```
**********************************************************
   38145 Wed Feb 27 14:11:45 2019
new/usr/src/Makefile.master
10468 __ctype_mask[EOF] has been working by accident
10469 GCC's -faggressive-loop-optimizations is too aggressive
10470 array over-read in has_saved_fp()
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: John Levon <john.levon@joyent.com>
**********************************************************
```

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

#
# Copyright (c) 1989, 2010, Oracle and/or its affiliates. All rights reserved.
# Copyright (c) 2012 by Delphix. All rights reserved.
# Copyright 2014 Garrett D'Amore <garrett@damore.org>
# Copyright 2015, OmniTI Computer Consulting, Inc. All rights reserved.
# Copyright 2015 Gary Mills
# Copyright 2015 Igor Kozhukhov <ikozhukhov@gmail.com>
# Copyright 2016 Toomas Soome <tsoome@me.com>
# Copyright 2018 OmniOS Community Edition (OmniOSce) Association.
# Copyright (c) 2019, Joyent, Inc.
#

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

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

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

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



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

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

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

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

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

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

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

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

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

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

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

```
124 # Build uses java7 by default.  Pass one the variables below set to empty
125 # string in the environment to override.
126 BLD_JAVA_6=     $(POUND_SIGN)
127 BLD_JAVA_8=     $(POUND_SIGN)

129 GNUC_ROOT=      /opt/gcc/4.4.4
130 GCCLIBDIR=      $(GNUC_ROOT)/lib
131 GCCLIBDIR64=    $(GNUC_ROOT)/lib/$(MACH64)

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

135 RPCGEN=         /usr/bin/rpcgen
136 STABS=          $(ONBLD_TOOLS)/bin/$(MACH)/stabs
137 ELFEXTRACT=     $(ONBLD_TOOLS)/bin/$(MACH)/elfextract
138 MBH_PATCH=      $(ONBLD_TOOLS)/bin/$(MACH)/mbh_patch
139 BTXLD=          $(ONBLD_TOOLS)/bin/$(MACH)/btxld
140 VTFONTCVT=      $(ONBLD_TOOLS)/bin/$(MACH)/vtfontcvt
141 # echo(1) and true(1) are specified without absolute paths, so that the shell
142 # spawned by make(1) may use the built-in versions.  This is minimally
143 # problematic, as the shell spawned by make(1) is known and under control, the
144 # only risk being if the shell falls back to $PATH.
145 #
146 # We specifically want an echo(1) that does interpolation of escape sequences,
147 # which ksh93, /bin/sh, and bash will all provide.
148 ECHO=           echo
149 TRUE=           true
150 INS=            $(ONBLD_TOOLS)/bin/$(MACH)/install
151 SYMLINK=        /usr/bin/ln -s
152 LN=             /usr/bin/ln
153 MKDIR=          /usr/bin/mkdir
154 CHMOD=          /usr/bin/chmod
155 MV=             /usr/bin/mv -f
156 RM=             /usr/bin/rm -f
157 CUT=            /usr/bin/cut
158 NM=             /usr/ccs/bin/nm
159 DIFF=           /usr/bin/diff
160 GREP=           /usr/bin/grep
161 EGREP=          /usr/bin/egrep
162 ELFWRAP=        /usr/bin/elfwrap
163 KSH93=          /usr/bin/ksh93
164 SED=            /usr/bin/sed
165 AWK=            /usr/bin/nawk
166 CP=             /usr/bin/cp -f
167 MCS=            /usr/ccs/bin/mcs
168 CAT=            /usr/bin/cat
169 ELFDUMP=        /usr/ccs/bin/elfdump
170 M4=             /usr/bin/m4
171 GM4=            /usr/bin/gm4
172 STRIP=          /usr/ccs/bin/strip
173 LEX=            /usr/ccs/bin/lex
174 FLEX=           /usr/bin/flex
175 YACC=           /usr/ccs/bin/yacc
176 BISON=          /usr/bin/bison
177 CPP=            /usr/lib/cpp
178 ANSI_CPP=       $(GNUC_ROOT)/bin/cpp
179 JAVAC=          $(JAVA_ROOT)/bin/javac
180 JAVAH=          $(JAVA_ROOT)/bin/javah
181 JAVADOC=        $(JAVA_ROOT)/bin/javadoc
182 RMIC=           $(JAVA_ROOT)/bin/rmic
183 JAR=            $(JAVA_ROOT)/bin/jar
184 CTFCONVERT=     $(ONBLD_TOOLS)/bin/$(MACH)/ctfconvert
185 CTFDIFF=        $(ONBLD_TOOLS)/bin/$(MACH)/ctfdiff
186 CTFMERGE=       $(ONBLD_TOOLS)/bin/$(MACH)/ctfmerge
187 CTFSTABS=       $(ONBLD_TOOLS)/bin/$(MACH)/ctfstabs
188 CTFSTRIP=       $(ONBLD_TOOLS)/bin/$(MACH)/ctfstrip
189 NDRGEN=         $(ONBLD_TOOLS)/bin/$(MACH)/ndrgen
```



```
190 GENOFFSETS=     $(ONBLD_TOOLS)/bin/genoffsets
191 XREF=           $(ONBLD_TOOLS)/bin/xref
192 FIND=           /usr/bin/find
193 PERL=           /usr/bin/perl
194 PERL_VERSION=   5.10.0
195 PERL_PKGVERS=   -510
196 PERL_ARCH =             i86pc-solaris-64int
197 $(SPARC_BLD)PERL_ARCH = sun4-solaris-64int
198 PYTHON_VERSION= 2.7
199 PYTHON_PKGVERS= -27
200 PYTHON_SUFFIX=
201 PYTHON=          /usr/bin/python$(PYTHON_VERSION)
202 PYTHON3_VERSION=        3.5
203 PYTHON3_PKGVERS=        -35
204 PYTHON3_SUFFIX=         m
205 PYTHON3=               /usr/bin/python$(PYTHON3_VERSION)
206 $(BUILDPY3TOOLS)TOOLS_PYTHON=           $(PYTHON3)
207 $(BUILDPY2TOOLS)TOOLS_PYTHON=           $(PYTHON)
208 SORT=           /usr/bin/sort
209 TR=             /usr/bin/tr
210 TOUCH=          /usr/bin/touch
211 WC=             /usr/bin/wc
212 XARGS=          /usr/bin/xargs
213 ELFEDIT=        /usr/bin/elfedit
214 DTRACE=         /usr/sbin/dtrace -xnolibs
215 UNIQ=           /usr/bin/uniq
216 TAR=            /usr/bin/tar
217 ASTBINDIR=      /usr/ast/bin
218 MSGCC=          $(ASTBINDIR)/msgcc
219 MSGFMT=         /usr/bin/msgfmt -s
220 LCDEF=          $(ONBLD_TOOLS)/bin/$(MACH)/localedef
221 TIC=            $(ONBLD_TOOLS)/bin/$(MACH)/tic
222 ZIC=            $(ONBLD_TOOLS)/bin/$(MACH)/zic
223 OPENSSL=        /usr/bin/openssl
224 CPCGEN=         $(ONBLD_TOOLS)/bin/$(MACH)/cpcgen

226 FILEMODE=       644
227 DIRMODE=        755

229 # Declare that nothing should be built in parallel.
230 # Individual Makefiles can use the .PARALLEL target to declare otherwise.
231 .NO_PARALLEL:

233 # For stylistic checks
234 #
235 # Note that the X and C checks are not used at this time and may need
236 # modification when they are actually used.
237 #
238 CSTYLE=         $(ONBLD_TOOLS)/bin/cstyle
239 CSTYLE_TAIL=
240 HDRCHK=         $(ONBLD_TOOLS)/bin/hdrchk
241 HDRCHK_TAIL=
242 JSTYLE=         $(ONBLD_TOOLS)/bin/jstyle

244 DOT_H_CHECK=    \
245         @$(ECHO) "checking $<"; $(CSTYLE) $< $(CSTYLE_TAIL); \
246         $(HDRCHK) $< $(HDRCHK_TAIL)

248 DOT_X_CHECK=    \
249         @$(ECHO) "checking $<"; $(RPCGEN) -C -h $< | $(CSTYLE) $(CSTYLE_TAIL); \
250         $(RPCGEN) -C -h $< | $(HDRCHK) $< $(HDRCHK_TAIL)

252 DOT_C_CHECK=    \
253         @$(ECHO) "checking $<"; $(CSTYLE) $< $(CSTYLE_TAIL)

255 MANIFEST_CHECK= \
