

```

*****
34577 Mon Jan 20 11:27:42 2014
new/usr/src/Makefile.master
4457 we apparently change .comment of almost every userland object
*****
1 #
2 # CDDL HEADER START
3 #
4 # The contents of this file are subject to the terms of the
5 # Common Development and Distribution License (the "License").
6 # You may not use this file except in compliance with the License.
7 #
8 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
9 # or http://www.opensolaris.org/os/licensing.
10 # See the License for the specific language governing permissions
11 # and limitations under the License.
12 #
13 # When distributing Covered Code, include this CDDL HEADER in each
14 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
15 # If applicable, add the following below this CDDL HEADER, with the
16 # fields enclosed by brackets "[]" replaced with your own identifying
17 # information: Portions Copyright [yyyy] [name of copyright owner]
18 #
19 # CDDL HEADER END
20 #
21 #
22 #
23 # Copyright (c) 1989, 2010, Oracle and/or its affiliates. All rights reserved.
24 # Copyright (c) 2012 by Delphix. All rights reserved.
25 #
26 #
27 #
28 # Makefile.master, global definitions for system source
29 #
30 ROOT=          /proto
31 #
32 #
33 # Adjunct root, containing an additional proto area to be used for headers
34 # and libraries.
35 #
36 ADJUNCT_PROTO=
37 #
38 #
39 # Adjunct for building things that run on the build machine.
40 #
41 NATIVE_ADJUNCT= /usr
42 #
43 #
44 # RELEASE_BUILD should be cleared for final release builds.
45 # NOT_RELEASE_BUILD is exactly what the name implies.
46 #
47 # __GNUC toggles the building of ON components using gcc and related tools.
48 # Normally set to '#', set it to '' to do gcc build.
49 #
50 # The declaration POUND_SIGN is always '#'. This is needed to get around the
51 # make feature that '#' is always a comment delimiter, even when escaped or
52 # quoted. We use this macro expansion method to get POUND_SIGN rather than
53 # always breaking out a shell because the general case can cause a noticeable
54 # slowdown in build times when so many Makefiles include Makefile.master.
55 #
56 # While the majority of users are expected to override the setting below
57 # with an env file (via nightly or bldenv), if you aren't building that way
58 # (ie, you're using "ws" or some other bootstrapping method) then you need
59 # this definition in order to avoid the subshell invocation mentioned above.
60 #

```

```

62 PRE_POUND=          pre\#
63 POUND_SIGN=         $(PRE_POUND:pre\#=%)
64 #
65 NOT_RELEASE_BUILD=
66 RELEASE_BUILD=      $(POUND_SIGN)
67 $(RELEASE_BUILD)NOT_RELEASE_BUILD= $(POUND_SIGN)
68 PATCH_BUILD=        $(POUND_SIGN)
69 #
70 # SPARC_BLD is '#' for an Intel build.
71 # INTEL_BLD is '#' for a Sparc build.
72 SPARC_BLD_1=        $(MACH:i386=$(POUND_SIGN))
73 SPARC_BLD=          $(SPARC_BLD_1:sparc=)
74 INTEL_BLD_1=        $(MACH:sparc=$(POUND_SIGN))
75 INTEL_BLD=          $(INTEL_BLD_1:i386=)
76 #
77 # The variables below control the compilers used during the build.
78 # There are a number of permutations.
79 #
80 # __GNUC and __SUNC control (and indicate) the primary compiler.  Whichever
81 # one is not POUND_SIGN is the primary, with the other as the shadow.  They
82 # may also be used to control entirely compiler-specific Makefile assignments.
83 # __SUNC and Sun Studio are the default.
84 #
85 # __GNUC64 indicates that the 64bit build should use the GNU C compiler.
86 # There is no Sun C analogue.
87 #
88 # The following version-specific options are operative regardless of which
89 # compiler is primary, and control the versions of the given compilers to be
90 # used.  They also allow compiler-version specific Makefile fragments.
91 #
92 #
93 __GNUC=              $(POUND_SIGN)
94 $(__GNUC)__SUNC=    $(POUND_SIGN)
95 __GNUC64=           $(__GNUC)
96 #
97 # CLOSED is the root of the tree that contains source which isn't released
98 # as open source
99 CLOSED=              $(SRC)/../closed
100 #
101 # BUILD_TOOLS is the root of all tools including compilers.
102 # ONBLD_TOOLS is the root of all the tools that are part of SUNWonbld.
103 #
104 BUILD_TOOLS=         /ws/onnv-tools
105 ONBLD_TOOLS=         $(BUILD_TOOLS)/onbld
106 #
107 JAVA_ROOT=          /usr/java
108 #
109 SFW_ROOT=            /usr/sfw
110 SFWINCDIR=          $(SFW_ROOT)/include
111 SFWLIBDIR=           $(SFW_ROOT)/lib
112 SFWLIBDIR64=        $(SFW_ROOT)/lib/$(MACH64)
113 #
114 GCC_ROOT=            /opt/gcc/4.4.4
115 GCCLIBDIR=          $(GCC_ROOT)/lib
116 GCCLIBDIR64=        $(GCC_ROOT)/lib/$(MACH64)
117 #
118 DOCBOOK_XSL_ROOT=  /usr/share/sgml/docbook/xsl-stylesheets
119 #
120 RPCGEN=              /usr/bin/rpcgen
121 STABS=               $(ONBLD_TOOLS)/bin/$(MACH)/stabs
122 ELFEXTRACT=         $(ONBLD_TOOLS)/bin/$(MACH)/elfextract
123 MBH_PATCH=          $(ONBLD_TOOLS)/bin/$(MACH)/mbh_patch
124 ECHO=                echo
125 INS=                 install
126 TRUE=               true
127 SYMLINK=            /usr/bin/ln -s

