```
**********************************************************
   34933 Mon Feb  9 11:03:02 2015
new/usr/src/Makefile.master
4863 illumos-gate can't be built with fresh perl versions
**********************************************************
```

```
   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) 1989, 2010, Oracle and/or its affiliates. All rights reserved.
  24 # Copyright (c) 2012 by Delphix. All rights reserved.
  25 # Copyright 2014 Garrett D'Amore <garrett@damore.org>
  26 # Copyright 2015, OmniTI Computer Consulting, Inc. All rights reserved.
  27 #

  29 #
  30 # Makefile.master, global definitions for system source
  31 #
  32 ROOT=           /proto

  34 #
  35 # Adjunct root, containing an additional proto area to be used for headers
  36 # and libraries.
  37 #
  38 ADJUNCT_PROTO=

  40 #
  41 # Adjunct for building things that run on the build machine.
  42 #
  43 NATIVE_ADJUNCT= /usr

  45 #
  46 # RELEASE_BUILD should be cleared for final release builds.
  47 # NOT_RELEASE_BUILD is exactly what the name implies.
  48 #
  49 # __GNUC toggles the building of ON components using gcc and related tools.
  50 # Normally set to '#', set it to '' to do gcc build.
  51 #
  52 # The declaration POUND_SIGN is always '#'. This is needed to get around the
  53 # make feature that '#' is always a comment delimiter, even when escaped or
  54 # quoted. We use this macro expansion method to get POUND_SIGN rather than
  55 # always breaking out a shell because the general case can cause a noticable
  56 # slowdown in build times when so many Makefiles include Makefile.master.
  57 #
  58 # While the majority of users are expected to override the setting below
  59 # with an env file (via nightly or bldenv), if you aren't building that way
  60 # (ie, you're using "ws" or some other bootstrapping method) then you need
  61 # this definition in order to avoid the subshell invocation mentioned above.
```



```
  62 #

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

  67 NOT_RELEASE_BUILD=
  68 RELEASE_BUILD=                          $(POUND_SIGN)
  69 $(RELEASE_BUILD)NOT_RELEASE_BUILD=      $(POUND_SIGN)
  70 PATCH_BUILD=                            $(POUND_SIGN)

  72 # SPARC_BLD is '#' for an Intel build.
  73 # INTEL_BLD is '#' for a Sparc build.
  74 SPARC_BLD_1=    $(MACH:i386=$(POUND_SIGN))
  75 SPARC_BLD=      $(SPARC_BLD_1:sparc=)
  76 INTEL_BLD_1=    $(MACH:sparc=$(POUND_SIGN))
  77 INTEL_BLD=      $(INTEL_BLD_1:i386=)

  79 # The variables below control the compilers used during the build.
  80 # There are a number of permutations.
  81 #
  82 # __GNUC and __SUNC control (and indicate) the primary compiler.  Whichever
  83 # one is not POUND_SIGN is the primary, with the other as the shadow.  They
  84 # may also be used to control entirely compiler-specific Makefile assignments.
  85 # __GNUC and GCC are the default.
  86 #
  87 # __GNUC64 indicates that the 64bit build should use the GNU C compiler.
  88 # There is no Sun C analogue.
  89 #
  90 # The following version-specific options are operative regardless of which
  91 # compiler is primary, and control the versions of the given compilers to be
  92 # used.  They also allow compiler-version specific Makefile fragments.
  93 #

  95 __SUNC=                 $(POUND_SIGN)
  96 $(__SUNC)__GNUC=        $(POUND_SIGN)
  97 __GNUC64=               $(__GNUC)

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

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

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

 109 JAVA_ROOT=      /usr/java

 111 SFW_ROOT=       /usr/sfw
 112 SFWINCDIR=      $(SFW_ROOT)/include
 113 SFWLIBDIR=      $(SFW_ROOT)/lib
 114 SFWLIBDIR64=    $(SFW_ROOT)/lib/$(MACH64)

 116 GCC_ROOT=       /opt/gcc/4.4.4
 117 GCCLIBDIR=      $(GCC_ROOT)/lib
 118 GCCLIBDIR64=    $(GCC_ROOT)/lib/$(MACH64)

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

 122 RPCGEN=         /usr/bin/rpcgen
 123 STABS=          $(ONBLD_TOOLS)/bin/$(MACH)/stabs
 124 ELFEXTRACT=     $(ONBLD_TOOLS)/bin/$(MACH)/elfextract
 125 MBH_PATCH=      $(ONBLD_TOOLS)/bin/$(MACH)/mbh_patch
 126 ECHO=           echo
 127 INS=            install
```