```

```
256         @$(ECHO) "checking $<"; \
257         SVCCFG_DTD=$(SRC)/cmd/svc/dtd/service_bundle.dtd.1 \
258         SVCCFG_REPOSITORY=$(SRC)/cmd/svc/seed/global.db \
259         SVCCFG_CONFIGD_PATH=$(SRC)/cmd/svc/configd/svc.configd-native \
260         $(SRC)/cmd/svc/svccfg/svccfg-native validate $<

262 INS.file=       $(RM) $@; $(INS) -s -m $(FILEMODE) -f $(@D) $<
263 INS.dir=        $(INS) -s -d -m $(DIRMODE) $@
264 # installs and renames at once
265 #
266 INS.rename=     $(INS.file); $(MV) $(@D)/$(<F) $@

268 # install a link
269 INSLINKTARGET=  $<
270 INS.link=       $(RM) $@; $(LN) $(INSLINKTARGET) $@
271 INS.symlink=    $(RM) $@; $(SYMLINK) $(INSLINKTARGET) $@

273 # The path to python that will be used for the shebang line when installing
274 # python scripts to the proto area. This is overridden by makefiles to
275 # select to the correct version.
276 PYSHEBANG=      $(PYTHON)

278 #
279 # Python bakes the mtime of the .py file into the compiled .pyc and
280 # rebuilds if the baked-in mtime != the mtime of the source file
281 # (rather than only if it's less than), thus when installing python
282 # files we must make certain to not adjust the mtime of the source
283 # (.py) file.
284 #
285 INS.pyfile=     $(RM) $@; $(SED) \
286                 -e "1s:^\#!@PYTHON@:\#!$(PYSHEBANG):" \
287                 -e "1s:^\#!@TOOLS_PYTHON@:\#!$(TOOLS_PYTHON):" \
288                 < $< > $@; $(CHMOD) $(FILEMODE) $@; $(TOUCH) -r $< $@

290 # MACH must be set in the shell environment per uname -p on the build host
291 # More specific architecture variables should be set in lower makefiles.
292 #
293 # MACH64 is derived from MACH, and BUILD64 is set to '#' for
294 # architectures on which we do not build 64-bit versions.
295 # (There are no such architectures at the moment.)
296 #
297 # Set BUILD64=# in the environment to disable 64-bit amd64
298 # builds on i386 machines.

300 MACH64_1=       $(MACH:sparc=sparcv9)
301 MACH64=         $(MACH64_1:i386=amd64)

303 MACH32_1=       $(MACH:sparc=sparcv7)
304 MACH32=         $(MACH32_1:i386=i86)

306 sparc_BUILD64=
307 i386_BUILD64=
308 BUILD64=        $($(MACH)_BUILD64)

310 #
311 # C compiler mode. Future compilers may change the default on us,
312 # so force extended ANSI mode globally. Lower level makefiles can
313 # override this by setting CCMODE.
314 #
315 CCMODE=                 -Xa
316 CCMODE64=               -Xa

318 #
319 # C compiler verbose mode. This is so we can enable it globally,
320 # but turn it off in the lower level makefiles of things we cannot
321 # (or aren't going to) fix.
```



```
322 #
323 CCVERBOSE=              -v

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

329 # set this to the secret flag "-Wc,-Qiselect-regsym=0" to disable register
330 # symbols (used to detect conflicts between objects that use global registers)
331 # we disable this now for safety, and because genunix doesn't link with
332 # this feature (the v9 default) enabled.
333 #
334 # REGSYM is separate since the C++ driver syntax is different.
335 CCREGSYM=               -Wc,-Qiselect-regsym=0
336 CCCREGSYM=              -Qoption cg -Qiselect-regsym=0

338 # Prevent the removal of static symbols by the SPARC code generator (cg).
339 # The x86 code generator (ube) does not remove such symbols and as such
340 # using this workaround is not applicable for x86.
341 #
342 CCSTATICSYM=            -Wc,-Qassembler-ounrefsym=0
343 #
344 # generate 32-bit addresses in the v9 kernel. Saves memory.
345 CCABS32=                -Wc,-xcode=abs32
346 #
347 # generate v9 code which tolerates callers using the v7 ABI, for the sake of
348 # system calls.
349 CC32BITCALLERS=         -_gcc=-massume-32bit-callers

351 # GCC, especially, is increasingly beginning to auto-inline functions and
352 # sadly does so separately not under the general -fno-inline-functions
353 # Additionally, we wish to prevent optimisations which cause GCC to clone
354 # functions -- in particular, these may cause unhelpful symbols to be
355 # emitted instead of function names
356 CCNOAUTOINLINE= \
357         -_gcc=-fno-inline-small-functions \
358         -_gcc=-fno-inline-functions-called-once \
359         -_gcc=-fno-ipa-cp \
360         -_gcc7=-fno-ipa-icf \
361         -_gcc8=-fno-ipa-icf \
362         -_gcc7=-fno-clone-functions \
363         -_gcc8=-fno-clone-functions

365 # GCC may put functions in different named sub-sections of .text based on
366 # their presumed calling frequency.  At least in the kernel, where we actually
367 # deliver relocatable objects, we don't want this to happen.
368 #
369 # Since at present we don't benefit from this even in userland, we disable it gl
370 # but the application of this may move into usr/src/uts/ in future.
371 CCNOREORDER=    \
372         -_gcc7=-fno-reorder-functions \
373         -_gcc8=-fno-reorder-functions

375 #
376 # gcc has a rather aggressive optimization on by default that infers loop
377 # bounds based on undefined behavior (!!).  This can lead to some VERY
378 # surprising optimizations -- ones that may be technically correct in the
379 # strictest sense but also result in incorrect program behavior.  We turn
380 # this optimization off, with extreme prejudice.
381 #
382 CCNOAGGRESSIVELOOPS= \
383         -_gcc7=-fno-aggressive-loop-optimizations \
384         -_gcc8=-fno-aggressive-loop-optimizations

386 # One optimization the compiler might perform is to turn this:
387 #       #pragma weak foo
```

```
388 #       extern int foo;
389 #       if (&foo)
390 #               foo = 5;
391 # into
392 #       foo = 5;
393 # Since we do some of this (foo might be referenced in common kernel code
394 # but provided only for some cpu modules or platforms), we disable this
395 # optimization.
396 #
397 sparc_CCUNBOUND = -Wd,-xsafe=unboundsym
398 i386_CCUNBOUND  =
399 CCUNBOUND       = $($(MACH)_CCUNBOUND)

401 #
402 # compiler '-xarch' flag. This is here to centralize it and make it
403 # overridable for testing.
404 sparc_XARCH=    -m32
405 sparcv9_XARCH=  -m64
406 i386_XARCH=     -m32
407 amd64_XARCH=    -m64 -Ui386 -U__i386

409 # assembler '-xarch' flag.  Different from compiler '-xarch' flag.
410 sparc_AS_XARCH=         -xarch=v8plus
411 sparcv9_AS_XARCH=       -xarch=v9
412 i386_AS_XARCH=
413 amd64_AS_XARCH=         -xarch=amd64 -P -Ui386 -U__i386

415 #
416 # These flags define what we need to be 'standalone' i.e. -not- part
417 # of the rather more cosy userland environment.  This basically means
418 # the kernel.
419 #
420 # XX64  future versions of gcc will make -mcmodel=kernel imply -mno-red-zone
421 #
422 sparc_STAND_FLAGS=      -_gcc=-ffreestanding
423 sparcv9_STAND_FLAGS=    -_gcc=-ffreestanding
424 # Disabling MMX also disables 3DNow, disabling SSE also disables all later
425 # additions to SSE (SSE2, AVX ,etc.)
426 NO_SIMD=                -_gcc=-mno-mmx -_gcc=-mno-sse