```

new/usr/src/Makefile.master

```

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

181 FILEMODE= 644
182 DIRMODE= 755

184 #
185 # The version of the patch makeup table optimized for build-time use. Used
186 # during patch builds only.
187 $(PATCH_BUILD)PMTMO_FILE=$(SRC)/patch_makeup_table.mo

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

193 # For stylistic checks

```

3

new/usr/src/Makefile.master

```

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

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

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

212 DOT_C_CHECK= \
213     @$(ECHO) "checking $<"; $(CSTYLE) $< $(CSTYLE_TAIL)

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

222 INS.file= $(RM) $@; $(INS) -s -m $(FILEMODE) -f $(@D) $<
223 INS.dir= $(INS) -s -d -m $(DIRMODE) $@
224 # installs and renames at once
225 #
226 INS.rename= $(INS.file); $(MV) $@D/$(<F) $@

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

233 #
234 # Python bakes the mtime of the .py file into the compiled .pyc and
235 # rebuilds if the baked-in mtime != the mtime of the source file
236 # (rather than only if it's less than), thus when installing python
237 # files we must make certain to not adjust the mtime of the source
238 # (.py) file.
239 #
240 INS.pyfile= $(INS.file); $(TOUCH) -r $< $@

242 # MACH must be set in the shell environment per uname -p on the build host
243 # More specific architecture variables should be set in lower makefiles.
244 #
245 # MACH64 is derived from MACH, and BUILD64 is set to '#' for
246 # architectures on which we do not build 64-bit versions.
247 # (There are no such architectures at the moment.)
248 #
249 # Set BUILD64=# in the environment to disable 64-bit amd64
250 # builds on i386 machines.

252 MACH64_1= $(MACH:sparc=sparcv9)
253 MACH64= $(MACH64_1:i386=amd64)

255 MACH32_1= $(MACH:sparc=sparcv7)
256 MACH32= $(MACH32_1:i386=i86)

258 sparc_BUILD64=
259 i386_BUILD64=

```

4

```

260 BUILD64=          $($(MACH)_BUILD64)

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

270 #
271 # C compiler verbose mode. This is so we can enable it globally,
272 # but turn it off in the lower level makefiles of things we cannot
273 # (or aren't going to) fix.
274 #
275 CCVERBOSE=         -v

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

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

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

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

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

```

```

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

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

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

356 SAVEARGS=          -Wu,-save_args
357 amd64_STAND_FLAGS  += $(SAVEARGS)

359 STAND_FLAGS_32 = $($(MACH)_STAND_FLAGS)
360 STAND_FLAGS_64 = $($(MACH64)_STAND_FLAGS)

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

371 #
372 # turn warnings into errors (C)
373 CERRWARN = -errtags=yes -errwarn=%all
374 CERRWARN += -erroff=E_EMPTY_TRANSLATION_UNIT
375 CERRWARN += -erroff=E_STATEMENT_NOT_REACHED

377 CERRWARN += -_gcc=-Wno-missing-braces
378 CERRWARN += -_gcc=-Wno-sign-compare
379 CERRWARN += -_gcc=-Wno-unknown-pragmas
380 CERRWARN += -_gcc=-Wno-unused-parameter
381 CERRWARN += -_gcc=-Wno-missing-field-initializers

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

386 # DEBUG v. -nd make for frequent unused variables, empty conditions, etc. in
387 # -nd builds
388 $(RELEASE_BUILD)CERRWARN += -_gcc=-Wno-unused
389 $(RELEASE_BUILD)CERRWARN += -_gcc=-Wno-empty-body

391 #

```

```

392 # turn warnings into errors (C++)
393 CCERRWARN=          -xwe

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

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

409 sparc_ASFLAGS=      $(sparc_AS_XARCH)
410 sparcv9_ASFLAGS=    $(sparcv9_AS_XARCH)
411 i386_ASFLAGS=       $(i386_AS_XARCH)
412 amd64_ASFLAGS=     $(amd64_AS_XARCH)

414 #
415 sparc_COPTFLAG=     -xO3
416 sparcv9_COPTFLAG=  -xO3
417 i386_COPTFLAG=     -O
418 amd64_COPTFLAG=    -xO3

420 COPTFLAG=           $($(MACH)_COPTFLAG)
421 COPTFLAG64=        $($(MACH64)_COPTFLAG)

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

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

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

439 #
440 # Default debug format for Sun Studio 11 is dwarf, so force it to
441 # generate stabs.
442 #
443 DEBUGFORMAT=       -xdebugformat=stabs

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

452 CTF_FLAGS_sparcv9 = $(CTF_FLAGS_sparc)
453 CTF_FLAGS_amd64   = $(CTF_FLAGS_i386)

455 # Sun Studio produces broken userland code when saving arguments.
456 $(__GNUC)CTF_FLAGS_amd64 += $(SAVEARGS)