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

187 FILEMODE=       644
188 DIRMODE=        755

190 #
191 # The version of the patch makeup table optimized for build-time use.  Used
192 # during patch builds only.
193 $(PATCH_BUILD)PMTMO_FILE=$(SRC)/patch_makeup_table.mo
```



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

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

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

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

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

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

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

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

239 #
240 # Python bakes the mtime of the .py file into the compiled .pyc and
241 # rebuilds if the baked-in mtime != the mtime of the source file
242 # (rather than only if it's less than), thus when installing python
243 # files we must make certain to not adjust the mtime of the source
244 # (.py) file.
245 #
246 INS.pyfile=     $(INS.file); $(TOUCH) -r $< $@

248 # MACH must be set in the shell environment per uname -p on the build host
249 # More specific architecture variables should be set in lower makefiles.
250 #
251 # MACH64 is derived from MACH, and BUILD64 is set to '#' for
252 # architectures on which we do not build 64-bit versions.
253 # (There are no such architectures at the moment.)
254 #
255 # Set BUILD64=# in the environment to disable 64-bit amd64
256 # builds on i386 machines.

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

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

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

268 #
269 # C compiler mode. Future compilers may change the default on us,
270 # so force extended ANSI mode globally. Lower level makefiles can
271 # override this by setting CCMODE.
272 #
273 CCMODE=                 -Xa
274 CCMODE64=               -Xa

276 #
277 # C compiler verbose mode. This is so we can enable it globally,
278 # but turn it off in the lower level makefiles of things we cannot
279 # (or aren't going to) fix.
280 #
281 CCVERBOSE=              -v

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

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

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

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

318 # One optimization the compiler might perform is to turn this:
319 #       #pragma weak foo
320 #       extern int foo;
321 #       if (&foo)
322 #               foo = 5;
323 # into
324 #       foo = 5;
325 # Since we do some of this (foo might be referenced in common kernel code
```



```
326 # but provided only for some cpu modules or platforms), we disable this
327 # optimization.
328 #
329 sparc_CCUNBOUND = -Wd,-xsafe=unboundsym
330 i386_CCUNBOUND  =
331 CCUNBOUND       = $($(MACH)_CCUNBOUND)

333 #
334 # compiler '-xarch' flag. This is here to centralize it and make it
335 # overridable for testing.
336 sparc_XARCH=    -m32
337 sparcv9_XARCH=  -m64
338 i386_XARCH=
339 amd64_XARCH=    -m64 -Ui386 -U__i386

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

347 #
348 # These flags define what we need to be 'standalone' i.e. -not- part
349 # of the rather more cosy userland environment.  This basically means
350 # the kernel.
351 #
352 # XX64  future versions of gcc will make -mcmodel=kernel imply -mno-red-zone
353 #
354 sparc_STAND_FLAGS=      -_gcc=-ffreestanding
355 sparcv9_STAND_FLAGS=    -_gcc=-ffreestanding
356 # Disabling MMX also disables 3DNow, disabling SSE also disables all later
357 # additions to SSE (SSE2, AVX ,etc.)
358 NO_SIMD=                -_gcc=-mno-mmx -_gcc=-mno-sse
359 i386_STAND_FLAGS=       -_gcc=-ffreestanding $(NO_SIMD)
360 amd64_STAND_FLAGS=      -xmodel=kernel $(NO_SIMD)

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

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

368 #
369 # disable the incremental linker
370 ILDOFF=                 -xildoff
371 #
372 XDEPEND=                -xdepend
373 XFFLAG=                 -xF=%all
374 XESS=                   -xs
375 XSTRCONST=              -xstrconst

377 #
378 # turn warnings into errors (C)
379 CERRWARN = -errtags=yes -errwarn=%all
380 CERRWARN += -erroff=E_EMPTY_TRANSLATION_UNIT
381 CERRWARN += -erroff=E_STATEMENT_NOT_REACHED

383 CERRWARN += -_gcc=-Wno-missing-braces
384 CERRWARN += -_gcc=-Wno-sign-compare
385 CERRWARN += -_gcc=-Wno-unknown-pragmas
386 CERRWARN += -_gcc=-Wno-unused-parameter
387 CERRWARN += -_gcc=-Wno-missing-field-initializers

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

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

397 #
398 # turn warnings into errors (C++)
399 CCERRWARN=              -xwe

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

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

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

420 #
421 sparc_COPTFLAG=         -xO3
422 sparcv9_COPTFLAG=       -xO3
423 i386_COPTFLAG=          -O
424 amd64_COPTFLAG=         -xO3

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

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

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

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

445 #
446 # Default debug format for Sun Studio 11 is dwarf, so force it to
447 # generate stabs.
448 #
449 DEBUGFORMAT=    -xdebugformat=stabs

451 #
452 # Flags used to build in debug mode for ctf generation.  Bugs in the Devpro
453 # compilers currently prevent us from building with cc-emitted DWARF.
454 #
455 CTF_FLAGS_sparc = -g -Wc,-Qiselect-T1 $(C99MODE) $(CNOGLOBAL) $(CDWARFSTR)
456 CTF_FLAGS_i386  = -g $(C99MODE) $(CNOGLOBAL) $(CDWARFSTR)
```



```
458 CTF_FLAGS_sparcv9       = $(CTF_FLAGS_sparc)
459 CTF_FLAGS_amd64         = $(CTF_FLAGS_i386)

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

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

468 #
469 # Flags used with genoffsets
470 #
471 GOFLAGS = -_noecho \
472         $(CALLSYMS) \
473         $(CDWARFSTR)

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

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

481 #
482 # tradeoff time for space (smaller is better)
483 #
484 sparc_SPACEFLAG         = -xspace -W0,-Lt
485 sparcv9_SPACEFLAG       = -xspace -W0,-Lt
486 i386_SPACEFLAG          = -xspace
487 amd64_SPACEFLAG         =

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

492 #
493 # The Sun Studio 11 compiler has changed the behaviour of integer
494 # wrap arounds and so a flag is needed to use the legacy behaviour
495 # (without this flag panics/hangs could be exposed within the source).
496 #
497 sparc_IROPTFLAG         = -W2,-xwrap_int
498 sparcv9_IROPTFLAG       = -W2,-xwrap_int
499 i386_IROPTFLAG          =
500 amd64_IROPTFLAG         =

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

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

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

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

518 #
519 # These variables are intended ONLY for use by developers to safely pass extra
520 # flags to the compilers without unintentionally overriding Makefile-set
521 # flags.  They should NEVER be set to any value in a Makefile.
522 #
523 # They come last in the associated FLAGS variable such that they can
```

```
524 # explicitly override things if necessary, there are gaps in this, but it's
525 # the best we can manage.
526 #
527 CUSERFLAGS              =
528 CUSERFLAGS64            = $(CUSERFLAGS)
529 CCUSERFLAGS             =
530 CCUSERFLAGS64           = $(CCUSERFLAGS)

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

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

554 DTEXTDOM=-DTEXT_DOMAIN=\"$(TEXT_DOMAIN)\"       # For messaging.
555 DTS_ERRNO=-D_TS_ERRNO
556 CPPFLAGS.master=$(DTEXTDOM) $(DTS_ERRNO) \
557         $(ENVCPPFLAGS1) $(ENVCPPFLAGS2) $(ENVCPPFLAGS3) $(ENVCPPFLAGS4) \
558         $(ADJUNCT_PROTO:%=-I%/usr/include)
559 CPPFLAGS.native=$(ENVCPPFLAGS1) $(ENVCPPFLAGS2) $(ENVCPPFLAGS3) \
560                 $(ENVCPPFLAGS4) -I$(NATIVE_ADJUNCT)/include
561 CPPFLAGS=       $(CPPFLAGS.master)
562 AS_CPPFLAGS=    $(CPPFLAGS.master)
563 JAVAFLAGS=      -deprecation

565 #
566 # For source message catalogue
567 #
568 .SUFFIXES: $(SUFFIXES) .i .po
569 MSGROOT= $(ROOT)/catalog
570 MSGDOMAIN= $(MSGROOT)/$(TEXT_DOMAIN)
571 MSGDOMAINPOFILE = $(MSGDOMAIN)/$(POFILE)
572 DCMSGDOMAIN= $(MSGROOT)/LC_TIME/$(TEXT_DOMAIN)
573 DCMSGDOMAINPOFILE = $(DCMSGDOMAIN)/$(DCFILE:.dc=.po)

575 CLOBBERFILES += $(POFILE) $(POFILES)
576 COMPILE.cpp= $(CC) -E -C $(CFLAGS) $(CPPFLAGS)
577 XGETTEXT= /usr/bin/xgettext
578 XGETFLAGS= -c TRANSLATION_NOTE
579 GNUXGETTEXT= /usr/gnu/bin/xgettext
580 GNUXGETFLAGS= --add-comments=TRANSLATION_NOTE --keyword=_ \
581         --strict --no-location --omit-header
582 BUILD.po= $(XGETTEXT) $(XGETFLAGS) -d $(<F) $<.i ;\
583         $(RM)   $@ ;\
584         $(SED) "/^domain/d" < $(<F).po > $@ ;\
585         $(RM) $(<F).po $<.i

587 #
588 # This is overwritten by local Makefile when PROG is a list.
589 #
```



```
590 POFILE= $(PROG).po

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

609 sparc_CCOPTFLAG=        -O
610 sparcv9_CCOPTFLAG=      -O
611 i386_CCOPTFLAG=         -O
612 amd64_CCOPTFLAG=        -O

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

621 #
622 #
623 #
624 ELFWRAP_FLAGS   =
625 ELFWRAP_FLAGS64 =       -64

627 #
628 # Various mapfiles that are used throughout the build, and delivered to
629 # /usr/lib/ld.
630 #
631 MAPFILE.NED_i386 =      $(SRC)/common/mapfiles/common/map.noexdata
632 MAPFILE.NED_sparc =
633 MAPFILE.NED =           $(MAPFILE.NED_$(MACH))
634 MAPFILE.PGA =           $(SRC)/common/mapfiles/common/map.pagealign
635 MAPFILE.NES =           $(SRC)/common/mapfiles/common/map.noexstk
636 MAPFILE.FLT =           $(SRC)/common/mapfiles/common/map.filter
637 MAPFILE.LEX =           $(SRC)/common/mapfiles/common/map.lex.yy

639 #
640 # Generated mapfiles that are compiler specific, and used throughout the
641 # build.  These mapfiles are not delivered in /usr/lib/ld.
642 #
643 MAPFILE.NGB_sparc=      $(SRC)/common/mapfiles/gen/sparc_cc_map.noexeglobs
644 $(__GNUC64)MAPFILE.NGB_sparc= \
645                         $(SRC)/common/mapfiles/gen/sparc_gcc_map.noexeglobs
646 MAPFILE.NGB_sparcv9=    $(SRC)/common/mapfiles/gen/sparcv9_cc_map.noexeglobs
647 $(__GNUC64)MAPFILE.NGB_sparcv9= \
648                         $(SRC)/common/mapfiles/gen/sparcv9_gcc_map.noexeglobs
649 MAPFILE.NGB_i386=       $(SRC)/common/mapfiles/gen/i386_cc_map.noexeglobs
650 $(__GNUC64)MAPFILE.NGB_i386= \
651                         $(SRC)/common/mapfiles/gen/i386_gcc_map.noexeglobs
652 MAPFILE.NGB_amd64=      $(SRC)/common/mapfiles/gen/amd64_cc_map.noexeglobs
653 $(__GNUC64)MAPFILE.NGB_amd64= \
654                         $(SRC)/common/mapfiles/gen/amd64_gcc_map.noexeglobs
655 MAPFILE.NGB =           $(MAPFILE.NGB_$(MACH))
```

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

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

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

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

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

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

LINK.c=         $(CC) $(CFLAGS) $(CPPFLAGS) $(LDFLAGS)
LINK64.c=       $(CC) $(CFLAGS64) $(CPPFLAGS) $(LDFLAGS)
NORUNPATH=      -norunpath -nolib
LINK.cc=        $(CCC) $(CCFLAGS) $(CPPFLAGS) $(NORUNPATH) \
                $(LDFLAGS) $(CCNEEDED)
LINK64.cc=      $(CCC) $(CCFLAGS64) $(CPPFLAGS) $(NORUNPATH) \
                $(LDFLAGS) $(CCNEEDED)

#
# lint macros
#
# Note that the undefine of __PRAGMA_REDEFINE_EXTNAME can be removed once
# ON is built with a version of lint that has the fix for 4484186.
#
ALWAYS_LINT_DEFS =      -errtags=yes -s
```