427 i386_STAND_FLAGS=       -_gcc=-ffreestanding $(NO_SIMD)
428 amd64_STAND_FLAGS=      -xmodel=kernel $(NO_SIMD)

430 SAVEARGS=               -Wu,-save_args
431 amd64_STAND_FLAGS       += $(SAVEARGS)

433 STAND_FLAGS_32 = $($(MACH)_STAND_FLAGS)
434 STAND_FLAGS_64 = $($(MACH64)_STAND_FLAGS)

436 #
437 # disable the incremental linker
438 ILDOFF=                 -xildoff
439 #
440 XFFLAG=                 -xF=%all
441 XESS=                   -xs
442 XSTRCONST=              -xstrconst

444 #
445 # turn warnings into errors (C)
446 CERRWARN = -errtags=yes -errwarn=%all
447 CERRWARN += -erroff=E_EMPTY_TRANSLATION_UNIT
448 CERRWARN += -erroff=E_STATEMENT_NOT_REACHED

450 CERRWARN += -_gcc=-Wno-missing-braces
451 CERRWARN += -_gcc=-Wno-sign-compare
452 CERRWARN += -_gcc=-Wno-unknown-pragmas
453 CERRWARN += -_gcc=-Wno-unused-parameter
```



```
454 CERRWARN += -_gcc=-Wno-missing-field-initializers

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

459 # DEBUG v. -nd make for frequent unused variables, empty conditions, etc. in
460 # -nd builds
461 $(RELEASE_BUILD)CERRWARN += -_gcc=-Wno-unused
462 $(RELEASE_BUILD)CERRWARN += -_gcc=-Wno-empty-body

464 CERRWARN += -_smatch=-p=illumos_user
465 include $(SRC)/Makefile.smatch

467 #
468 # turn warnings into errors (C++)
469 CCERRWARN=              -xwe

471 # C standard.  Keep Studio flags until we get rid of lint.
472 CSTD_GNU89=     -xc99=%none
473 CSTD_GNU99=     -xc99=%all
474 CSTD=           $(CSTD_GNU89)
475 C99LMODE=       $(CSTD:-xc99%=-Xc99%)

477 # In most places, assignments to these macros should be appended with +=
478 # (CPPFLAGS.first allows values to be prepended to CPPFLAGS).
479 sparc_CFLAGS=   $(sparc_XARCH) $(CCSTATICSYM)
480 sparcv9_CFLAGS= $(sparcv9_XARCH) -dalign $(CCVERBOSE) $(V9ABIWARN) $(CCREGSYM) \
481                 $(CCSTATICSYM)
482 i386_CFLAGS=    $(i386_XARCH)
483 amd64_CFLAGS=   $(amd64_XARCH)

485 sparc_ASFLAGS=  $(sparc_AS_XARCH)
486 sparcv9_ASFLAGS=$(sparcv9_AS_XARCH)
487 i386_ASFLAGS=   $(i386_AS_XARCH)
488 amd64_ASFLAGS=  $(amd64_AS_XARCH)

490 #
491 sparc_COPTFLAG=         -xO3
492 sparcv9_COPTFLAG=       -xO3
493 i386_COPTFLAG=          -O
494 amd64_COPTFLAG=         -xO3

496 COPTFLAG= $($(MACH)_COPTFLAG)
497 COPTFLAG64= $($(MACH64)_COPTFLAG)

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

503 # Direct the Sun Studio compiler to use a static globalization prefix based on t
504 # name of the module rather than something unique. Otherwise, objects
505 # will not build deterministically, as subsequent compilations of identical
506 # source will yeild objects that always look different.
507 #
508 # In the same spirit, this will also remove the date from the N_OPT stab.
509 CGLOBALSTATIC= -W0,-xglobalstatic

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

515 #
516 # We force the compilers to generate the debugging information best understood
517 # by the CTF tools.  With Sun Studio this is stabs due to bugs in the Studio
518 # compilers.  With GCC this is DWARF v2.
519 #
```

```
520 DEBUGFORMAT=    -_cc=-xdebugformat=stabs -_gcc=-gdwarf-2

522 #
523 # Ask the compiler to include debugging information
524 #
525 CCGDEBUG= -g $(DEBUGFORMAT)

527 #
528 # Flags used to build in debug mode for ctf generation.
529 #
530 CTF_FLAGS_sparc = $(CCGDEBUG) -Wc,-Qiselect-T1 $(CSTD) $(CNOGLOBAL)
531 CTF_FLAGS_i386  = $(CCGDEBUG) $(CSTD) $(CNOGLOBAL)

533 CTF_FLAGS_sparcv9       = $(CTF_FLAGS_sparc)
534 CTF_FLAGS_amd64         = $(CTF_FLAGS_i386)

536 # Sun Studio produces broken userland code when saving arguments.
537 $(__GNUC)CTF_FLAGS_amd64 += $(SAVEARGS)

539 CTF_FLAGS_32    = $(CTF_FLAGS_$(MACH))
540 CTF_FLAGS_64    = $(CTF_FLAGS_$(MACH64))
541 CTF_FLAGS       = $(CTF_FLAGS_32)

543 #
544 # Flags used with genoffsets
545 #
546 GENOFFSETS_FLAGS = $(CALLSYMS)

548 OFFSETS_CREATE = $(GENOFFSETS) -s $(CTFSTABS) -r $(CTFCONVERT) \
549         $(CW) --noecho $(CW_CC_COMPILERS) -- $(GENOFFSETS_FLAGS) \
550         $(CFLAGS) $(CPPFLAGS)

552 OFFSETS_CREATE64 = $(GENOFFSETS) -s $(CTFSTABS) -r $(CTFCONVERT) \
553         $(CW) --noecho $(CW_CC_COMPILERS) -- $(GENOFFSETS_FLAGS) \
554         $(CFLAGS64) $(CPPFLAGS)

556 #
557 # tradeoff time for space (smaller is better)
558 #
559 sparc_SPACEFLAG         = -xspace -W0,-Lt
560 sparcv9_SPACEFLAG       = -xspace -W0,-Lt
561 i386_SPACEFLAG          = -xspace
562 amd64_SPACEFLAG         =

564 SPACEFLAG               = $($(MACH)_SPACEFLAG)
565 SPACEFLAG64             = $($(MACH64)_SPACEFLAG)

567 #
568 # The Sun Studio 11 compiler has changed the behaviour of integer
569 # wrap arounds and so a flag is needed to use the legacy behaviour
570 # (without this flag panics/hangs could be exposed within the source).
571 #
572 sparc_IROPTFLAG         = -W2,-xwrap_int
573 sparcv9_IROPTFLAG       = -W2,-xwrap_int
574 i386_IROPTFLAG          =
575 amd64_IROPTFLAG         =

577 IROPTFLAG               = $($(MACH)_IROPTFLAG)
578 IROPTFLAG64             = $($(MACH64)_IROPTFLAG)

580 sparc_XREGSFLAG         = -xregs=no%appl
581 sparcv9_XREGSFLAG       = -xregs=no%appl
582 i386_XREGSFLAG          =
583 amd64_XREGSFLAG         =

585 XREGSFLAG               = $($(MACH)_XREGSFLAG)
```



```
586 XREGSFLAG64             = $($(MACH64)_XREGSFLAG)

588 # dmake SOURCEDEBUG=yes ... enables source-level debugging information, and
589 # avoids stripping it.
590 SOURCEDEBUG     = $(POUND_SIGN)
591 SRCDBGBLD       = $(SOURCEDEBUG:yes=)

593 #
594 # These variables are intended ONLY for use by developers to safely pass extra
595 # flags to the compilers without unintentionally overriding Makefile-set
596 # flags.  They should NEVER be set to any value in a Makefile.
597 #
598 # They come last in the associated FLAGS variable such that they can
599 # explicitly override things if necessary, there are gaps in this, but it's
600 # the best we can manage.
601 #
602 CUSERFLAGS              =
603 CUSERFLAGS64            = $(CUSERFLAGS)
604 CCUSERFLAGS             =
605 CCUSERFLAGS64           = $(CCUSERFLAGS)

607 CSOURCEDEBUGFLAGS       =
608 CCSOURCEDEBUGFLAGS      =
609 $(SRCDBGBLD)CSOURCEDEBUGFLAGS   = $(CCGDEBUG) -xs
610 $(SRCDBGBLD)CCSOURCEDEBUGFLAGS  = $(CCGDEBUG) -xs

612 CFLAGS=         $(COPTFLAG) $($(MACH)_CFLAGS) $(SPACEFLAG) $(CCMODE) \
613                 $(ILDOFF) $(CERRWARN) $(CSTD) $(CCUNBOUND) $(IROPTFLAG) \
614                 $(CGLOBALSTATIC) $(CCNOAUTOINLINE) $(CCNOREORDER) \
615                 $(CCNOAGGRESSIVELOOPS) \
616                 $(CSOURCEDEBUGFLAGS) $(CUSERFLAGS)
617 CFLAGS64=       $(COPTFLAG64) $($(MACH64)_CFLAGS) $(SPACEFLAG64) $(CCMODE64) \
618                 $(ILDOFF) $(CERRWARN) $(CSTD) $(CCUNBOUND) $(IROPTFLAG64) \
619                 $(CGLOBALSTATIC) $(CCNOAUTOINLINE) $(CCNOREORDER) \
620                 $(CCNOAGGRESSIVELOOPS) \
621                 $(CSOURCEDEBUGFLAGS) $(CUSERFLAGS64)
622 #
623 # Flags that are used to build parts of the code that are subsequently