```

```

458 CTF_FLAGS_32      = $(CTF_FLAGS_$(MACH)) $(DEBUGFORMAT)
459 CTF_FLAGS_64      = $(CTF_FLAGS_$(MACH64)) $(DEBUGFORMAT)
460 CTF_FLAGS          = $(CTF_FLAGS_32)

462 #
463 # Flags used with genoffsets
464 #
465 GOFLAGS = -noecho \
466           $(CALLSYMS) \
467           $(CDWARFSTR)

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

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

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

483 SPACEFLAG           = $($(MACH)_SPACEFLAG)
484 SPACEFLAG64         = $($(MACH64)_SPACEFLAG)

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

496 IROPTFLAG           = $($(MACH)_IROPTFLAG)
497 IROPTFLAG64         = $($(MACH64)_IROPTFLAG)

499 sparc_XREGSFLAG      = -xregs=no%appl
500 sparcv9_XREGSFLAG   = -xregs=no%appl
501 i386_XREGSFLAG       =
502 amd64_XREGSFLAG      =

504 XREGSFLAG           = $($(MACH)_XREGSFLAG)
505 XREGSFLAG64         = $($(MACH64)_XREGSFLAG)

507 # dmake SOURCEDEBUG=yes ... enables source-level debugging information, and
508 # avoids stripping it.
509 SOURCEDEBUG          = $(POUND_SIGN)
510 SRCDBGBLD            = $(SOURCEDEBUG:yes=)

512 #
513 # These variables are intended ONLY for use by developers to safely pass extra
514 # flags to the compilers without unintentionally overriding Makefile-set
515 # flags.  They should NEVER be set to any value in a Makefile.
516 #
517 # They come last in the associated FLAGS variable such that they can
518 # explicitly override things if necessary, there are gaps in this, but it's
519 # the best we can manage.
520 #
521 CUSERFLAGS           =
522 CUSERFLAGS64         = $(CUSERFLAGS)
523 CCUSERFLAGS          =

```

```

524 CCUSERFLAGS64      = $(CCUSERFLAGS)

526 CSOURCEDEBUGFLAGS =
527 CCSOURCEDEBUGFLAGS =
528 $(SRCDGBLD)CSOURCEDEBUGFLAGS = -g -xs
529 $(SRCDGBLD)CCSOURCEDEBUGFLAGS = -g -xs

531 CFLAGS=             $(COPTFLAG) $($ (MACH)_CFLAGS) $(SPACEFLAG) $(CCMODE) \
532 $(ILDOFF) $(CERRWARN) $(C99MODE) $(CCUNBOUND) $(IROPTFLAG) \
533 $(CGLOBALSTATIC) $(CCNOAUTOINLINE) $(CSOURCEDEBUGFLAGS) \
534 $(USERFLAGS)
535 CFLAGS64=           $(COPTFLAG64) $($ (MACH64)_CFLAGS) $(SPACEFLAG64) $(CCMODE64) \
536 $(ILDOFF) $(CERRWARN) $(C99MODE) $(CCUNBOUND) $(IROPTFLAG64) \
537 $(CGLOBALSTATIC) $(CCNOAUTOINLINE) $(CSOURCEDEBUGFLAGS) \
538 $(USERFLAGS64)
539 #
540 # Flags that are used to build parts of the code that are subsequently
541 # run on the build machine (also known as the NATIVE_BUILD).
542 #
543 NATIVE_CFLAGS=      $(COPTFLAG) $($ (NATIVE_MACH)_CFLAGS) $(CCMODE) \
544 $(ILDOFF) $(CERRWARN) $(C99MODE) $($ (NATIVE_MACH)_CCUNBOUND) \
545 $(IROPTFLAG) $(CGLOBALSTATIC) $(CCNOAUTOINLINE) \
546 $(CSOURCEDEBUGFLAGS) $(USERFLAGS)

548 DTEXTDOM=-DTEXT_DOMAIN="\$(TEXT_DOMAIN)" # For messaging.
549 DTS_ERRNO=-D_TS_ERRNO
550 CPPFLAGS.master=$(DTEXTDOM) $(DTS_ERRNO) \
551 $(ENVCPPFLAGS1) $(ENVCPPFLAGS2) $(ENVCPPFLAGS3) $(ENVCPPFLAGS4) \
552 $(ADJUNCT_PROTO:%=-I%/usr/include)
553 CPPFLAGS.native=$(ENVCPPFLAGS1) $(ENVCPPFLAGS2) $(ENVCPPFLAGS3) \
554 $(ENVCPPFLAGS4) -I$(NATIVE_ADJUNCT)/include
555 CPPFLAGS=          $(CPPFLAGS.master)
556 AS_CPPFLAGS=       $(CPPFLAGS.master)
557 JAVAFLAGS=         -deprecation

559 #
560 # For source message catalogue
561 #
562 .SUFFIXES: $(SUFFIXES) .i .po
563 MSGROOT= $(ROOT)/catalog
564 MSGDOMAIN= $(MSGROOT)/$(TEXT_DOMAIN)
565 MSGDOMAINPOFILE = $(MSGDOMAIN)/$(POFILE)
566 DCMSGDOMAIN= $(MSGROOT)/LC_TIME/$(TEXT_DOMAIN)
567 DCMSGDOMAINPOFILE = $(DCMSGDOMAIN)/$(DCFILE:.dc=.po)

569 CLOBBERFILES += $(POFILE) $(POFILES)
570 COMPILE.cpp= $(CC) -E -C $(CFLAGS) $(CPPFLAGS)
571 XGETTEXT= /usr/bin/xgettext
572 XGETTEXTFLAGS= -c TRANSLATION_NOTE
573 GNUXGETTEXT= /usr/gnu/bin/xgettext
574 GNUXGETTEXTFLAGS= --add-comments=TRANSLATION_NOTE --keyword= \
575 --strict --no-location --omit-header
576 BUILD.po= $(XGETTEXT) $(XGETTEXTFLAGS) -d $(<F) $(<.i ;\
577 $(RM) $@ ;\
578 $(SED) "/^domain/d" < $(<F).po > $@ ;\
579 $(RM) $(<F).po $(<.i

581 #
582 # This is overwritten by local Makefile when PROG is a list.
583 #
584 POFILE= $(PROG).po

586 sparc_CCFLAGS=      -cg92 -compat=4 \
587 -Option ccfe -messages=no%anachronism \
588 $(CCERRWARN)
589 sparcv9_CCFLAGS=    $(sparcv9_XARCH) -dalign -compat=5 \