```
ALWAYS_LINT_DEFS +=     -erroff=E_PTRDIFF_OVERFLOW
ALWAYS_LINT_DEFS +=     -erroff=E_ASSIGN_NARROW_CONV
ALWAYS_LINT_DEFS +=     -U__PRAGMA_REDEFINE_EXTNAME
ALWAYS_LINT_DEFS +=     $(C99LMODE)
ALWAYS_LINT_DEFS +=     -errsecurity=$(SECLEVEL)
ALWAYS_LINT_DEFS +=     -erroff=E_SEC_CREAT_WITHOUT_EXCL
ALWAYS_LINT_DEFS +=     -erroff=E_SEC_FORBIDDEN_WARN_CREAT
# XX64 -- really only needed for amd64 lint
ALWAYS_LINT_DEFS +=     -erroff=E_ASSIGN_INT_TO_SMALL_INT
ALWAYS_LINT_DEFS +=     -erroff=E_CAST_INT_CONST_TO_SMALL_INT
ALWAYS_LINT_DEFS +=     -erroff=E_CAST_INT_TO_SMALL_INT
ALWAYS_LINT_DEFS +=     -erroff=E_CAST_TO_PTR_FROM_INT
ALWAYS_LINT_DEFS +=     -erroff=E_COMP_INT_WITH_LARGE_INT
ALWAYS_LINT_DEFS +=     -erroff=E_INTEGRAL_CONST_EXP_EXPECTED
ALWAYS_LINT_DEFS +=     -erroff=E_PASS_INT_TO_SMALL_INT
ALWAYS_LINT_DEFS +=     -erroff=E_PTR_CONV_LOSES_BITS

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

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

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

# Define native compilation macros
#

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

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

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

sparcv9_CC=             $(ONBLD_TOOLS)/bin/$(MACH)/cw -_cc
$(__GNUC64)sparcv9_CC=  $(ONBLD_TOOLS)/bin/$(MACH)/cw -_gcc
```

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

795 i386_CC=               $(ONBLD_TOOLS)/bin/$(MACH)/cw -_cc
796 $(__GNUC)i386_CC=      $(ONBLD_TOOLS)/bin/$(MACH)/cw -_gcc
797 i386_CCC=              $(ONBLD_TOOLS)/bin/$(MACH)/cw -_CC
798 $(__GNUC)i386_CCC=     $(ONBLD_TOOLS)/bin/$(MACH)/cw -_g++
799 i386_CPP=              /usr/ccs/lib/cpp
800 i386_AS=               /usr/ccs/bin/as
801 $(__GNUC)i386_AS=      $(ONBLD_TOOLS)/bin/$(MACH)/aw
802 i386_LD=               /usr/ccs/bin/ld
803 i386_LINT=             $(SPRO_ROOT)/sunstudio12.1/bin/lint

805 amd64_CC=              $(ONBLD_TOOLS)/bin/$(MACH)/cw -_cc
806 $(__GNUC64)amd64_CC=   $(ONBLD_TOOLS)/bin/$(MACH)/cw -_gcc
807 amd64_CCC=             $(ONBLD_TOOLS)/bin/$(MACH)/cw -_CC
808 $(__GNUC64)amd64_CCC=  $(ONBLD_TOOLS)/bin/$(MACH)/cw -_g++
809 amd64_CPP=             /usr/ccs/lib/cpp
810 amd64_AS=              $(ONBLD_TOOLS)/bin/$(MACH)/aw
811 amd64_LD=              /usr/ccs/bin/ld
812 amd64_LINT=            $(SPRO_ROOT)/sunstudio12.1/bin/lint

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

821 #
822 # Makefile.master.64 overrides these settings
823 #
824 CC=                   $(NATIVECC)
825 CCC=                  $(NATIVECCC)
826 CPP=                  $(NATIVECPP)
827 AS=                   $(NATIVEAS)
828 LD=                   $(NATIVELD)
829 LINT=                 $(NATIVELINT)

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

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

840 BDIRECT=        -Bdirect
841 BDYNAMIC=       -Bdynamic
842 BLOCAL=         -Blocal
843 BNODIRECT=      -Bnodirect
844 BREDUCE=        -Breduce
845 BSTATIC=        -Bstatic

847 ZDEFS=          -zdefs
848 ZDIRECT=        -zdirect
849 ZIGNORE=        -zignore
850 ZINITFIRST=     -zinitfirst
851 ZINTERPOSE=     -zinterpose
852 ZLAZYLOAD=      -zlazyload
853 ZLOADFLTR=      -zloadfltr
```



```
854 ZMULDEFS=       -zmuldefs
855 ZNODEFAULTLIB=  -znodefaultlib
856 ZNODEFS=        -znodefs
857 ZNODELETE=      -znodelete
858 ZNODLOPEN=      -znodlopen
859 ZNODUMP=        -znodump
860 ZNOLAZYLOAD=    -znolazyload
861 ZNOLDYNSYM=     -znoldynsym
862 ZNORELOC=       -znoreloc
863 ZNOVERSION=     -znoversion
864 ZRECORD=        -zrecord
865 ZREDLOCSYM=     -zredlocsym
866 ZTEXT=          -ztext
867 ZVERBOSE=       -zverbose

869 GSHARED=        -G
870 CCMT=           -mt

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

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

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

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

897 AS_PICFLAGS=            $(C_PICFLAGS)
898 AS_BIGPICFLAGS=         $(C_BIGPICFLAGS)

900 #
901 # Default label for CTF sections
902 #
903 CTFCVTFLAGS=            -i -L VERSION
904 $(SRCDBGBLD)CTFCVTFLAGS         += -g

906 #
907 # Override to pass module-specific flags to ctfmerge.  Currently used only by
908 # krtld to turn on fuzzy matching, and source-level debugging to inhibit
909 # stripping.
910 #
911 CTFMRGFLAGS=
912 $(SRCDBGBLD)CTFMRGFLAGS         += -g


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

917 ELFSIGN_O=      $(TRUE)
918 ELFSIGN_CRYPTO= $(ELFSIGN_O)
919 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=                        :
$(RELEASE_BUILD)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)
        $(RM) y.tab.c
.y.o:
        $(YACC.y) $<
        $(COMPILE.c) -o $@ y.tab.c $(CTFCONVERT_HOOK)
        $(POST_PROCESS_O)
```

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

1066 .bin.o:
1067         $(COMPILE.b) -o $@ $<
1068         $(POST_PROCESS_O)

1070 .java.class:
1071         $(COMPILE.java) $<

1073 # Bourne and Korn shell script message catalog build rules.
1074 # We extract all gettext strings with sed(1) (being careful to permit
1075 # multiple gettext strings on the same line), weed out the dups, and
1076 # build the catalogue with awk(1).

1078 .sh.po .ksh.po:
1079         $(SED) -n -e ":a"                                   \
1080                   -e "h"                                        \
1081                   -e "s/.*gettext *\(\"[^\"]*\"\).*/\1/p"       \
1082                   -e "x"                                        \
1083                   -e "s/\(.*\)gettext *\"[^\"]*\"\(.*\)/\1\2/"  \
1084                   -e "t a"                                      \
1085                 $< | sort -u | awk '{ print "msgid\t" $$0 "\nmsgstr" }' > $@

1087 #
1088 # Python and Perl executable and message catalog build rules.
1089 #
1090 .SUFFIXES: .pl .pm .py .pyc

1092 .pl:
1093         $(RM) $@;
1094         $(SED) -e "s@TEXT_DOMAIN@\"$(TEXT_DOMAIN)\"@" $< > $@;
1095         $(CHMOD) +x $@

1097 .py:
1098         $(RM) $@; $(CAT) $< > $@; $(CHMOD) +x $@

1100 .py.pyc:
1101         $(RM) $@
1102         $(PYTHON) -mpy_compile $<
1103         @[ $(<)c = $@ ] || $(MV) $(<)c $@

1105 .py.po:
1106         $(GNUXGETTEXT) $(GNUXGETFLAGS) -d $(<F:%.py=%) $< ;

1108 .pl.po .pm.po:
1109         $(XGETTEXT) $(XGETFLAGS) -d $(<F) $< ;
1110         $(RM)   $@ ;
1111         $(SED) "/^domain/d" < $(<F).po > $@ ;
1112         $(RM) $(<F).po

1114 #
1115 # When using xgettext, we want messages to go to the default domain,
1116 # rather than the specified one.  This special version of the
1117 # COMPILE.cpp macro effectively prevents expansion of TEXT_DOMAIN,
```



```
1118 # causing xgettext to put all messages into the default domain.
1119 #
1120 CPPFORPO=$(COMPILE.cpp:\"$(TEXT_DOMAIN)\"=TEXT_DOMAIN)

1122 .c.i:
1123         $(CPPFORPO) $< > $@

1125 .h.i:
1126         $(CPPFORPO) $< > $@

1128 .y.i:
1129         $(YACC) -d $<
1130         $(CPPFORPO) y.tab.c  > $@
1131         $(RM) y.tab.c