624 # run on the build machine (also known as the NATIVE_BUILD).
625 #
626 NATIVE_CFLAGS=  $(COPTFLAG) $($(NATIVE_MACH)_CFLAGS) $(CCMODE) \
627                 $(ILDOFF) $(CERRWARN) $(CSTD) $($(NATIVE_MACH)_CCUNBOUND) \
628                 $(IROPTFLAG) $(CGLOBALSTATIC) $(CCNOAUTOINLINE) \
629                 $(CCNOREORDER) $(CSOURCEDEBUGFLAGS) $(CUSERFLAGS)

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

643 #
644 # For source message catalogue
645 #
646 .SUFFIXES: $(SUFFIXES) .i .po
647 MSGROOT= $(ROOT)/catalog
648 MSGDOMAIN= $(MSGROOT)/$(TEXT_DOMAIN)
649 MSGDOMAINPOFILE = $(MSGDOMAIN)/$(POFILE)
650 DCMSGDOMAIN= $(MSGROOT)/LC_TIME/$(TEXT_DOMAIN)
651 DCMSGDOMAINPOFILE = $(DCMSGDOMAIN)/$(DCFILE:.dc=.po)
```

```
653 CLOBBERFILES += $(POFILE) $(POFILES)
654 COMPILE.cpp= $(CC) -E -C $(CFLAGS) $(CPPFLAGS)
655 XGETTEXT= /usr/bin/xgettext
656 XGETFLAGS= -c TRANSLATION_NOTE
657 GNUXGETTEXT= /usr/gnu/bin/xgettext
658 GNUXGETFLAGS= --add-comments=TRANSLATION_NOTE --keyword=_ \
659         --strict --no-location --omit-header
660 BUILD.po= $(XGETTEXT) $(XGETFLAGS) -d $(<F) $<.i ;\
661         $(RM)   $@ ;\
662         $(SED) "/^domain/d" < $(<F).po > $@ ;\
663         $(RM) $(<F).po $<.i

665 #
666 # This is overwritten by local Makefile when PROG is a list.
667 #
668 POFILE= $(PROG).po

670 sparc_CCFLAGS=          -cg92 -compat=4 \
671                         -Qoption ccfe -messages=no%anachronism \
672                         $(CCERRWARN)
673 sparcv9_CCFLAGS=        $(sparcv9_XARCH) -dalign -compat=5 \
674                         -Qoption ccfe -messages=no%anachronism \
675                         -Qoption ccfe -features=no%conststrings \
676                         $(CCCREGSYM) \
677                         $(CCERRWARN)
678 i386_CCFLAGS=           -compat=4 \
679                         -Qoption ccfe -messages=no%anachronism \
680                         -Qoption ccfe -features=no%conststrings \
681                         $(CCERRWARN)
682 amd64_CCFLAGS=          $(amd64_XARCH) -compat=5 \
683                         -Qoption ccfe -messages=no%anachronism \
684                         -Qoption ccfe -features=no%conststrings \
685                         $(CCERRWARN)

687 sparc_CCOPTFLAG=        -O
688 sparcv9_CCOPTFLAG=      -O
689 i386_CCOPTFLAG=         -O
690 amd64_CCOPTFLAG=        -O

692 CCOPTFLAG=      $($(MACH)_CCOPTFLAG)
693 CCOPTFLAG64=    $($(MACH64)_CCOPTFLAG)
694 CCFLAGS=        $(CCOPTFLAG) $($(MACH)_CCFLAGS) $(CCSOURCEDEBUGFLAGS) \
695                 $(CCUSERFLAGS)
696 CCFLAGS64=      $(CCOPTFLAG64) $($(MACH64)_CCFLAGS) $(CCSOURCEDEBUGFLAGS) \
697                 $(CCUSERFLAGS64)

699 #
700 #
701 #
702 ELFWRAP_FLAGS   =
703 ELFWRAP_FLAGS64 =       -64

705 #
706 # Various mapfiles that are used throughout the build, and delivered to
707 # /usr/lib/ld.
708 #
709 MAPFILE.NED_i386 =      $(SRC)/common/mapfiles/common/map.noexdata
710 MAPFILE.NED_sparc =
711 MAPFILE.NED =           $(MAPFILE.NED_$(MACH))
712 MAPFILE.PGA =           $(SRC)/common/mapfiles/common/map.pagealign
713 MAPFILE.NES =           $(SRC)/common/mapfiles/common/map.noexstk
714 MAPFILE.FLT =           $(SRC)/common/mapfiles/common/map.filter
715 MAPFILE.LEX =           $(SRC)/common/mapfiles/common/map.lex.yy

717 #
```



```
718 # Generated mapfiles that are compiler specific, and used throughout the
719 # build.  These mapfiles are not delivered in /usr/lib/ld.
720 #
721 MAPFILE.NGB_sparc=      $(SRC)/common/mapfiles/gen/sparc_cc_map.noexeglobs
722 $(__GNUC64)MAPFILE.NGB_sparc= \
723                         $(SRC)/common/mapfiles/gen/sparc_gcc_map.noexeglobs
724 MAPFILE.NGB_sparcv9=    $(SRC)/common/mapfiles/gen/sparcv9_cc_map.noexeglobs
725 $(__GNUC64)MAPFILE.NGB_sparcv9= \
726                         $(SRC)/common/mapfiles/gen/sparcv9_gcc_map.noexeglobs
727 MAPFILE.NGB_i386=       $(SRC)/common/mapfiles/gen/i386_cc_map.noexeglobs
728 $(__GNUC64)MAPFILE.NGB_i386= \
729                         $(SRC)/common/mapfiles/gen/i386_gcc_map.noexeglobs
730 MAPFILE.NGB_amd64=      $(SRC)/common/mapfiles/gen/amd64_cc_map.noexeglobs
731 $(__GNUC64)MAPFILE.NGB_amd64= \
732                         $(SRC)/common/mapfiles/gen/amd64_gcc_map.noexeglobs
733 MAPFILE.NGB =           $(MAPFILE.NGB_$(MACH))

735 #
736 # A generic interface mapfile name, used by various dynamic objects to define
737 # the interfaces and interposers the object must export.
738 #
739 MAPFILE.INT =           mapfile-intf

741 #
742 # LDLIBS32 and LDLIBS64 can be set in the environment to override the following
743 # assignments.
744 #
745 # These environment settings make sure that no libraries are searched outside
746 # of the local workspace proto area:
747 #       LDLIBS32=-YP,$ROOT/lib:$ROOT/usr/lib
748 #       LDLIBS64=-YP,$ROOT/lib/$MACH64:$ROOT/usr/lib/$MACH64
749 #
750 LDLIBS32 =      $(ENVLDLIBS1) $(ENVLDLIBS2) $(ENVLDLIBS3)
751 LDLIBS32 +=     $(ADJUNCT_PROTO:%=-L%/usr/lib -L%/lib)
752 LDLIBS.cmd =    $(LDLIBS32)
753 LDLIBS.lib =    $(LDLIBS32)

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

760 #
761 # Define compilation macros.
762 #
763 COMPILE.c=      $(CC) $(CFLAGS) $(CPPFLAGS) -c
764 COMPILE64.c=    $(CC) $(CFLAGS64) $(CPPFLAGS) -c
765 COMPILE.cc=     $(CCC) $(CCFLAGS) $(CPPFLAGS) -c
766 COMPILE64.cc=   $(CCC) $(CCFLAGS64) $(CPPFLAGS) -c
767 COMPILE.s=      $(AS) $(ASFLAGS) $(AS_CPPFLAGS)
768 COMPILE64.s=    $(AS) $(ASFLAGS) $($(MACH64)_AS_XARCH) $(AS_CPPFLAGS)
769 COMPILE.d=      $(DTRACE) -G -32
770 COMPILE64.d=    $(DTRACE) -G -64
771 COMPILE.b=      $(ELFWRAP) $(ELFWRAP_FLAGS$(CLASS))
772 COMPILE64.b=    $(ELFWRAP) $(ELFWRAP_FLAGS$(CLASS))

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

777 #
778 # Link time macros
779 #
780 CCNEEDED                = -lC
781 CCEXTNEEDED             = -lCrun -lCstd
782 $(__GNUC)CCNEEDED       = -L$(GCCLIBDIR) -lstdc++ -lgcc_s
783 $(__GNUC)CCEXTNEEDED    = $(CCNEEDED)
```

```
785 LINK.c=         $(CC) $(CFLAGS) $(CPPFLAGS) $(LDFLAGS)
786 LINK64.c=       $(CC) $(CFLAGS64) $(CPPFLAGS) $(LDFLAGS)
787 NORUNPATH=      -norunpath -nolib
788 LINK.cc=        $(CCC) $(CCFLAGS) $(CPPFLAGS) $(NORUNPATH) \
789                 $(LDFLAGS) $(CCNEEDED)
790 LINK64.cc=      $(CCC) $(CCFLAGS64) $(CPPFLAGS) $(NORUNPATH) \
791                 $(LDFLAGS) $(CCNEEDED)

793 #
794 # lint macros
795 #
796 # Note that the undefine of __PRAGMA_REDEFINE_EXTNAME can be removed once
797 # ON is built with a version of lint that has the fix for 4484186.
798 #
799 ALWAYS_LINT_DEFS =      -errtags=yes -s
800 ALWAYS_LINT_DEFS +=     -erroff=E_PTRDIFF_OVERFLOW
801 ALWAYS_LINT_DEFS +=     -erroff=E_ASSIGN_NARROW_CONV
802 ALWAYS_LINT_DEFS +=     -U__PRAGMA_REDEFINE_EXTNAME
803 ALWAYS_LINT_DEFS +=     $(C99LMODE)
804 ALWAYS_LINT_DEFS +=     -errsecurity=$(SECLEVEL)
805 ALWAYS_LINT_DEFS +=     -erroff=E_SEC_CREAT_WITHOUT_EXCL
806 ALWAYS_LINT_DEFS +=     -erroff=E_SEC_FORBIDDEN_WARN_CREAT