```

```

590 -Option ccfe -messages=no%anachronism \
591 -Option ccfe -features=no%conststrings \
592 $(CCREGSYM) \
593 $(CCERRWARN)
594 i386_CCFLAGS=       -compat=4 \
595 -Option ccfe -messages=no%anachronism \
596 -Option ccfe -features=no%conststrings \
597 $(CCERRWARN)
598 amd64_CCFLAGS=      $(amd64_XARCH) -compat=5 \
599 -Option ccfe -messages=no%anachronism \
600 -Option ccfe -features=no%conststrings \
601 $(CCERRWARN)

603 sparc_CCOPTFLAG=   -O
604 sparcv9_CCOPTFLAG= -O
605 i386_CCOPTFLAG=    -O
606 amd64_CCOPTFLAG=   -O

608 CCOPTFLAG=          $($ (MACH)_CCOPTFLAG)
609 CCOPTFLAG64=        $($ (MACH64)_CCOPTFLAG)
610 CCFLAGS=             $(CCOPTFLAG) $($ (MACH)_CCFLAGS) $(CCSOURCEDEBUGFLAGS) \
611 $(CCUSERFLAGS)
612 CCFLAGS64=           $(CCOPTFLAG64) $($ (MACH64)_CCFLAGS) $(CCSOURCEDEBUGFLAGS) \
613 $(CCUSERFLAGS64)

615 #
616 #
617 #
618 ELFWRAP_FLAGS =
619 ELFWRAP_FLAGS64 = -64

621 #
622 # Various mapfiles that are used throughout the build, and delivered to
623 # /usr/lib/ld.
624 #
625 MAPFILE.NED_i386 = $(SRC)/common/mapfiles/common/map.noexdata
626 MAPFILE.NED_sparc =
627 MAPFILE.NED = $(MAPFILE.NED_$(MACH))
628 MAPFILE.PGA = $(SRC)/common/mapfiles/common/map.pagealign
629 MAPFILE.NES = $(SRC)/common/mapfiles/common/map.noexstk
630 MAPFILE.FLT = $(SRC)/common/mapfiles/common/map.filter
631 MAPFILE.LEX = $(SRC)/common/mapfiles/common/map.lex.yy

633 #
634 # Generated mapfiles that are compiler specific, and used throughout the
635 # build. These mapfiles are not delivered in /usr/lib/ld.
636 #
637 MAPFILE.NGB_sparc= $(SRC)/common/mapfiles/gen/sparc_cc_map.noexglobs
638 $(__GNUCC64)MAPFILE.NGB_sparc= \
639 $(SRC)/common/mapfiles/gen/sparc_gcc_map.noexglobs
640 MAPFILE.NGB_sparcv9= $(SRC)/common/mapfiles/gen/sparcv9_cc_map.noexglobs
641 $(__GNUCC64)MAPFILE.NGB_sparcv9= \
642 $(SRC)/common/mapfiles/gen/sparcv9_gcc_map.noexglobs
643 MAPFILE.NGB_i386= $(SRC)/common/mapfiles/gen/i386_cc_map.noexglobs
644 $(__GNUCC64)MAPFILE.NGB_i386= \
645 $(SRC)/common/mapfiles/gen/i386_gcc_map.noexglobs
646 MAPFILE.NGB_amd64= $(SRC)/common/mapfiles/gen/amd64_cc_map.noexglobs
647 $(__GNUCC64)MAPFILE.NGB_amd64= \
648 $(SRC)/common/mapfiles/gen/amd64_gcc_map.noexglobs
649 MAPFILE.NGB = $(MAPFILE.NGB_$(MACH))

651 #
652 # A generic interface mapfile name, used by various dynamic objects to define
653 # the interfaces and interposers the object must export.
654 #
655 MAPFILE.INT = mapfile-intf