1133 .l.i:
1134         $(LEX) $<
1135         $(CPPFORPO) lex.yy.c  > $@
1136         $(RM) lex.yy.c

1138 .c.po:
1139         $(CPPFORPO) $< > $<.i
1140         $(BUILD.po)

1142 .y.po:
1143         $(YACC) -d $<
1144         $(CPPFORPO) y.tab.c  > $<.i
1145         $(BUILD.po)
1146         $(RM) y.tab.c

1148 .l.po:
1149         $(LEX) $<
1150         $(CPPFORPO) lex.yy.c  > $<.i
1151         $(BUILD.po)
1152         $(RM) lex.yy.c

1154 #
1155 # Rules to perform stylistic checks
1156 #
1157 .SUFFIXES: .x .xml .check .xmlchk

1159 .h.check:
1160         $(DOT_H_CHECK)

1162 .x.check:
1163         $(DOT_X_CHECK)

1165 .xml.xmlchk:
1166         $(MANIFEST_CHECK)

1168 #
1169 # Include rules to render automated sccs get rules "safe".
1170 #
1171 include $(SRC)/Makefile.noget
```

```
**********************************************************
    1215 Mon Feb  9 11:03:02 2015
new/usr/src/cmd/perl/Makefile.perl
4863 illumos-gate can't be built with fresh perl versions
**********************************************************
```

```
#
# 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,
# PERL_VERSION used to be set here,
# but as it is also needed in usr/src/pkg/Makefile,
# the definition was moved to usr/src/Makefile.master

PERL_ARCH = i86pc-solaris-64int
$(SPARC_BLD)PERL_ARCH = sun4-solaris-64int

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

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

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

ROOTPERLMOD = $(ROOTPERLMODDIR)/$(MODULE).pm
ROOTPERLEXT = $(ROOTPERLEXTDIR)/$(MODULE).so

C99MODE = $(C99_ENABLE)
```

****
1421 Mon Feb 9 11:03:03 2015
new/usr/src/cmd/perl/Makefile.targ
4863 illumos-gate can't be built with fresh perl versions
****

```
   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 2015, OmniTI Computer Consulting, Inc. All rights reserved.
  13 # Copyright 2014, OmniTI Computer Consulting, Inc. All rights reserved.
  14 #

  16 # Link against libc as perl solaris specs
  17 $(PERLEXT):= LDLIBS += -lc

  19 # Allow for undefined symbols satisfied by perl
  20 $(PERLEXT):= ZDEFS =

  22 $(ROOTPERLEXT) := FILEMODE = 0555
  23 $(ROOTPERLMOD) := FILEMODE = 0444

  25 # CFLAGS for perl, specifically.
  26 PCFLAGS= -DPERL_EUPXS_ALWAYS_EXPORT -D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64 \
  27         -DPERL_USE_SAFE_PUTENV -D_TS_ERRNO
  26 PCFLAGS= -D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64 -DPERL_USE_SAFE_PUTENV \
  27         -D_TS_ERRNO

  29 $(MACH):
  30         $(INS.dir)

  32 $(PERLEXT): $(MACH)/$(MODULE).o
  33         $(BUILD.SO) $(MACH)/$(MODULE).o

  35 $(MACH)/$(MODULE).o: $(MACH)/$(MODULE).c
  36         $(COMPILE.c) $(PCFLAGS) $(C_PICFLAGS) -I$(PERLINCDIR) $< -o $@

  38 $(MACH)/$(MODULE).c: $(MACH) $(MODULE).xs
  39         $(PERLDIR)/bin/xsubpp $(XSUBPPFLAGS) $(MODULE).xs >$@

  41 $(ROOTPERLMODDIR):
  42         $(INS.dir)

  44 $(ROOTPERLMOD): $(ROOTPERLMODDIR) $(MODULE).pm
  45         $(RM) $@; $(INS) -s -m $(FILEMODE) -f $^

  47 $(ROOTPERLEXTDIR):
  48         $(INS.dir)

  50 $(ROOTPERLEXT): $(ROOTPERLEXTDIR) $(MACH)/$(MODULE).so
  51         $(RM) $@; $(INS) -s -m $(FILEMODE) -f $^
```

```
**********************************************************
   24415 Mon Feb  9 11:03:03 2015
new/usr/src/pkg/Makefile
4863 illumos-gate can't be built with fresh perl versions
**********************************************************
```

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

#
# Copyright (c) 2010, Oracle and/or its affiliates. All rights reserved.
# Copyright 2015, OmniTI Computer Consulting, Inc. All rights reserved.
#

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

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

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

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

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



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

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

EXCEPTIONS= packaging

PKGMOGRIFY= pkgmogrify

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

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

#
# The tools proto must be specified for dependency generation.
# Publication from the tools proto area is managed in the
```

```
128 # publication rule.
129 #
130 $(PDIR)/developer-build-onbld.dep:= PKGROOT= $(TOOLSROOT)

132 PKGPUBLISHER= $(PKGPUBLISHER_REDIST)

134 #
135 # To get these defaults, manifests should simply refer to $(PKGVERS).
136 #
137 PKGVERS_COMPONENT= 0.$(RELEASE)
138 PKGVERS_BUILTON= $(RELEASE)
139 PKGVERS_BRANCH= 0.$(ONNV_BUILDNUM)
140 PKGVERS= $(PKGVERS_COMPONENT),$(PKGVERS_BUILTON)-$(PKGVERS_BRANCH)

142 #
143 # The ARCH32 and ARCH64 macros are used in the manifests to express
144 # architecture-specific subdirectories in the installation paths
145 # for isaexec'd commands.
146 #
147 # We can't simply use $(MACH32) and $(MACH64) here, because they're
148 # only defined for the build architecture.  To do cross-platform
149 # packaging, we need both values.
150 #
151 i386_ARCH32= i86
152 sparc_ARCH32= sparcv7
153 i386_ARCH64= amd64
154 sparc_ARCH64= sparcv9

156 #
157 # macros and transforms needed by pkgmogrify
158 #
159 # If you append to this list using target-specific assignments (:=),
160 # be very careful that the targets are of the form $(PDIR)/pkgname.  If
161 # you use a higher level target, or a package list, you'll trigger a
162 # complete reprocessing of all manifests because they'll fail command
163 # dependency checking.
164 #
165 PM_TRANSFORMS= common_actions publish restart_fmri facets defaults \
166         extract_metadata
167 PM_INC= transforms manifests

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

184 PKGDEP_TOKENS_i386= \
185         'PLATFORM=i86hvm' \
186         'PLATFORM=i86pc' \
187         'PLATFORM=i86xpv' \
188         'ISALIST=amd64' \
189         'ISALIST=i386'
190 PKGDEP_TOKENS_sparc= \
191         'PLATFORM=sun4u' \
192         'PLATFORM=sun4v' \
193         'ISALIST=sparcv9' \
```



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

197 #
198 # The package lists are generated with $(PKGDEP_TYPE) as their
199 # dependency types, so that they can be included by either an
200 # incorporation or a group package.
201 #
202 $(PDIR)/osnet-redist.mog := PKGDEP_TYPE= require
203 $(PDIR)/osnet-incorporation.mog:= PKGDEP_TYPE= incorporate

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

208 #
209 # All packaging build products should go into $(PDIR), so they don't
210 # need to be included separately in CLOBBERFILES.
211 #
212 CLOBBERFILES= $(PDIR) proto_list_$(PKGMACH) install-$(PKGMACH).out \
213         license-list

215 #
216 # By default, PKGS will list all manifests.  To build and/or publish a
217 # subset of packages, override this on the command line or in the
218 # build environment and then reference (implicitly or explicitly) the all
219 # or install targets.
220 #
221 MANIFESTS :sh= (cd manifests; print *.mf)
222 PKGS= $(MANIFESTS:%.mf=%)
223 DEP_PKGS= $(PKGS:%=$(PDIR)/%.dep)
224 PROC_PKGS= $(PKGS:%=$(PDIR)/%.mog)

226 #
227 # Track the synthetic manifests separately so we can properly express
228 # build rules and dependencies.  The synthetic and real packages use
229 # different sets of transforms and macros for pkgmogrify.
230 #
231 SYNTH_PKGS= osnet-incorporation osnet-redist
232 DEP_SYNTH_PKGS= $(SYNTH_PKGS:%=$(PDIR)/%.dep)
233 PROC_SYNTH_PKGS= $(SYNTH_PKGS:%=$(PDIR)/%.mog)

235 #
236 # Root of pkg image to use for dependency resolution
237 # Normally / on the machine used to build the binaries
238 #
239 PKGDEP_RESOLVE_IMAGE = /

