSION)/lib/$(PERL_ARCH64)/Sun/Solaris/Intrs.pm
$(ENABLE_PERL64)file \
    path=usr/perl5/$(PERL_VERSION)/lib/$(PERL_ARCH64)/Sun/Solaris/Kstat.pm
$(ENABLE_PERL64)file \
    path=usr/perl5/$(PERL_VERSION)/lib/$(PERL_ARCH64)/Sun/Solaris/Lgrp.pm
$(ENABLE_PERL64)file \
    path=usr/perl5/$(PERL_VERSION)/lib/$(PERL_ARCH64)/Sun/Solaris/Project.pm
$(ENABLE_PERL64)file \
    path=usr/perl5/$(PERL_VERSION)/lib/$(PERL_ARCH64)/Sun/Solaris/Task.pm
$(ENABLE_PERL64)file \
    path=usr/perl5/$(PERL_VERSION)/lib/$(PERL_ARCH64)/Sun/Solaris/Utils.pm
$(ENABLE_PERL64)file \
    path=usr/perl5/$(PERL_VERSION)/lib/$(PERL_ARCH64)/auto/Sun/Solaris/Intrs/Int
$(ENABLE_PERL64)file \
    path=usr/perl5/$(PERL_VERSION)/lib/$(PERL_ARCH64)/auto/Sun/Solaris/Kstat/Kst
$(ENABLE_PERL64)file \
    path=usr/perl5/$(PERL_VERSION)/lib/$(PERL_ARCH64)/auto/Sun/Solaris/Lgrp/Lgrp
$(ENABLE_PERL64)file \
    path=usr/perl5/$(PERL_VERSION)/lib/$(PERL_ARCH64)/auto/Sun/Solaris/Project/P
$(ENABLE_PERL64)file \
    path=usr/perl5/$(PERL_VERSION)/lib/$(PERL_ARCH64)/auto/Sun/Solaris/Task/Task
$(ENABLE_PERL64)file \
    path=usr/perl5/$(PERL_VERSION)/lib/$(PERL_ARCH64)/auto/Sun/Solaris/Utils/Uti
file path=usr/perl5/$(PERL_VERSION)/lib/Sun/Solaris/BSM/_BSMparse.pm
file path=usr/perl5/$(PERL_VERSION)/lib/Sun/Solaris/Pg.pm
file path=usr/share/man/man3perl/Kstat.3perl
file path=usr/share/man/man3perl/Lgrp.3perl
file path=usr/share/man/man3perl/Project.3perl
file path=usr/share/man/man3perl/Task.3perl
license cr_Sun license=cr_Sun
```

```
 126 license usr/src/cmd/perl/THIRDPARTYLICENSE \
 127     license=usr/src/cmd/perl/THIRDPARTYLICENSE
```