```

```

657 #
658 # LDLIBS32 can be set in the environment to override the following assignment.
659 # LDLIBS64 can be set to override the assignment made in Makefile.master.64.
660 # These environment settings make sure that no libraries are searched outside
661 # of the local workspace proto area:
662 #     LDLIBS32=-YP,$ROOT/lib:$ROOT/usr/lib
663 #     LDLIBS64=-YP,$ROOT/lib/$MACH64:$ROOT/usr/lib/$MACH64
664 #
665 LDLIBS32 = $(ENVLDLIBS1) $(ENVLDLIBS2) $(ENVLDLIBS3)
666 LDLIBS32 += $(ADJUNCT_PROTO:%=-L%/usr/lib -L%/lib)
667 LDLIBS.cmd = $(LDLIBS32)
668 LDLIBS.lib = $(LDLIBS32)
669 #
670 # Define compilation macros.
671 #
672 COMPILE.c= $(CC) $(CFLAGS) $(CPPFLAGS) -c
673 COMPILE64.c= $(CC) $(CFLAGS64) $(CPPFLAGS) -c
674 COMPILE.cc= $(CCC) $(CCFLAGS) $(CPPFLAGS) -c
675 COMPILE64.cc= $(CCC) $(CCFLAGS64) $(CPPFLAGS) -c
676 COMPILE.s= $(AS) $(ASFLAGS) $(AS_CPPFLAGS)
677 COMPILE64.s= $(AS) $(ASFLAGS) $(MACH64_AS_XARCH) $(AS_CPPFLAGS)
678 COMPILE.d= $(DTRACE) -G -32
679 COMPILE64.d= $(DTRACE) -G -64
680 COMPILE.b= $(ELFWRAP) $(ELFWRAP_FLAGS$(CLASS))
681 COMPILE64.b= $(ELFWRAP) $(ELFWRAP_FLAGS$(CLASS))

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

686 #
687 # Link time macros
688 #
689 CCNEEDED = -lc
690 CCEXTNEEDED = -lcrun -lcstd
691 $(__GNUC)CCNEEDED = -L$(GCCLIBDIR) -lstc++ -lgcc_s
692 $(__GNUC)CCEXTNEEDED = $(CCNEEDED)

694 LINK.c= $(CC) $(CFLAGS) $(CPPFLAGS) $(LDFLAGS)
695 LINK64.c= $(CC) $(CFLAGS64) $(CPPFLAGS) $(LDFLAGS)
696 NORUNPATH= -norunpath -nolib
697 LINK.cc= $(CCC) $(CCFLAGS) $(CPPFLAGS) $(NORUNPATH) \
698 $(LDFLAGS) $(CCNEEDED)
699 LINK64.cc= $(CCC) $(CCFLAGS64) $(CPPFLAGS) $(NORUNPATH) \
700 $(LDFLAGS) $(CCNEEDED)

702 #
703 # lint macros
704 #
705 # Note that the undefine of __PRAGMA_REDEFINE_EXTNAME can be removed once
706 # ON is built with a version of lint that has the fix for 4484186.
707 #
708 ALWAYS_LINT_DEFS = -errtags=yes -s
709 ALWAYS_LINT_DEFS += -erroff=E_PTRDIFF_OVERFLOW
710 ALWAYS_LINT_DEFS += -erroff=E_ASSIGN_NARROW_CONV
711 ALWAYS_LINT_DEFS += -U__PRAGMA_REDEFINE_EXTNAME
712 ALWAYS_LINT_DEFS += $(C99LMODE)
713 ALWAYS_LINT_DEFS += -errsecurity=$(SECLEVEL)
714 ALWAYS_LINT_DEFS += -erroff=E_SEC_CREAT_WITHOUT_EXCL
715 ALWAYS_LINT_DEFS += -erroff=E_SEC_FORBIDDEN_WARN_CREAT
716 # XX64 -- really only needed for amd64 lint
717 ALWAYS_LINT_DEFS += -erroff=E_ASSIGN_INT_TO_SMALL_INT
718 ALWAYS_LINT_DEFS += -erroff=E_CAST_INT_CONST_TO_SMALL_INT
719 ALWAYS_LINT_DEFS += -erroff=E_CAST_INT_TO_SMALL_INT
720 ALWAYS_LINT_DEFS += -erroff=E_CAST_TO_PTR_FROM_INT
721 ALWAYS_LINT_DEFS += -erroff=E_COMP_INT_WITH_LARGE_INT

```

```

722 ALWAYS_LINT_DEFS += -erroff=E_INTEGRAL_CONST_EXP_EXPECTED
723 ALWAYS_LINT_DEFS += -erroff=E_PASS_INT_TO_SMALL_INT
724 ALWAYS_LINT_DEFS += -erroff=E_PTR_CONV_LOSES_BITS

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

730 SECLEVEL= core
731 LINT.c= $(LINT) $(ONLY_LINT_DEFS) $(LINTFLAGS) $(CPPFLAGS) \
732 $(ALWAYS_LINT_DEFS)
733 LINT64.c= $(LINT) $(ONLY_LINT_DEFS) $(LINTFLAGS64) $(CPPFLAGS) \
734 $(ALWAYS_LINT_DEFS)
735 LINT.s= $(LINT.c)

737 # For some future builds, NATIVE_MACH and MACH might be different.
738 # Therefore, NATIVE_MACH needs to be redefined in the
739 # environment as 'uname -p' to override this macro.
740 #
741 # For now at least, we cross-compile amd64 on i386 machines.
742 NATIVE_MACH= $(MACH:amd64=i386)

744 # Define native compilation macros
745 #

747 # Base directory where compilers are loaded.
748 # Defined here so it can be overridden by developer.
749 #
750 SPRO_ROOT= $(BUILD_TOOLS)/SUNWsprio
751 SPRO_VROOT= $(SPRO_ROOT)/SS12
752 GNU_ROOT= $(SFW_ROOT)

754 # Till SS12ul formally becomes the NV CBE, LINT is hard
755 # coded to be picked up from the $SPRO_ROOT/sunstudio12.1/
756 # location. Impacted variables are sparc_LINT, sparcv9_LINT,
757 # i386_LINT, amd64_LINT.
758 # Reset them when SS12ul is rolled out.
759 #

761 # Specify platform compiler versions for languages
762 # that we use (currently only c and c++).
763 #
764 sparc_CC= $(ONBLD_TOOLS)/bin/$(MACH)/cw -_cc
765 $(__GNUC)sparc_CC= $(ONBLD_TOOLS)/bin/$(MACH)/cw -_gcc
766 sparc_CCC= $(ONBLD_TOOLS)/bin/$(MACH)/cw -_CC
767 $(__GNUC)sparc_CCC= $(ONBLD_TOOLS)/bin/$(MACH)/cw -_g++
768 sparc_CPP= /usr/ccs/lib/cpp
769 sparc_AS= /usr/ccs/bin/as -xregsym=no
770 sparc_LD= /usr/ccs/bin/ld
771 sparc_LINT= $(SPRO_ROOT)/sunstudio12.1/bin/lint

773 sparcv9_CC= $(ONBLD_TOOLS)/bin/$(MACH)/cw -_cc
774 $(__GNUC64)sparcv9_CC= $(ONBLD_TOOLS)/bin/$(MACH)/cw -_gcc
775 sparcv9_CCC= $(ONBLD_TOOLS)/bin/$(MACH)/cw -_CC
776 $(__GNUC64)sparcv9_CCC= $(ONBLD_TOOLS)/bin/$(MACH)/cw -_g++
777 sparcv9_CPP= /usr/ccs/lib/cpp
778 sparcv9_AS= /usr/ccs/bin/as -xregsym=no
779 sparcv9_LD= /usr/ccs/bin/ld
780 sparcv9_LINT= $(SPRO_ROOT)/sunstudio12.1/bin/lint

782 i386_CC= $(ONBLD_TOOLS)/bin/$(MACH)/cw -_cc
783 $(__GNUC)i386_CC= $(ONBLD_TOOLS)/bin/$(MACH)/cw -_gcc
784 i386_CCC= $(ONBLD_TOOLS)/bin/$(MACH)/cw -_CC
785 $(__GNUC)i386_CCC= $(ONBLD_TOOLS)/bin/$(MACH)/cw -_g++
786 i386_CPP= /usr/ccs/lib/cpp
787 i386_AS= /usr/ccs/bin/as