241 #
242 # For each package, we determine the target repository based on
243 # manifest-embedded metadata.  Because we make that determination on
244 # the fly, the publication target cannot be expressed as a
245 # subdirectory inside the unknown-by-the-makefile target repository.
246 #
247 # In order to limit the target set to real files in known locations,
248 # we use a ".pub" file in $(PDIR) for each processed manifest, regardless
249 # of content or target repository.
250 #
251 PUB_PKGS= $(SYNTH_PKGS:%=$(PDIR)/%.pub) $(PKGS:%=$(PDIR)/%.pub)

253 #
254 # Any given repository- and status-specific package list may be empty,
255 # but we can only determine that dynamically, so we always generate all
256 # lists for each repository we're building.
257 #
258 # The meanings of each package status are as follows:
259 #
```

```
260 #       PKGSTAT         meaning
261 #       ----------      ------------------------------------------------------
262 #       noincorp        Do not include in incorporation or group package
263 #       obsolete        Include in incorporation, but not group package
264 #       renamed         Include in incorporation, but not group package
265 #       current         Include in incorporation and group package
266 #
267 # Since the semantics of the "noincorp" package status dictate that
268 # such packages are not included in the incorporation or group packages,
269 # there is no need to build noincorp package lists.
270 #
271 PKGLISTS= \
272         $(REPOS:%=$(PDIR)/packages.%.current) \
273         $(REPOS:%=$(PDIR)/packages.%.renamed) \
274         $(REPOS:%=$(PDIR)/packages.%.obsolete)

276 .KEEP_STATE:

278 .PARALLEL: $(PKGS) $(PROC_PKGS) $(DEP_PKGS) \
279         $(PROC_SYNTH_PKGS) $(DEP_SYNTH_PKGS) $(PUB_PKGS)

281 #
282 # For a single manifest, the dependency chain looks like this:
283 #
284 #       raw manifest (mypkg.mf)
285 #               |
286 #               | use pkgmogrify to process raw manifest
287 #               |
288 #       processed manifest (mypkg.mog)
289 #               |
290 #          *    | use pkgdepend generate to generate dependencies
291 #               |
292 #       manifest with TBD dependencies (mypkg.dep)
293 #               |
294 #          %    | use pkgdepend resolve to resolve dependencies
295 #               |
296 #       manifest with dependencies resolved (mypkg.res)
297 #               |
298 #               | use pkgsend to publish the package
299 #               |
300 #       placeholder to indicate successful publication (mypkg.pub)
301 #
302 # * This may be suppressed via SUPPRESSPKGDEP.  The resulting
303 #   packages will install correctly, but care must be taken to
304 #   install all dependencies, because pkg will not have the input
305 #   it needs to determine this automatically.
306 #
307 # % This is included in this diagram to make the picture complete, but
308 #   this is a point of synchronization in the build process.
309 #   Dependency resolution is actually done once on the entire set of
310 #   manifests, not on a per-package basis.
311 #
312 # The full dependency chain for generating everything that needs to be
313 # published, without actually publishing it, looks like this:
314 #
315 #       processed synthetic packages
316 #               |               |
317 #       package lists       synthetic package manifests
318 #               |
319 #       processed real packages
320 #           |           |
321 #       package dir     real package manifests
322 #
323 # Here, each item is a set of real or synthetic packages.  For this
324 # portion of the build, no reference is made to the proto area.  It is
325 # therefore suitable for the "all" target, as opposed to "install."
```



```
326 #
327 # Since each of these steps is expressed explicitly, "all" need only
328 # depend on the head of the chain.
329 #
330 # From the end of manifest processing, the publication dependency
331 # chain looks like this:
332 #
333 #               repository metadata (catalogs and search indices)
334 #                       |
335 #                       | pkgrepo refresh
336 #                       |
337 #               published packages
338 #                 |             |
339 #                 |             | pkgsend publish
340 #                 |             |
341 #         repositories      resolved dependencies
342 #                 |                     |
343 # pkgsend         |                     | pkgdepend resolve
344 # create-repository                     |
345 #                 |             generated dependencies
346 #        repo directories               |
347 #                                       | pkgdepend
348 #                                       |
349 #                                 processed manifests
350 #

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

354 all: $(ALL_TARGETS)

356 #
357 # This will build the directory to contain the processed manifests
358 # and the metadata symlinks.
359 #
360 $(PDIR):
361         @print "Creating $(@)"
362         $(PKGDEBUG)$(INS.dir)

364 #
365 # This rule resolves dependencies across all published manifests.
366 #
367 # We shouldn't have to ignore the error from pkgdepend, but until
368 # 16012 and its dependencies are resolved, pkgdepend will always exit
369 # with an error.
370 #
371 $(PDIR)/gendeps: $(DEP_SYNTH_PKGS) $(DEP_PKGS)
372         -$(PKGDEBUG)if [ "$(SUPPRESSPKGDEP)" = "true" ]; then \
373                 print "Suppressing dependency resolution"; \
374                 for p in $(DEP_PKGS:%.dep=%); do \
375                         $(CP) $$p.dep $$p.res; \
376                 done; \
377         else \
378                 print "Resolving dependencies"; \
379                 pkgdepend -R $(PKGDEP_RESOLVE_IMAGE) resolve \
380                     -m $(DEP_SYNTH_PKGS) $(DEP_PKGS); \
381                 for p in $(DEP_SYNTH_PKGS:%.dep=%) $(DEP_PKGS:%.dep=%); do \
382                         if [ "$$(print $$p.metadata.*)" = \
383                              "$$(print $$p.metadata.noincorp.*)" ]; \
384                         then \
385                                 print "Removing dependency versions from $$p"; \
386                                 $(PKGMOGRIFY) $(PKGMOG_VERBOSE) \
387                                     -O $$p.res -I transforms \
388                                     strip_versions $$p.dep.res; \
389                                 $(RM) $$p.dep.res; \
390                         else \
391                                 $(MV) $$p.dep.res $$p.res; \
```

```
392                         fi; \
393                 done; \
394         fi
395         $(PKGDEBUG)$(TOUCH) $(@)

397 install: $(ALL_TARGETS) repository-metadata

399 repository-metadata: publish_pkgs
400         $(PKGDEBUG)for r in $(REPOS); do \
401                 pkgrepo refresh -s $(PKGDEST)/repo.$$r; \
402         done

404 #
405 # Since we create zero-length processed manifests for a graceful abort
406 # from pkgmogrify, we need to detect that here and make no effort to
407 # publish the package.
408 #
409 # For all other packages, we publish them regardless of status.  We
410 # derive the target repository as a component of the metadata-derived
411 # symlink for each package.
412 #
413 publish_pkgs: $(REPOS:%=$(PKGDEST)/repo.%) $(PDIR)/gendeps .WAIT $(PUB_PKGS)

415 #
416 # Before publishing, we want to pull the license files from $CODEMGR_WS
417 # into the proto area.  This allows us to NOT pass $SRC (or
418 # $CODEMGR_WS) as a basedir for publication.
419 #
420 $(PUB_PKGS): stage-licenses

422 #
423 # Initialize the empty on-disk repositories
424 #
425 $(REPOS:%=$(PKGDEST)/repo.%):
426         @print "Initializing $(@F)"
427         $(PKGDEBUG)$(INS.dir)
428         $(PKGDEBUG)pkgsend -s file://$(@) create-repository \
429                 --set-property publisher.prefix=$(PKGPUBLISHER)

431 #
432 # rule to process real manifests
433 #
434 # To allow redistributability and package status to change, we must
435 # remove not only the actual build target (the processed manifest), but
436 # also the incidental ones (the metadata-derived symlinks).
437 #
438 # If pkgmogrify exits cleanly but fails to create the specified output
439 # file, it means that it encountered an abort directive.  That means
440 # that this package should not be published for this particular build
441 # environment.  Since we can't prune such packages from $(PKGS)
442 # retroactively, we need to create an empty target file to keep make
443 # from trying to rebuild it every time.  For these empty targets, we
444 # do not create metadata symlinks.
445 #
446 # Automatic dependency resolution to files is also done at this phase of
447 # processing.  The skipped packages are skipped due to existing bugs
448 # in pkgdepend.
449 #
450 # The incorporation dependency is tricky: it needs to go into all
451 # current and renamed manifests (ie all incorporated packages), but we
452 # don't know which those are until after we run pkgmogrify.  So
453 # instead of expressing it as a transform, we tack it on ex post facto.
454 #
455 # Implementation notes:
456 #
457 # - The first $(RM) must not match other manifests, or we'll run into
```