807 # XX64 -- really only needed for amd64 lint
808 ALWAYS_LINT_DEFS +=     -erroff=E_ASSIGN_INT_TO_SMALL_INT
809 ALWAYS_LINT_DEFS +=     -erroff=E_CAST_INT_CONST_TO_SMALL_INT
810 ALWAYS_LINT_DEFS +=     -erroff=E_CAST_INT_TO_SMALL_INT
811 ALWAYS_LINT_DEFS +=     -erroff=E_CAST_TO_PTR_FROM_INT
812 ALWAYS_LINT_DEFS +=     -erroff=E_COMP_INT_WITH_LARGE_INT
813 ALWAYS_LINT_DEFS +=     -erroff=E_INTEGRAL_CONST_EXP_EXPECTED
814 ALWAYS_LINT_DEFS +=     -erroff=E_PASS_INT_TO_SMALL_INT
815 ALWAYS_LINT_DEFS +=     -erroff=E_PTR_CONV_LOSES_BITS

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

821 SECLEVEL=       core
822 LINT.c=         $(LINT) $(ONLY_LINT_DEFS) $(LINTFLAGS) $(CPPFLAGS) \
823                 $(ALWAYS_LINT_DEFS)
824 LINT64.c=       $(LINT) $(ONLY_LINT_DEFS) $(LINTFLAGS64) $(CPPFLAGS) \
825                 $(ALWAYS_LINT_DEFS)
826 LINT.s=         $(LINT.c)

828 # For some future builds, NATIVE_MACH and MACH might be different.
829 # Therefore, NATIVE_MACH needs to be redefined in the
830 # environment as ‘uname -p‘ to override this macro.
831 #
832 # For now at least, we cross-compile amd64 on i386 machines.
833 NATIVE_MACH=    $(MACH:amd64=i386)

835 # Define native compilation macros
836 #

838 # Base directory where compilers are loaded.
839 # Defined here so it can be overridden by developer.
840 #
841 SPRO_ROOT=              $(BUILD_TOOLS)/SUNWspro
842 SPRO_VROOT=             $(SPRO_ROOT)/SS12
843 GNU_ROOT=               /usr

845 $(__GNUC)PRIMARY_CC=    gcc4,$(GNUC_ROOT)/bin/gcc,gnu
846 $(__SUNC)PRIMARY_CC=    studio12,$(SPRO_VROOT)/bin/cc,sun
847 $(__GNUC)PRIMARY_CCC=   gcc4,$(GNUC_ROOT)/bin/g++,gnu
848 $(__SUNC)PRIMARY_CCC=   studio12,$(SPRO_VROOT)/bin/CC,sun
```



```
850 CW_CC_COMPILERS=        $(PRIMARY_CC:%=--primary %) $(SHADOW_CCS:%=--shadow %)
851 CW_CCC_COMPILERS=       $(PRIMARY_CCC:%=--primary %) $(SHADOW_CCCS:%=--shadow %)


854 # Till SS12u1 formally becomes the NV CBE, LINT is hard
855 # coded to be picked up from the $SPRO_ROOT/sunstudio12.1/
856 # location. Impacted variables are sparc_LINT, sparcv9_LINT,
857 # i386_LINT, amd64_LINT.
858 # Reset them when SS12u1 is rolled out.
859 #

861 # Specify platform compiler versions for languages
862 # that we use (currently only c and c++).
863 #
864 CW=                     $(ONBLD_TOOLS)/bin/$(MACH)/cw

866 BUILD_CC=               $(CW) $(CW_CC_COMPILERS) --
867 BUILD_CCC=              $(CW) -C $(CW_CCC_COMPILERS) --
868 BUILD_CPP=              /usr/ccs/lib/cpp
869 BUILD_LD=               /usr/ccs/bin/ld
870 BUILD_LINT=             $(SPRO_ROOT)/sunstudio12.1/bin/lint

872 $(MACH)_CC=             $(BUILD_CC)
873 $(MACH)_CCC=            $(BUILD_CCC)
874 $(MACH)_CPP=            $(BUILD_CPP)
875 $(MACH)_LD=             $(BUILD_LD)
876 $(MACH)_LINT=           $(BUILD_LINT)
877 $(MACH64)_CC=           $(BUILD_CC)
878 $(MACH64)_CCC=          $(BUILD_CCC)
879 $(MACH64)_CPP=          $(BUILD_CPP)
880 $(MACH64)_LD=           $(BUILD_LD)
881 $(MACH64)_LINT=         $(BUILD_LINT)

883 sparc_AS=               /usr/ccs/bin/as -xregsym=no
884 sparcv9_AS=             $($(MACH)_AS)

886 i386_AS=                /usr/ccs/bin/as
887 $(__GNUC)i386_AS=       $(ONBLD_TOOLS)/bin/$(MACH)/aw
888 amd64_AS=               $(ONBLD_TOOLS)/bin/$(MACH)/aw

890 NATIVECC=              $($(NATIVE_MACH)_CC)
891 NATIVECCC=             $($(NATIVE_MACH)_CCC)
892 NATIVECPP=             $($(NATIVE_MACH)_CPP)
893 NATIVEAS=              $($(NATIVE_MACH)_AS)
894 NATIVELD=              $($(NATIVE_MACH)_LD)
895 NATIVELINT=            $($(NATIVE_MACH)_LINT)

897 #
898 # Makefile.master.64 overrides these settings
899 #
900 CC=                     $(NATIVECC)
901 CCC=                    $(NATIVECCC)
902 CPP=                    $(NATIVECPP)
903 AS=                     $(NATIVEAS)
904 LD=                     $(NATIVELD)
905 LINT=                   $(NATIVELINT)

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

910 BDIRECT=        -Bdirect
911 BDYNAMIC=       -Bdynamic
912 BLOCAL=         -Blocal
913 BNODIRECT=      -Bnodirect
914 BREDUCE=        -Breduce
915 BSTATIC=        -Bstatic
```

```
917 ZDEFS=          -zdefs
918 ZDIRECT=        -zdirect
919 ZIGNORE=        -zignore
920 ZINITFIRST=     -zinitfirst
921 ZINTERPOSE=     -zinterpose
922 ZLAZYLOAD=      -zlazyload
923 ZLOADFLTR=      -zloadfltr
924 ZMULDEFS=       -zmuldefs
925 ZNODEFAULTLIB=  -znodefaultlib
926 ZNODEFS=        -znodefs
927 ZNODELETE=      -znodelete
928 ZNODLOPEN=      -znodlopen
929 ZNODUMP=        -znodump
930 ZNOLAZYLOAD=    -znolazyload
931 ZNOLDYNSYM=     -znoldynsym
932 ZNORELOC=       -znoreloc
933 ZNOVERSION=     -znoversion
934 ZRECORD=        -zrecord
935 ZREDLOCSYM=     -zredlocsym
936 ZTEXT=          -ztext
937 ZVERBOSE=       -zverbose

939 GSHARED=        -G
940 CCMT=           -mt

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

945 sparc_C_PICFLAGS =      -fpic
946 sparcv9_C_PICFLAGS =    -fpic
947 i386_C_PICFLAGS =       -fpic
948 amd64_C_PICFLAGS =      -fpic
949 C_PICFLAGS =            $($(MACH)_C_PICFLAGS)
950 C_PICFLAGS64 =          $($(MACH64)_C_PICFLAGS)

952 sparc_C_BIGPICFLAGS =   -fPIC
953 sparcv9_C_BIGPICFLAGS = -fPIC
954 i386_C_BIGPICFLAGS =    -fPIC
955 amd64_C_BIGPICFLAGS =   -fPIC
956 C_BIGPICFLAGS =         $($(MACH)_C_BIGPICFLAGS)
957 C_BIGPICFLAGS64 =       $($(MACH64)_C_BIGPICFLAGS)

959 # CC requires there to be no space between '-K' and 'pic' or 'PIC'.
960 # and does not support -f
961 sparc_CC_PICFLAGS =     -_cc=-Kpic -_gcc=-fpic
962 sparcv9_CC_PICFLAGS =   -_cc=-KPIC -_gcc=-fPIC
963 i386_CC_PICFLAGS =      -_cc=-Kpic -_gcc=-fpic
964 amd64_CC_PICFLAGS =     -_cc=-Kpic -_gcc=-fpic
965 CC_PICFLAGS =           $($(MACH)_CC_PICFLAGS)
966 CC_PICFLAGS64 =         $($(MACH64)_CC_PICFLAGS)

968 AS_PICFLAGS=            -K pic
969 AS_BIGPICFLAGS=         -K PIC

971 #
972 # Default label for CTF sections
973 #
974 CTFCVTFLAGS=            -i -L VERSION

976 #
977 # Override to pass module-specific flags to ctfmerge.  Currently used only by
978 # krtld to turn on fuzzy matching, and source-level debugging to inhibit
979 # stripping.
980 #
981 CTFMRGFLAGS=
```



```
983 #
984 # Make the transition between old and new CTF Tools. The new ctf tools
985 # do not support stabs (eg. Sun Studio). By setting BUILD_OLD_CTF_TOOLS
986 # here or in the environment file, the old ones will be built.
987 #
988 BUILD_NEW_CTF_TOOLS=
989 BUILD_OLD_CTF_TOOLS=$(POUND_SIGN)
990 $(BUILD_OLD_CTF_TOOLS)BUILD_NEW_CTF_TOOLS= $(POUND_SIGN)

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

994 # Rules (normally from make.rules) and macros which are used for post
995 # processing files. Normally, these do stripping of the comment section
996 # automatically.
997 #    RELEASE_CM:        Should be editted to reflect the release.
998 #    POST_PROCESS_O:    Post-processing for '.o' files.
999 #    POST_PROCESS_A:    Post-processing for '.a' files (currently null).