```

```

788 $(__GNUC)i386_AS=      $(ONBLD_TOOLS)/bin/$(MACH)/aw
789 i386_LD=              /usr/ccs/bin/ld
790 i386_LINT=            $(SPRO_ROOT)/sunstudio12.1/bin/lint

792 amd64_CC=            $(ONBLD_TOOLS)/bin/$(MACH)/cw -_cc
793 $(__GNUC64)amd64_CC=  $(ONBLD_TOOLS)/bin/$(MACH)/cw -_gcc
794 amd64_CCC=          $(ONBLD_TOOLS)/bin/$(MACH)/cw -_CC
795 $(__GNUC64)amd64_CCC= $(ONBLD_TOOLS)/bin/$(MACH)/cw -_g++
796 amd64_CPP=          /usr/ccs/lib/cpp
797 amd64_AS=           $(ONBLD_TOOLS)/bin/$(MACH)/aw
798 amd64_LD=           /usr/ccs/bin/ld
799 amd64_LINT=         $(SPRO_ROOT)/sunstudio12.1/bin/lint

801 NATIVECC=           $($ (NATIVE_MACH)_CC)
802 NATIVECCC=          $($ (NATIVE_MACH)_CCC)
803 NATIVECPP=          $($ (NATIVE_MACH)_CPP)
804 NATIVEAS=           $($ (NATIVE_MACH)_AS)
805 NATIVELD=           $($ (NATIVE_MACH)_LD)
806 NATIVELINT=         $($ (NATIVE_MACH)_LINT)

808 #
809 # Makefile.master.64 overrides these settings
810 #
811 CC=                  $(NATIVECC)
812 CCC=                 $(NATIVECCC)
813 CPP=                 $(NATIVECPP)
814 AS=                  $(NATIVEAS)
815 LD=                  $(NATIVELD)
816 LINT=                $(NATIVELINT)

818 # The real compilers used for this build
819 CW_CC_CMD=           $(CC) -_compiler
820 CW_CCC_CMD=          $(CCC) -_compiler
821 REAL_CC=             $(CW_CC_CMD:sh)
822 REAL_CCC=            $(CW_CCC_CMD:sh)

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

827 BDIRECT=            -Bdirect
828 BDYNAMIC=            -Bdynamic
829 BLOCAL=              -Blocal
830 BNODIRECT=           -Bnodirect
831 BREDUCE=             -Breduce
832 BSTATIC=             -Bstatic

834 ZDEFS=               -zdefs
835 ZDIRECT=             -zdirect
836 ZIGNORE=             -zignore
837 ZINITFIRST=         -zinitfirst
838 ZINTERPOSE=         -zinterpose
839 ZLAZYLOAD=          -zlazyload
840 ZLOADFLTR=          -zloadfltr
841 ZMULDEFS=           -zmuldefs
842 ZNODEFAULTLIB=     -znodefaultlib
843 ZNODEFS=            -znodefs
844 ZNODELETE=         -znodelete
845 ZNODLOPEN=         -znodlopen
846 ZNODUMP=            -znodump
847 ZNOLAZYLOAD=       -znolazyload
848 ZNOLDYNSYM=        -znoldynsym
849 ZNORELOC=          -znoreloc
850 ZNOVERSION=        -znoversion
851 ZRECORD=           -zrecord
852 ZREDLOCSYM=        -zredlocsym
853 ZTEXT=              -ztext

```

```

854 ZVERBOSE=           -zverbose

856 GSHARED=            -G
857 CCMT=               -mt

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

862 sparc_C_PICFLAGS =   -K pic
863 sparcv9_C_PICFLAGS = -K pic
864 i386_C_PICFLAGS =   -K pic
865 amd64_C_PICFLAGS =   -K pic
866 C_PICFLAGS =         $($ (MACH)_C_PICFLAGS)
867 C_PICFLAGS64 =       $($ (MACH64)_C_PICFLAGS)

869 sparc_C_BIGPICFLAGS = -K PIC
870 sparcv9_C_BIGPICFLAGS = -K PIC
871 i386_C_BIGPICFLAGS = -K PIC
872 amd64_C_BIGPICFLAGS = -K PIC
873 C_BIGPICFLAGS =     $($ (MACH)_C_BIGPICFLAGS)
874 C_BIGPICFLAGS64 =   $($ (MACH64)_C_BIGPICFLAGS)

876 # CC requires there to be no space between '-K' and 'pic' or 'PIC'.
877 sparc_CC_PICFLAGS =  -Kpic
878 sparcv9_CC_PICFLAGS = -KPIC
879 i386_CC_PICFLAGS =   -Kpic
880 amd64_CC_PICFLAGS =   -Kpic
881 CC_PICFLAGS =        $($ (MACH)_CC_PICFLAGS)
882 CC_PICFLAGS64 =     $($ (MACH64)_CC_PICFLAGS)

884 AS_PICFLAGS=         $(C_PICFLAGS)
885 AS_BIGPICFLAGS=     $(C_BIGPICFLAGS)

887 #
888 # Default label for CTF sections
889 #
890 CTFCVTFLAGS=         -i -L VERSION
891 $(SRCDBGBLD)CTFCVTFLAGS += -g

893 #
894 # Override to pass module-specific flags to ctfmerge. Currently used only by
895 # krtld to turn on fuzzy matching, and source-level debugging to inhibit
896 # stripping.
897 #
898 CTFMRGFLAGS=
899 $(SRCDBGBLD)CTFMRGFLAGS += -g

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

904 ELFSIGN_O=           $(TRUE)
905 ELFSIGN_CRYPTO=     $(ELFSIGN_O)
906 ELFSIGN_OBJECT=     $(ELFSIGN_O)

908 # Rules (normally from make.rules) and macros which are used for post
909 # processing files. Normally, these do stripping of the comment section
910 # automatically.
911 #   RELEASE_CM:      Should be edited to reflect the release.
912 #   POST_PROCESS_O:  Post-processing for '.o' files.
913 #   POST_PROCESS_A:  Post-processing for '.a' files (currently null).
914 #   POST_PROCESS_SO: Post-processing for '.so' files.
915 #   POST_PROCESS:    Post-processing for executable files (no suffix).
916 # Note that these macros are not completely generalized as they are to be
917 # used with the file name to be processed following.
918 #
919 # It is left as an exercise to Release Engineering to embellish the generation