```
458 #   race conditions with parallel manifest processing.
459 #
460 # - The make macros [ie $(MACRO)] are evaluated when the makefile is
461 #   read in, and will result in a fixed, macro-expanded rule for each
462 #   target enumerated in $(PROC_PKGS).
463 #
464 # - The shell variables (ie $$VAR) are assigned on the fly, as the rule
465 #   is executed.  The results may only be referenced in the shell in
466 #   which they are assigned, so from the perspective of make, all code
467 #   that needs these variables needs to be part of the same line of
468 #   code.  Hence the use of command separators and line continuation
469 #   characters.
470 #
471 # - The extract_metadata transforms are designed to spit out shell
472 #   variable assignments to stdout.  Those are published to the
473 #   .vars temporary files, and then used as input to the eval
474 #   statement.  This is done in stages specifically so that pkgmogrify
475 #   can signal failure if the manifest has a syntactic or other error.
476 #   The eval statement should begin with the default values, and the
477 #   output from pkgmogrify (if any) should be in the form of a
478 #   variable assignment to override those defaults.
479 #
480 # - When this rule completes execution, it must leave an updated
481 #   target file ($@) in place, or make will reprocess the package
482 #   every time it encounters it as a dependency.  Hence the "touch"
483 #   statement to ensure that the target is created, even when
484 #   pkgmogrify encounters an abort in the publish transforms.
485 #

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

489 $(PDIR)/%.mog: manifests/%.mf
490         @print "Processing manifest $(<F)"
491         @env PKGFMT_OUTPUT=v1 pkgfmt -c $<
492         $(PKGDEBUG)$(RM) $(@) $(@:%.mog=%) $(@:%.mog=%.nodepend) \
493                 $(@:%.mog=%.lics) $(PDIR)/$(@F:%.mog=%).metadata.* $(@).vars
494         $(PKGDEBUG)$(PKGMOGRIFY) $(PKGMOG_VERBOSE) $(PM_INC:%= -I %) \
495                 $(PKGMOG_DEFINES:%=-D %) -P $(@).vars -O $(@) \
496                 $(<) $(PM_TRANSFORMS)
497         $(PKGDEBUG)eval REPO=redist PKGSTAT=current NODEPEND=$(SUPPRESSPKGDEP) \
498                 '$(CAT) -s $(@).vars'; \
499         if [ -f $(@) ]; then \
500                 if [ "$$NODEPEND" != "false" ]; then \
501                         $(TOUCH) $(@:%.mog=%.nodepend); \
502                 fi; \
503                 $(LN) -s $(@F) \
504                         $(PDIR)/$(@F:%.mog=%).metadata.$$PKGSTAT.$$REPO; \
505                 if [ \( "$$PKGSTAT" = "current" \) -o \
506                      \( "$$PKGSTAT" = "renamed" \) ]; \
507                         then print $(PKGDEP_INCORP) >> $(@); \
508                 fi; \
509                 print $$LICS > $(@:%.mog=%.lics); \
510         else \
511                 $(TOUCH) $(@) $(@:%.mog=%.lics); \
512         fi
513         $(PKGDEBUG)$(RM) $(@).vars

515 $(PDIR)/%.dep: $(PDIR)/%.mog
516         @print "Generating dependencies for $(<F)"
517         $(PKGDEBUG)$(RM) $(@)
518         $(PKGDEBUG)if [ ! -f $(@:%.dep=%.nodepend) ]; then \
519                 pkgdepend generate -m $(PKGDEP_TOKENS:%=-D %) $(<) \
520                         $(PKGROOT) > $(@); \
521         else \
522                 $(CP) $(<) $(@); \
523         fi
```

```
525 #
526 # The full chain implies that there should be a .dep.res suffix rule,
527 # but dependency generation is done on a set of manifests, rather than
528 # on a per-manifest basis.  Instead, see the gendeps rule above.
529 #

531 $(PDIR)/%.pub: $(PDIR)/%.res
532         $(PKGDEBUG)m=$$(basename $(@:%.pub=%).metadata.*); \
533         r=$${m#$(@F:%.pub=%.metadata.)+(?).}; \
534         if [ -s $(<) ]; then \
535                 print "Publishing $(@F:%.pub=%) to $$r repository"; \
536                 pkgsend -s file://$(PKGDEST)/repo.$$r publish \
537                     -d $(PKGROOT) -d $(TOOLSROOT) \
538                     -d license_files -d $(PKGROOT)/licenses \
539                     --fmri-in-manifest --no-index --no-catalog $(<) \
540                     > /dev/null; \
541         fi; \
542         $(TOUCH) $(@);

544 #
545 # rule to build the synthetic manifests
546 #
547 # This rule necessarily has PKGDEP_TYPE that changes according to
548 # the specific synthetic manifest.  Rather than escape command
549 # dependency checking for the real manifest processing, or failing to
550 # express the (indirect) dependency of synthetic manifests on real
551 # manifests, we simply split this rule out from the one above.
552 #
553 # The implementation notes from the previous rule are applicable
554 # here, too.
555 #
556 $(PROC_SYNTH_PKGS): $(PKGLISTS) $$(@F:%.mog=%.mf)
557         @print "Processing synthetic manifest $(@F:%.mog=%.mf)"
558         $(PKGDEBUG)$(RM) $(@) $(PDIR)/$(@F:%.mog=%).metadata.* $(@).vars
559         $(PKGDEBUG)$(PKGMOGRIFY) $(PKGMOG_VERBOSE) -I transforms -I $(PDIR) \
560                 $(PKGMOG_DEFINES:%=-D %) -D PKGDEP_TYPE=$(PKGDEP_TYPE) \
561                 -P $(@).vars -O $(@) $(@F:%.mog=%.mf) \
562                 $(PM_TRANSFORMS) synthetic
563         $(PKGDEBUG)eval REPO=redist PKGSTAT=current '$(CAT) -s $(@).vars'; \
564         if [ -f $(@) ]; then \
565                 $(LN) -s $(@F) \
566                         $(PDIR)/$(@F:%.mog=%).metadata.$$PKGSTAT.$$REPO; \
567         else \
568                 $(TOUCH) $(@); \
569         fi
570         $(PKGDEBUG)$(RM) $(@).vars

572 $(DEP_SYNTH_PKGS): $$(@:%.dep=%.mog)
573         @print "Skipping dependency generation for $(@F:%.dep=%)"
574         $(PKGDEBUG)$(CP) $(@:%.dep=%.mog) $(@)

576 clean:

578 clobber: clean
579         $(RM) -r $(CLOBBERFILES)

581 #
582 # This rule assumes that all links in the $PKGSTAT directories
583 # point to valid manifests, and will fail the make run if one
584 # does not contain an fmri.
585 #
586 # We do this in the BEGIN action instead of using pattern matching
587 # because we expect the fmri to be at or near the first line of each input
588 # file, and this way lets us avoid reading the rest of the file after we
589 # find what we need.
```



```
590 #
591 # We keep track of a failure to locate an fmri, so we can fail the
592 # make run, but we still attempt to process each package in the
593 # repo/pkgstat-specific subdir, in hopes of maybe giving some
594 # additional useful info.
595 #
596 # The protolist is used for bfu archive creation, which may be invoked
597 # interactively by the user.  Both protolist and PKGLISTS targets
598 # depend on $(PROC_PKGS), but protolist builds them recursively.
599 # To avoid collisions, we insert protolist into the dependency chain
600 # here.  This has two somewhat subtle benefits: it allows bfu archive
601 # creation to work correctly, even when -a was not part of NIGHTLY_OPTIONS,
602 # and it ensures that a protolist file here will always correspond to the
603 # contents of the processed manifests, which can vary depending on build
604 # environment.
605 #
606 $(PKGLISTS): $(PROC_PKGS)
607         $(PKGDEBUG)sdotr=$(@F:packages.%=%); \
608         r=$${sdotr%.+(?)}; s=$${sdotr#+(?).}; \
609         print "Generating $$r $$s package list"; \
610         $(RM) $(@); $(TOUCH) $(@); \
611         $(NAWK) 'BEGIN { \
612                 if (ARGC < 2) { \
613                         exit; \
614                 } \
615                 retcode = 0; \
616                 for (i = 1; i < ARGC; i++) { \
617                         do { \
618                                 e = getline f < ARGV[i]; \
619                         } while ((e == 1) && (f !~ /name=pkg.fmri/)); \
620                         close(ARGV[i]); \
621                         if (e == 1) { \
622                                 l = split(f, a, "="); \
623                                 print "depend fmri=" a[l], \
624                                         "type=$$(PKGDEP_TYPE)"; \
625                         } else { \
626                                 print "no fmri in " ARGV[i] >> "/dev/stderr"; \
627                                 retcode = 2; \
628                         } \
629                 } \
630                 exit retcode; \
631         }' 'find $(PDIR) -type l -a \( $(PKGS:%=-name %.metadata.$$s.$$r -o) \
632                 -name NOSUCHFILE \)' >> $(@)