1000 #    POST_PROCESS_SO:   Post-processing for '.so' files.
1001 #    POST_PROCESS:      Post-processing for executable files (no suffix).
1002 # Note that these macros are not completely generalized as they are to be
1003 # used with the file name to be processed following.
1004 #
1005 # It is left as an exercise to Release Engineering to embellish the generation
1006 # of the release comment string.
1007 #
1008 #       If this is a standard development build:
1009 #               compress the comment section (mcs -c)
1010 #               add the standard comment (mcs -a $(RELEASE_CM))
1011 #               add the development specific comment (mcs -a $(DEV_CM))
1012 #
1013 #       If this is an installation build:
1014 #               delete the comment section (mcs -d)
1015 #               add the standard comment (mcs -a $(RELEASE_CM))
1016 #               add the development specific comment (mcs -a $(DEV_CM))
1017 #
1018 #       If this is an release build:
1019 #               delete the comment section (mcs -d)
1020 #               add the standard comment (mcs -a $(RELEASE_CM))
1021 #
1022 # The following list of macros are used in the definition of RELEASE_CM
1023 # which is used to label all binaries in the build:
1024 #
1025 #       RELEASE         Specific release of the build, eg: 5.2
1026 #       RELEASE_MAJOR   Major version number part of $(RELEASE)
1027 #       RELEASE_MINOR   Minor version number part of $(RELEASE)
1028 #       VERSION         Version of the build (alpha, beta, Generic)
1029 #       PATCHID         If this is a patch this value should contain
1030 #                       the patchid value (eg: "Generic 100832-01"), otherwise
1031 #                       it will be set to $(VERSION)
1032 #       RELEASE_DATE    Date of the Release Build
1033 #       PATCH_DATE      Date the patch was created, if this is blank it
1034 #                       will default to the RELEASE_DATE
1035 #
1036 RELEASE_MAJOR=  5
1037 RELEASE_MINOR=  11
1038 RELEASE=        $(RELEASE_MAJOR).$(RELEASE_MINOR)
1039 VERSION=        SunOS Development
1040 PATCHID=        $(VERSION)
1041 RELEASE_DATE=   release date not set
1042 PATCH_DATE=     $(RELEASE_DATE)
1043 RELEASE_CM=     "@($(POUND_SIGN))SunOS $(RELEASE) $(PATCHID) $(PATCH_DATE)"
1044 DEV_CM=         "@($(POUND_SIGN))SunOS Internal Development: non-nightly build"

1046 PROCESS_COMMENT=  @?${MCS} -d -a $(RELEASE_CM) -a $(DEV_CM)
1047 $(RELEASE_BUILD)PROCESS_COMMENT=   @?${MCS} -d -a $(RELEASE_CM)
```

```
1049 STRIP_STABS=                       $(STRIP) -x $@
1050 $(SRCDBGBLD)STRIP_STABS=           :

1052 POST_PROCESS_O=
1053 POST_PROCESS_A=
1054 POST_PROCESS_SO=        $(PROCESS_COMMENT) $@ ; $(STRIP_STABS) ; \
1055                         $(ELFSIGN_OBJECT)
1056 POST_PROCESS=           $(PROCESS_COMMENT) $@ ; $(STRIP_STABS) ; \
1057                         $(ELFSIGN_OBJECT)

1059 #
1060 # chk4ubin is a tool that inspects a module for a symbol table
1061 # ELF section size which can trigger an OBP bug on older platforms.
1062 # This problem affects only specific sun4u bootable modules.
1063 #
1064 CHK4UBIN=       $(ONBLD_TOOLS)/bin/$(MACH)/chk4ubin
1065 CHK4UBINFLAGS=
1066 CHK4UBINARY=    $(CHK4UBIN) $(CHK4UBINFLAGS) $@

1068 #
1069 # PKGARCHIVE specifies the default location where packages should be
1070 # placed if built.
1071 #
1072 $(RELEASE_BUILD)PKGARCHIVESUFFIX=       -nd
1073 PKGARCHIVE=$(SRC)/../../packages/$(MACH)/nightly$(PKGARCHIVESUFFIX)

1075 #
1076 # The repositories will be created with these publisher settings.  To
1077 # update an image to the resulting repositories, this must match the
1078 # publisher name provided to "pkg set-publisher."
1079 #
1080 PKGPUBLISHER_REDIST=    on-nightly
1081 PKGPUBLISHER_NONREDIST= on-extra

1083 #       Default build rules which perform comment section post-processing.
1084 #
1085 .c:
1086         $(LINK.c) -o $@ $< $(LDLIBS)
1087         $(POST_PROCESS)
1088 .c.o:
1089         $(COMPILE.c) $(OUTPUT_OPTION) $< $(CTFCONVERT_HOOK)
1090         $(POST_PROCESS_O)
1091 .c.a:
1092         $(COMPILE.c) -o $% $<
1093         $(PROCESS_COMMENT) $%
1094         $(AR) $(ARFLAGS) $@ $%
1095         $(RM) $%
1096 .s.o:
1097         $(COMPILE.s) -o $@ $<
1098         $(POST_PROCESS_O)
1099 .s.a:
1100         $(COMPILE.s) -o $% $<
1101         $(PROCESS_COMMENT) $%
1102         $(AR) $(ARFLAGS) $@ $%
1103         $(RM) $%
1104 .cc:
1105         $(LINK.cc) -o $@ $< $(LDLIBS)
1106         $(POST_PROCESS)
1107 .cc.o:
1108         $(COMPILE.cc) $(OUTPUT_OPTION) $<
1109         $(POST_PROCESS_O)
1110 .cc.a:
1111         $(COMPILE.cc) -o $% $<
1112         $(AR) $(ARFLAGS) $@ $%
1113         $(PROCESS_COMMENT) $%
```



```
1114         $(RM) $%
1115 .y:
1116         $(YACC.y) $<
1117         $(LINK.c) -o $@ y.tab.c $(LDLIBS)
1118         $(POST_PROCESS)
1119         $(RM) y.tab.c
1120 .y.o:
1121         $(YACC.y) $<
1122         $(COMPILE.c) -o $@ y.tab.c $(CTFCONVERT_HOOK)
1123         $(POST_PROCESS_O)
1124         $(RM) y.tab.c
1125 .l:
1126         $(RM) $*.c
1127         $(LEX.l) $< > $*.c
1128         $(LINK.c) -o $@ $*.c -ll $(LDLIBS)
1129         $(POST_PROCESS)
1130         $(RM) $*.c
1131 .l.o:
1132         $(RM) $*.c
1133         $(LEX.l) $< > $*.c
1134         $(COMPILE.c) -o $@ $*.c $(CTFCONVERT_HOOK)
1135         $(POST_PROCESS_O)
1136         $(RM) $*.c

1138 .bin.o:
1139         $(COMPILE.b) -o $@ $<
1140         $(POST_PROCESS_O)

1142 .java.class:
1143         $(COMPILE.java) $<

1145 # Bourne and Korn shell script message catalog build rules.
1146 # We extract all gettext strings with sed(1) (being careful to permit
1147 # multiple gettext strings on the same line), weed out the dups, and
1148 # build the catalogue with awk(1).

1150 .sh.po .ksh.po:
1151         $(SED) -n -e ":a"                                       \
1152                   -e "h"                                        \
1153                   -e "s/.*gettext *\(\"[^\"]*\"\).*/\1/p"       \
1154                   -e "x"                                        \
1155                   -e "s/\(.*\)gettext *\"[^\"]*\"\(.*\)/\1\2/"  \
1156                   -e "t a"                                      \
1157                 $< | sort -u | $(AWK) '{ print "msgid\t" $$0 "\nmsgstr" }' > $@

1159 #
1160 # Python and Perl executable and message catalog build rules.
1161 #
1162 .SUFFIXES: .pl .pm .py .pyc

1164 .pl:
1165         $(RM) $@;
1166         $(SED) -e "s@TEXT_DOMAIN@\"$(TEXT_DOMAIN)\"@" $< > $@;
1167         $(CHMOD) +x $@

1169 .py:
1170         $(RM) $@; $(SED) \
1171                 -e "1s:^\#!@PYTHON@:\#!$(PYSHEBANG):" \
1172                 -e "1s:^\#!@TOOLS_PYTHON@:\#!$(TOOLS_PYTHON):" \
1173                 < $< > $@; $(CHMOD) +x $@

1175 .py.pyc:
1176         $(RM) $@
1177         $(PYTHON) -mpy_compile $<
1178         @[ $(<)c = $@ ] || $(MV) $(<)c $@
```