```

```

920 # of the release comment string.
921 #
922 #   If this is a standard development build:
923 #       compress the comment section (mcs -c)
924 #       add the standard comment (mcs -a $(RELEASE_CM))
925 #       add the development specific comment (mcs -a $(DEV_CM))
926 #
927 #   If this is an installation build:
928 #       delete the comment section (mcs -d)
929 #       add the standard comment (mcs -a $(RELEASE_CM))
930 #       add the development specific comment (mcs -a $(DEV_CM))
931 #
932 #   If this is an release build:
933 #       delete the comment section (mcs -d)
934 #       add the standard comment (mcs -a $(RELEASE_CM))
935 #
936 # The following list of macros are used in the definition of RELEASE_CM
937 # which is used to label all binaries in the build:
938 #
939 #   RELEASE      Specific release of the build, eg: 5.2
940 #   RELEASE_MAJOR Major version number part of $(RELEASE)
941 #   RELEASE_MINOR Minor version number part of $(RELEASE)
942 #   VERSION      Version of the build (alpha, beta, Generic)
943 #   PATCHID      If this is a patch this value should contain
944 #                 the patchid value (eg: "Generic 100832-01"), otherwise
945 #                 it will be set to $(VERSION)
946 #   RELEASE_DATE Date of the Release Build
947 #   PATCH_DATE   Date the patch was created, if this is blank it
948 #                 will default to the RELEASE_DATE
949 #
950 RELEASE_MAJOR= 5
951 RELEASE_MINOR= 11
952 RELEASE=      $(RELEASE_MAJOR).$(RELEASE_MINOR)
953 VERSION=      SunOS Development
954 PATCHID=      $(VERSION)
955 RELEASE_DATE= release date not set
956 PATCH_DATE=   $(RELEASE_DATE)
957 RELEASE_CM=   "@$(POUND_SIGN)SunOS $(RELEASE) $(PATCHID) $(PATCH_DATE)"
958 DEV_CM=       "@$(POUND_SIGN)SunOS Internal Development: non-nightly build"

960 PROCESS_COMMENT= @?${MCS} -d -a $(RELEASE_CM) -a $(DEV_CM)
961 $(RELEASE_BUILD)PROCESS_COMMENT= @?${MCS} -d -a $(RELEASE_CM)

963 STRIP_STABS=   :
964 $(RELEASE_BUILD)STRIP_STABS= $(STRIP) -x $@
965 $(SRCDGBLD)STRIP_STABS=     :

967 POST_PROCESS_O=
967 POST_PROCESS_O=      $(PROCESS_COMMENT) $@
968 POST_PROCESS_A=
969 POST_PROCESS_SO=     $(PROCESS_COMMENT) $@ ; $(STRIP_STABS) ; \
970                        $(ELFSIGN_OBJECT)
971 POST_PROCESS=       $(PROCESS_COMMENT) $@ ; $(STRIP_STABS) ; \
972                        $(ELFSIGN_OBJECT)

974 #
975 # chk4ubin is a tool that inspects a module for a symbol table
976 # ELF section size which can trigger an OBP bug on older platforms.
977 # This problem affects only specific sun4u bootable modules.
978 #
979 CHK4UBIN=      $(ONBLD_TOOLS)/bin/$(MACH)/chk4ubin
980 CHK4UBINFLAGS=
981 CHK4UBINARY=   $(CHK4UBIN) $(CHK4UBINFLAGS) $@

983 #
984 # PKGARCHIVE specifies the default location where packages should be