634 #
635 # rules to validate proto area against manifests, check for safe
636 # file permission modes, and generate a faux proto list
637 #
638 # For the check targets, the dependencies on $(PROC_PKGS) is specified
639 # as a subordinate make process in order to suppress output.
640 #
641 makesilent:
642         @$(MAKE) -e $(PROC_PKGS) PKGMACH=$(PKGMACH) \
643                 SUPPRESSPKGDEP=$(SUPPRESSPKGDEP) > /dev/null

645 #
646 # The .lics files were created during pkgmogrification, and list the
647 # set of licenses to pull from $SRC for each package.  Because
648 # licenses may be duplicated between packages, we uniquify them as
649 # well as aggregating them here.
650 #
651 license-list: makesilent
652         $(PKGDEBUG)( for l in 'cat $(PROC_PKGS:%.mog=%.lics)'; \
653                 do print $$l; done ) | sort -u > $@

655 #
```

```
656 # Staging the license and description files in the proto area allows
657 # us to do proper unreferenced file checking of both license and
658 # description files without blanket exceptions, and to pull license
659 # content without reference to $CODEMGR_WS during publication.
660 #
661 stage-licenses: license-list FRC
662         $(PKGDEBUG)$(MAKE) -e -f Makefile.lic \
663                 PKGDEBUG=$(PKGDEBUG) LICROOT=$(PKGROOT)/licenses \
664                 '$(NAWK) '{ \
665                         print "$(PKGROOT)/licenses/" $$0; \
666                         print "$(PKGROOT)/licenses/" $$0 ".descrip"; \
667                 }' license-list' > /dev/null;

669 protocmp: makesilent
670         @validate_pkg -a $(PKGMACH) -v \
671                 $(EXCEPTIONS:%=-e $(CODEMGR_WS)/exception_lists/%) \
672                 -m $(PDIR) -p $(PKGROOT) -p $(TOOLSROOT)

674 pmodes: makesilent
675         @validate_pkg -a $(PKGMACH) -M -m $(PDIR) \
676                 -e $(CODEMGR_WS)/exception_lists/pmodes

678 check: protocmp pmodes

680 protolist: proto_list_$(PKGMACH)

682 proto_list_$(PKGMACH): $(PROC_PKGS)
683         @validate_pkg -a $(PKGMACH) -L -m $(PDIR) > $(@)

685 $(PROC_PKGS): $(PDIR)

687 #
688 # This is a convenience target to allow package names to function as
689 # build targets.  Generally, using it is only useful when iterating on
690 # development of a manifest.
691 #
692 # When processing a manifest, use the basename (without extension) of
693 # the package.  When publishing, use the basename with a ".pub"
694 # extension.
695 #
696 # Other than during manifest development, the preferred usage is to
697 # avoid these targets and override PKGS on the make command line and
698 # use the provided all and install targets.
699 #
700 $(PKGS) $(SYNTH_PKGS): $(PDIR)/$$(@:%=%.mog)

702 $(PKGS:%=%.pub) $(SYNTH_PKGS:%=%.pub): $(PDIR)/$$(@)

704 #
705 # This is a convenience target to resolve dependencies without publishing
706 # packages.
707 #
708 gendeps: $(PDIR)/gendeps

710 #
711 # These are convenience targets for cross-platform packaging.  If you
712 # want to build any of "the normal" targets for a different
713 # architecture, simply use "arch/target" as your build target.
714 #
715 # Since the most common use case for this is "install," the architecture
716 # specific install targets have been further abbreviated to elide "/install."
717 #
718 i386/% sparc/%:
719         $(MAKE) -e $(@F) PKGMACH=$(@D) SUPPRESSPKGDEP=$(SUPPRESSPKGDEP)

721 i386 sparc: $$(@)/install
```



```
723 FRC:
```

```
**********************************************************
    3962 Mon Feb  9 11:03:03 2015
new/usr/src/pkg/manifests/runtime-perl-module-sun-solaris.mf
4863 illumos-gate can't be built with fresh perl versions
**********************************************************
#
# 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.
#

$(i386_ONLY)<transform file dir path=.*PLAT.* -> edit path PLAT i86pc>
$(sparc_ONLY)<transform file dir path=.*PLAT.* -> edit path PLAT sun4>

<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
dir path=usr/perl5/$(PERL_VERSION)/lib/PLAT-solaris-64int
dir path=usr/perl5/$(PERL_VERSION)/lib/PLAT-solaris-64int/Sun
dir path=usr/perl5/$(PERL_VERSION)/lib/PLAT-solaris-64int/Sun/Solaris
dir path=usr/perl5/$(PERL_VERSION)/lib/PLAT-solaris-64int/auto
dir path=usr/perl5/$(PERL_VERSION)/lib/PLAT-solaris-64int/auto/Sun
dir path=usr/perl5/$(PERL_VERSION)/lib/PLAT-solaris-64int/auto/Sun/Solaris
dir \
```



```
    path=usr/perl5/$(PERL_VERSION)/lib/PLAT-solaris-64int/auto/Sun/Solaris/Intrs
dir \
    path=usr/perl5/$(PERL_VERSION)/lib/PLAT-solaris-64int/auto/Sun/Solaris/Kstat
dir \
    path=usr/perl5/$(PERL_VERSION)/lib/PLAT-solaris-64int/auto/Sun/Solaris/Lgrp
dir \
    path=usr/perl5/$(PERL_VERSION)/lib/PLAT-solaris-64int/auto/Sun/Solaris/Proje
dir \
    path=usr/perl5/$(PERL_VERSION)/lib/PLAT-solaris-64int/auto/Sun/Solaris/Task
dir \
    path=usr/perl5/$(PERL_VERSION)/lib/PLAT-solaris-64int/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/$(PERL_ARCH)/auto/Sun/Solaris/Intrs/Intrs
    path=usr/perl5/$(PERL_VERSION)/lib/PLAT-solaris-64int/Sun/Solaris/Intrs.pm
file \
    path=usr/perl5/$(PERL_VERSION)/lib/$(PERL_ARCH)/auto/Sun/Solaris/Kstat/Kstat
    path=usr/perl5/$(PERL_VERSION)/lib/PLAT-solaris-64int/Sun/Solaris/Kstat.pm
file path=usr/perl5/$(PERL_VERSION)/lib/PLAT-solaris-64int/Sun/Solaris/Lgrp.pm
file \
    path=usr/perl5/$(PERL_VERSION)/lib/$(PERL_ARCH)/auto/Sun/Solaris/Lgrp/Lgrp.s
    path=usr/perl5/$(PERL_VERSION)/lib/PLAT-solaris-64int/Sun/Solaris/Project.pm
file path=usr/perl5/$(PERL_VERSION)/lib/PLAT-solaris-64int/Sun/Solaris/Task.pm
file \
    path=usr/perl5/$(PERL_VERSION)/lib/$(PERL_ARCH)/auto/Sun/Solaris/Project/Pro
    path=usr/perl5/$(PERL_VERSION)/lib/PLAT-solaris-64int/Sun/Solaris/Utils.pm
file \
    path=usr/perl5/$(PERL_VERSION)/lib/$(PERL_ARCH)/auto/Sun/Solaris/Task/Task.s
    path=usr/perl5/$(PERL_VERSION)/lib/PLAT-solaris-64int/auto/Sun/Solaris/Intrs
file \
    path=usr/perl5/$(PERL_VERSION)/lib/$(PERL_ARCH)/auto/Sun/Solaris/Utils/Utils
    path=usr/perl5/$(PERL_VERSION)/lib/PLAT-solaris-64int/auto/Sun/Solaris/Kstat
file \
    path=usr/perl5/$(PERL_VERSION)/lib/PLAT-solaris-64int/auto/Sun/Solaris/Lgrp/
file \
    path=usr/perl5/$(PERL_VERSION)/lib/PLAT-solaris-64int/auto/Sun/Solaris/Proje
file \
    path=usr/perl5/$(PERL_VERSION)/lib/PLAT-solaris-64int/auto/Sun/Solaris/Task/
file \
    path=usr/perl5/$(PERL_VERSION)/lib/PLAT-solaris-64int/auto/Sun/Solaris/Utils
file path=usr/perl5/$(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
license usr/src/cmd/perl/THIRDPARTYLICENSE \
    license=usr/src/cmd/perl/THIRDPARTYLICENSE
```

```
**********************************************************
    8808 Mon Feb  9 11:03:03 2015
new/usr/src/tools/env/illumos.sh
4863 illumos-gate can't be built with fresh perl versions
**********************************************************
```

```
#
# 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) 2005, 2010, Oracle and/or its affiliates. All rights reserved.
# Copyright 2010, 2011 Nexenta Systems, Inc.  All rights reserved.
# Copyright 2012 Joshua M. Clulow <josh@sysmgr.org>
# Copyright 2015, OmniTI Computer Consulting, Inc. All rights reserved.
#