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

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

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

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

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

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

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

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

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

.y.po:
        $(YACC) -d $<
        $(CPPFORPO) y.tab.c  > $<.i
        $(BUILD.po)
        $(RM) y.tab.c

.l.po:
        $(LEX) $<
        $(CPPFORPO) lex.yy.c  > $<.i
        $(BUILD.po)
        $(RM) lex.yy.c

#
# Rules to perform stylistic checks
#
.SUFFIXES: .x .xml .check .xmlchk

.h.check:
        $(DOT_H_CHECK)

.x.check:
        $(DOT_X_CHECK)

.xml.xmlchk:
        $(MANIFEST_CHECK)
```



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

****
10567 Wed Feb 27 14:11:45 2019
new/usr/src/lib/libc/port/locale/table.c
10468 __ctype_mask[EOF] has been working by accident
10469 GCC's -faggressive-loop-optimizations is too aggressive
10470 array over-read in has_saved_fp()
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: John Levon <john.levon@joyent.com>
****

```
/*
 * Copyright 2013 Garrett D'Amore <garrett@damore.org>
 * Copyright 2017 Nexenta Systems, Inc.
 * Copyright 2019 Joyent, Inc.
 * Copyright (c) 1993
 *      The Regents of the University of California.  All rights reserved.
 *
 * This code is derived from software contributed to Berkeley by
 * Paul Borman at Krystal Technologies.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 */

#include "lint.h"
#include <ctype.h>
#include <wchar.h>
#include "runetype.h"
#include "mblocal.h"
#include "_ctype.h"

#define _DEFRUNETYPE \
#define _DEFRUNETYPE { \
        /* 00 */ \
        _CTYPE_C, \
        _CTYPE_C, \
        _CTYPE_C, \
        _CTYPE_C, \
        _CTYPE_C, \
        _CTYPE_C, \
        _CTYPE_C, \
        _CTYPE_C, \
        /* 08 */ \
        _CTYPE_C, \
        _CTYPE_C|_CTYPE_S|_CTYPE_B, \
        _CTYPE_C|_CTYPE_S, \
```



```
        _CTYPE_C|_CTYPE_S, \
        _CTYPE_C|_CTYPE_S, \
        _CTYPE_C|_CTYPE_S, \
        _CTYPE_C, \
        _CTYPE_C, \
        /* 10 */ \
        _CTYPE_C, \
        _CTYPE_C, \
        _CTYPE_C, \
        _CTYPE_C, \
        _CTYPE_C, \
        _CTYPE_C, \
        _CTYPE_C, \
        _CTYPE_C, \
        /* 18 */ \
        _CTYPE_C, \
        _CTYPE_C, \
        _CTYPE_C, \
        _CTYPE_C, \
        _CTYPE_C, \
        _CTYPE_C, \
        _CTYPE_C, \
        _CTYPE_C, \
        /* 20 */ \
        _CTYPE_S|_CTYPE_B|_CTYPE_R, \
        _CTYPE_P|_CTYPE_R|_CTYPE_G, \
        _CTYPE_P|_CTYPE_R|_CTYPE_G, \
        _CTYPE_P|_CTYPE_R|_CTYPE_G, \
        _CTYPE_P|_CTYPE_R|_CTYPE_G, \
        _CTYPE_P|_CTYPE_R|_CTYPE_G, \
        _CTYPE_P|_CTYPE_R|_CTYPE_G, \
        _CTYPE_P|_CTYPE_R|_CTYPE_G, \
        /* 28 */ \
        _CTYPE_P|_CTYPE_R|_CTYPE_G, \
        _CTYPE_P|_CTYPE_R|_CTYPE_G, \
        _CTYPE_P|_CTYPE_R|_CTYPE_G, \
        _CTYPE_P|_CTYPE_R|_CTYPE_G, \
        _CTYPE_P|_CTYPE_R|_CTYPE_G, \
        _CTYPE_P|_CTYPE_R|_CTYPE_G, \
        _CTYPE_P|_CTYPE_R|_CTYPE_G, \
        _CTYPE_P|_CTYPE_R|_CTYPE_G, \
        /* 30 */ \
        _CTYPE_D|_CTYPE_R|_CTYPE_G|_CTYPE_X, \
        _CTYPE_D|_CTYPE_R|_CTYPE_G|_CTYPE_X, \
        _CTYPE_D|_CTYPE_R|_CTYPE_G|_CTYPE_X, \
        _CTYPE_D|_CTYPE_R|_CTYPE_G|_CTYPE_X, \
        _CTYPE_D|_CTYPE_R|_CTYPE_G|_CTYPE_X, \
        _CTYPE_D|_CTYPE_R|_CTYPE_G|_CTYPE_X, \
        _CTYPE_D|_CTYPE_R|_CTYPE_G|_CTYPE_X, \
        _CTYPE_D|_CTYPE_R|_CTYPE_G|_CTYPE_X, \
        /* 38 */ \
        _CTYPE_D|_CTYPE_R|_CTYPE_G|_CTYPE_X, \
        _CTYPE_D|_CTYPE_R|_CTYPE_G|_CTYPE_X, \
        _CTYPE_P|_CTYPE_R|_CTYPE_G, \
        _CTYPE_P|_CTYPE_R|_CTYPE_G, \
        _CTYPE_P|_CTYPE_R|_CTYPE_G, \
        _CTYPE_P|_CTYPE_R|_CTYPE_G, \
        _CTYPE_P|_CTYPE_R|_CTYPE_G, \
        _CTYPE_P|_CTYPE_R|_CTYPE_G, \
        /* 40 */ \
        _CTYPE_P|_CTYPE_R|_CTYPE_G, \
        _CTYPE_U|_CTYPE_X|_CTYPE_R|_CTYPE_G|_CTYPE_A, \
        _CTYPE_U|_CTYPE_X|_CTYPE_R|_CTYPE_G|_CTYPE_A, \
        _CTYPE_U|_CTYPE_X|_CTYPE_R|_CTYPE_G|_CTYPE_A, \
        _CTYPE_U|_CTYPE_X|_CTYPE_R|_CTYPE_G|_CTYPE_A, \
        _CTYPE_U|_CTYPE_X|_CTYPE_R|_CTYPE_G|_CTYPE_A, \
```