```

```

985 # placed if built.
986 #
987 $(RELEASE_BUILD)PKGARCHIVESUFFIX= -nd
988 PKGARCHIVE=$(SRC)/../..../packages/$(MACH)/nightly$(PKGARCHIVESUFFIX)

990 #
991 # The repositories will be created with these publisher settings. To
992 # update an image to the resulting repositories, this must match the
993 # publisher name provided to "pkg set-publisher."
994 #
995 PKGPUBLISHER_REDIST= on-nightly
996 PKGPUBLISHER_NONREDIST= on-extra

998 #   Default build rules which perform comment section post-processing.
999 #
1000 .c:
1001     $(LINK.c) -o $@ $< $(LDLIBS)
1002     $(POST_PROCESS)
1003 .c.o:
1004     $(COMPILE.c) $(OUTPUT_OPTION) $< $(CTFCONVERT_HOOK)
1005     $(POST_PROCESS_O)
1006 .c.a:
1007     $(COMPILE.c) -o $% $<
1008     $(PROCESS_COMMENT) $%
1009     $(AR) $(ARFLAGS) $@ $%
1010     $(RM) $%
1011 .s.o:
1012     $(COMPILE.s) -o $@ $<
1013     $(POST_PROCESS_O)
1014 .s.a:
1015     $(COMPILE.s) -o $% $<
1016     $(PROCESS_COMMENT) $%
1017     $(AR) $(ARFLAGS) $@ $%
1018     $(RM) $%
1019 .cc:
1020     $(LINK.cc) -o $@ $< $(LDLIBS)
1021     $(POST_PROCESS)
1022 .cc.o:
1023     $(COMPILE.cc) $(OUTPUT_OPTION) $<
1024     $(POST_PROCESS_O)
1025 .cc.a:
1026     $(COMPILE.cc) -o $% $<
1027     $(AR) $(ARFLAGS) $@ $%
1028     $(PROCESS_COMMENT) $%
1029     $(RM) $%
1030 .y:
1031     $(YACC.y) $<
1032     $(LINK.c) -o $@ y.tab.c $(LDLIBS)
1033     $(POST_PROCESS)
1034     $(RM) y.tab.c
1035 .y.o:
1036     $(YACC.y) $<
1037     $(COMPILE.c) -o $@ y.tab.c $(CTFCONVERT_HOOK)
1038     $(POST_PROCESS_O)
1039     $(RM) y.tab.c
1040 .l:
1041     $(RM) $*.c
1042     $(LEX.l) $< > $*.c
1043     $(LINK.c) -o $@ $*.c -ll $(LDLIBS)
1044     $(POST_PROCESS)
1045     $(RM) $*.c
1046 .l.o:
1047     $(RM) $*.c
1048     $(LEX.l) $< > $*.c
1049     $(COMPILE.c) -o $@ $*.c $(CTFCONVERT_HOOK)
1050     $(POST_PROCESS_O)

```



```

1051      $(RM) $*.c

1053 .bin.o:
1054      $(COMPILE.b) -o $@ $<
1055      $(POST_PROCESS_O)

1057 .java.class:
1058      $(COMPILE.java) $<

1060 # Bourne and Korn shell script message catalog build rules.
1061 # We extract all gettext strings with sed(1) (being careful to permit
1062 # multiple gettext strings on the same line), weed out the dups, and
1063 # build the catalogue with awk(1).

1065 .sh.po .ksh.po:
1066      $(SED) -n -e ":a" \
1067              -e "h" \
1068              -e "s/.*gettext *\(\("[^"]*\)"\).*\/1/p" \
1069              -e "x" \
1070              -e "s/\(.*\)gettext *\("[^"]*\)"\(\.*\)\/1\2/" \
1071              -e "t a" \
1072      $< | sort -u | awk '{ print "msgid\t" $$0 "\nmsgstr" }' > $@

1074 #
1075 # Python and Perl executable and message catalog build rules.
1076 #
1077 .SUFFIXES: .pl .pm .py .pyc

1079 .pl:
1080      $(RM) $@;
1081      $(SED) -e "s@TEXT_DOMAIN@"$(TEXT_DOMAIN)"@" $< > $@;
1082      $(CHMOD) +x $@

1084 .py:
1085      $(RM) $@; $(CAT) $< > $@; $(CHMOD) +x $@

1087 .py.pyc:
1088      $(RM) $@
1089      $(PYTHON) -mpy_compile $<
1090      @[ $(<)c = $@ ] || $(MV) $(<)c $@

1092 .py.po:
1093      $(GNUXGETTEXT) $(GNUXGETFLAGS) -d $(<F:%.py=%) $< ;

1095 .pl.po .pm.po:
1096      $(XGETTEXT) $(XGETFLAGS) -d $(<F) $< ;
1097      $(RM) $@ ;
1098      $(SED) "/^domain/d" < $(<F).po > $@ ;
1099      $(RM) $(<F).po

1101 #
1102 # When using xgettext, we want messages to go to the default domain,
1103 # rather than the specified one. This special version of the
1104 # COMPILER.cpp macro effectively prevents expansion of TEXT_DOMAIN,
1105 # causing xgettext to put all messages into the default domain.
1106 #
1107 CPPFORPO=$(COMPILE.cpp:"$(TEXT_DOMAIN)"=TEXT_DOMAIN)

1109 .c.i:
1110      $(CPPFORPO) $< > $@

1112 .h.i:
1113      $(CPPFORPO) $< > $@

1115 .y.i:
1116      $(YACC) -d $<

```

```

1117      $(CPPFORPO) y.tab.c > $@
1118      $(RM) y.tab.c

1120 .l.i:
1121      $(LEX) $<
1122      $(CPPFORPO) lex.yy.c > $@
1123      $(RM) lex.yy.c

1125 .c.po:
1126      $(CPPFORPO) $< > $<.i
1127      $(BUILD.po)

1129 .y.po:
1130      $(YACC) -d $<
1131      $(CPPFORPO) y.tab.c > $<.i
1132      $(BUILD.po)
1133      $(RM) y.tab.c

1135 .l.po:
1136      $(LEX) $<
1137      $(CPPFORPO) lex.yy.c > $<.i
1138      $(BUILD.po)
1139      $(RM) lex.yy.c

1141 #
1142 # Rules to perform stylistic checks
1143 #
1144 .SUFFIXES: .x .xml .check .xmlchk

1146 .h.check:
1147      $(DOT_H_CHECK)

1149 .x.check:
1150      $(DOT_X_CHECK)

1152 .xml.xmlchk:
1153      $(MANIFEST_CHECK)

1155 #
1156 # Include rules to render automated sccs get rules "safe".
1157 #
1158 include $(SRC)/Makefile.noget

```