# Configuration variables for the runtime environment of the nightly
# build script and other tools for construction and packaging of
# releases.
# This example is suitable for building an illumos workspace, which
# will contain the resulting archives. It is based off the onnv
# release. It sets NIGHTLY_OPTIONS to make nightly do:
#       DEBUG build only (-D, -F)
#       do not bringover from the parent (-n)
#       runs 'make check' (-C)
#       checks for new interfaces in libraries (-A)
#       runs lint in usr/src (-l plus the LINTDIRS variable)
#       sends mail on completion (-m and the MAILTO variable)
#       creates packages for PIT/RE (-p)
#       checks for changes in ELF runpaths (-r)
#       build and use this workspace's tools in $SRC/tools (-t)
#
# - This file is sourced by "bldenv.sh" and "nightly.sh" and should not
#   be executed directly.
# - This script is only interpreted by ksh93 and explicitly allows the
#   use of ksh93 language extensions.
#
export NIGHTLY_OPTIONS='-FnCDAlmprt'

#
# -- PLEASE READ THIS --
#
# The variables  GATE and CODEMGR_WS must always be customised to
# match your workspace/gate location!!
#
# -- PLEASE READ THIS --
#

# This is a variable for the rest of the script - GATE doesn't matter to
# nightly itself
export GATE='testws'
```



```
# CODEMGR_WS - where is your workspace at (or what should nightly name it)
export CODEMGR_WS="$HOME/ws/$GATE"

# Maximum number of dmake jobs.  The recommended number is 2 + NCPUS,
# where NCPUS is the number of logical CPUs on your build system.
function maxjobs
{
        nameref maxjobs=$1
        integer ncpu
        integer -r min_mem_per_job=512 # minimum amount of memory for a job

        ncpu=$(builtin getconf ; getconf 'NPROCESSORS_ONLN')
        (( maxjobs=ncpu + 2 ))

        # Throttle number of parallel jobs launched by dmake to a value which
        # gurantees that all jobs have enough memory. This was added to avoid
        # excessive paging/swapping in cases of virtual machine installations
        # which have lots of CPUs but not enough memory assigned to handle
        # that many parallel jobs
        if [[ $(/usr/sbin/prtconf 2>'/dev/null') == ~(E)Memory\ size:\ ([[:digit
                integer max_jobs_per_memory # parallel jobs which fit into physi
                integer physical_memory # physical memory installed

                # The array ".sh.match" contains the contents of capturing
                # brackets in the last regex, .sh.match[1] will contain
                # the value matched by ([[:digit:]]+), i.e. the amount of
                # memory installed
                physical_memory="10#${.sh.match[1]}"

                ((
                        max_jobs_per_memory=round(physical_memory/min_mem_per_jo
                        maxjobs=fmax(2, fmin(maxjobs, max_jobs_per_memory))
                ))
        fi

        return 0
}

maxjobs DMAKE_MAX_JOBS # "DMAKE_MAX_JOBS" passed as ksh(1) name reference
export DMAKE_MAX_JOBS

# path to onbld tool binaries
ONBLD_BIN='/opt/onbld/bin'

# PARENT_WS is used to determine the parent of this workspace. This is
# for the options that deal with the parent workspace (such as where the
# proto area will go).
export PARENT_WS=''

# CLONE_WS is the workspace nightly should do a bringover from.
export CLONE_WS='ssh://anonhg@hg.illumos.org/illumos-gate'

# The bringover, if any, is done as STAFFER.
# Set STAFFER to your own login as gatekeeper or developer
# The point is to use group "staff" and avoid referencing the parent
# workspace as root.
# Some scripts optionally send mail messages to MAILTO.
#
export STAFFER="$LOGNAME"
export MAILTO="$STAFFER"

# If you wish the mail messages to be From: an arbitrary address, export
# MAILFROM.
#export MAILFROM="user@example.com"
```

```
128 # The project (see project(4)) under which to run this build.  If not
129 # specified, the build is simply run in a new task in the current project.
130 export BUILD_PROJECT=''

132 # You should not need to change the next three lines
133 export ATLOG="$CODEMGR_WS/log"
134 export LOGFILE="$ATLOG/nightly.log"
135 export MACH="$(uname -p)"

137 #
138 #  The following two macros are the closed/crypto binaries.  Once
139 #  Illumos has totally freed itself, we can remove these references.
140 #
141 # Location of encumbered binaries.
142 export ON_CLOSED_BINS="$CODEMGR_WS/closed"
143 # Location of signed cryptographic binaries.
144 export ON_CRYPTO_BINS="$CODEMGR_WS/on-crypto.$MACH.tar.bz2"

146 # REF_PROTO_LIST - for comparing the list of stuff in your proto area
147 # with. Generally this should be left alone, since you want to see differences
148 # from your parent (the gate).
149 #
150 export REF_PROTO_LIST="$PARENT_WS/usr/src/proto_list_${MACH}"


153 export ROOT="$CODEMGR_WS/proto/root_${MACH}"
154 export SRC="$CODEMGR_WS/usr/src"
155 export MULTI_PROTO="no"

157 #
158 #       build environment variables, including version info for mcs, motd,
159 # motd, uname and boot messages. Mostly you shouldn't change this except
160 # when the release slips (nah) or you move an environment file to a new
161 # release
162 #
163 export VERSION="$GATE"

165 #
166 # the RELEASE and RELEASE_DATE variables are set in Makefile.master;
167 # there might be special reasons to override them here, but that
168 # should not be the case in general
169 #
170 # export RELEASE='5.11'
171 # export RELEASE_DATE='October 2007'

173 # proto area in parent for optionally depositing a copy of headers and
174 # libraries corresponding to the protolibs target
175 # not applicable given the NIGHTLY_OPTIONS
176 #
177 export PARENT_ROOT="$PARENT_WS/proto/root_$MACH"
178 export PARENT_TOOLS_ROOT="$PARENT_WS/usr/src/tools/proto/root_$MACH-nd"

180 # Package creation variables.  You probably shouldn't change these,
181 # either.
182 #
183 # PKGARCHIVE determines where the repository will be created.
184 #
185 # PKGPUBLISHER_REDIST controls the publisher setting for the repository.
186 #
187 export PKGARCHIVE="${CODEMGR_WS}/packages/${MACH}/nightly"
188 # export PKGPUBLISHER_REDIST='on-redist'

190 # Package manifest format version.
191 export PKGFMT_OUTPUT='v1'

193 # we want make to do as much as it can, just in case there's more than
```



```
194 # one problem.
195 export MAKEFLAGS='k'

197 # Magic variable to prevent the devpro compilers/teamware from sending
198 # mail back to devpro on every use.
199 export UT_NO_USAGE_TRACKING='1'

201 # Build tools - don't change these unless you know what you're doing.  These
202 # variables allows you to get the compilers and onbld files locally or
203 # through cachefs.  Set BUILD_TOOLS to pull everything from one location.
204 # Alternately, you can set ONBLD_TOOLS to where you keep the contents of
205 # SUNWonbld and SPRO_ROOT to where you keep the compilers.  SPRO_VROOT
206 # exists to make it easier to test new versions of the compiler.
207 export BUILD_TOOLS='/opt'
208 #export ONBLD_TOOLS='/opt/onbld'
209 export SPRO_ROOT='/opt/SUNWspro'
210 export SPRO_VROOT="$SPRO_ROOT"

212 # This goes along with lint - it is a series of the form "A [y|n]" which
213 # means "go to directory A and run 'make lint'" Then mail me (y) the
214 # difference in the lint output. 'y' should only be used if the area you're
215 # linting is actually lint clean or you'll get lots of mail.
216 # You shouldn't need to change this though.
217 #export LINTDIRS="$SRC y"

219 # Set this flag to 'n' to disable the automatic validation of the dmake
220 # version in use.  The default is to check it.
221 #CHECK_DMAKE='y'

223 # Set this flag to 'n' to disable the use of 'checkpaths'.  The default,
224 # if the 'N' option is not specified, is to run this test.
225 #CHECK_PATHS='y'

227 # POST_NIGHTLY can be any command to be run at the end of nightly.  See
228 # nightly(1) for interactions between environment variables and this command.
229 #POST_NIGHTLY=

231 # Uncomment this to disable support for SMB printing.
232 # export ENABLE_SMB_PRINTING='#'

234 # If your distro uses certain versions of Perl, make sure either Makefile.master
235 # contains your new defaults OR your .env file sets them.
236 # These are how you would override for building on OmniOS r151012, for example.
237 #export PERL_VERSION=5.16.1
238 #export PERL_ARCH=i86pc-solaris-thread-multi-64int
239 #export PERL_PKGVERS=-5161
```