```
123         _CTYPE_U|_CTYPE_X|_CTYPE_R|_CTYPE_G|_CTYPE_A, \
124         _CTYPE_U|_CTYPE_R|_CTYPE_G|_CTYPE_A, \
125         /* 48 */ \
126         _CTYPE_U|_CTYPE_R|_CTYPE_G|_CTYPE_A, \
127         _CTYPE_U|_CTYPE_R|_CTYPE_G|_CTYPE_A, \
128         _CTYPE_U|_CTYPE_R|_CTYPE_G|_CTYPE_A, \
129         _CTYPE_U|_CTYPE_R|_CTYPE_G|_CTYPE_A, \
130         _CTYPE_U|_CTYPE_R|_CTYPE_G|_CTYPE_A, \
131         _CTYPE_U|_CTYPE_R|_CTYPE_G|_CTYPE_A, \
132         _CTYPE_U|_CTYPE_R|_CTYPE_G|_CTYPE_A, \
133         _CTYPE_U|_CTYPE_R|_CTYPE_G|_CTYPE_A, \
134         /* 50 */ \
135         _CTYPE_U|_CTYPE_R|_CTYPE_G|_CTYPE_A, \
136         _CTYPE_U|_CTYPE_R|_CTYPE_G|_CTYPE_A, \
137         _CTYPE_U|_CTYPE_R|_CTYPE_G|_CTYPE_A, \
138         _CTYPE_U|_CTYPE_R|_CTYPE_G|_CTYPE_A, \
139         _CTYPE_U|_CTYPE_R|_CTYPE_G|_CTYPE_A, \
140         _CTYPE_U|_CTYPE_R|_CTYPE_G|_CTYPE_A, \
141         _CTYPE_U|_CTYPE_R|_CTYPE_G|_CTYPE_A, \
142         _CTYPE_U|_CTYPE_R|_CTYPE_G|_CTYPE_A, \
143         /* 58 */ \
144         _CTYPE_U|_CTYPE_R|_CTYPE_G|_CTYPE_A, \
145         _CTYPE_U|_CTYPE_R|_CTYPE_G|_CTYPE_A, \
146         _CTYPE_U|_CTYPE_R|_CTYPE_G|_CTYPE_A, \
147         _CTYPE_P|_CTYPE_R|_CTYPE_G, \
148         _CTYPE_P|_CTYPE_R|_CTYPE_G, \
149         _CTYPE_P|_CTYPE_R|_CTYPE_G, \
150         _CTYPE_P|_CTYPE_R|_CTYPE_G, \
151         _CTYPE_P|_CTYPE_R|_CTYPE_G, \
152         /* 60 */ \
153         _CTYPE_P|_CTYPE_R|_CTYPE_G, \
154         _CTYPE_L|_CTYPE_X|_CTYPE_R|_CTYPE_G|_CTYPE_A, \
155         _CTYPE_L|_CTYPE_X|_CTYPE_R|_CTYPE_G|_CTYPE_A, \
156         _CTYPE_L|_CTYPE_X|_CTYPE_R|_CTYPE_G|_CTYPE_A, \
157         _CTYPE_L|_CTYPE_X|_CTYPE_R|_CTYPE_G|_CTYPE_A, \
158         _CTYPE_L|_CTYPE_X|_CTYPE_R|_CTYPE_G|_CTYPE_A, \
159         _CTYPE_L|_CTYPE_X|_CTYPE_R|_CTYPE_G|_CTYPE_A, \
160         _CTYPE_L|_CTYPE_R|_CTYPE_G|_CTYPE_A, \
161         /* 68 */ \
162         _CTYPE_L|_CTYPE_R|_CTYPE_G|_CTYPE_A, \
163         _CTYPE_L|_CTYPE_R|_CTYPE_G|_CTYPE_A, \
164         _CTYPE_L|_CTYPE_R|_CTYPE_G|_CTYPE_A, \
165         _CTYPE_L|_CTYPE_R|_CTYPE_G|_CTYPE_A, \
166         _CTYPE_L|_CTYPE_R|_CTYPE_G|_CTYPE_A, \
167         _CTYPE_L|_CTYPE_R|_CTYPE_G|_CTYPE_A, \
168         _CTYPE_L|_CTYPE_R|_CTYPE_G|_CTYPE_A, \
169         _CTYPE_L|_CTYPE_R|_CTYPE_G|_CTYPE_A, \
170         /* 70 */ \
171         _CTYPE_L|_CTYPE_R|_CTYPE_G|_CTYPE_A, \
172         _CTYPE_L|_CTYPE_R|_CTYPE_G|_CTYPE_A, \
173         _CTYPE_L|_CTYPE_R|_CTYPE_G|_CTYPE_A, \
174         _CTYPE_L|_CTYPE_R|_CTYPE_G|_CTYPE_A, \
175         _CTYPE_L|_CTYPE_R|_CTYPE_G|_CTYPE_A, \
176         _CTYPE_L|_CTYPE_R|_CTYPE_G|_CTYPE_A, \
177         _CTYPE_L|_CTYPE_R|_CTYPE_G|_CTYPE_A, \
178         _CTYPE_L|_CTYPE_R|_CTYPE_G|_CTYPE_A, \
179         /* 78 */ \
180         _CTYPE_L|_CTYPE_R|_CTYPE_G|_CTYPE_A, \
181         _CTYPE_L|_CTYPE_R|_CTYPE_G|_CTYPE_A, \
182         _CTYPE_L|_CTYPE_R|_CTYPE_G|_CTYPE_A, \
183         _CTYPE_P|_CTYPE_R|_CTYPE_G, \
184         _CTYPE_P|_CTYPE_R|_CTYPE_G, \
185         _CTYPE_P|_CTYPE_R|_CTYPE_G, \
186         _CTYPE_P|_CTYPE_R|_CTYPE_G, \
187         _CTYPE_C
186         _CTYPE_C, \
```



```
187 }

189 #define _DEFMAPLOWER \
189 #define _DEFMAPLOWER { \
190         0x00,   0x01,   0x02,   0x03,   0x04,   0x05,   0x06,   0x07, \
191         0x08,   0x09,   0x0a,   0x0b,   0x0c,   0x0d,   0x0e,   0x0f, \
192         0x10,   0x11,   0x12,   0x13,   0x14,   0x15,   0x16,   0x17, \
193         0x18,   0x19,   0x1a,   0x1b,   0x1c,   0x1d,   0x1e,   0x1f, \
194         0x20,   0x21,   0x22,   0x23,   0x24,   0x25,   0x26,   0x27, \
195         0x28,   0x29,   0x2a,   0x2b,   0x2c,   0x2d,   0x2e,   0x2f, \
196         0x30,   0x31,   0x32,   0x33,   0x34,   0x35,   0x36,   0x37, \
197         0x38,   0x39,   0x3a,   0x3b,   0x3c,   0x3d,   0x3e,   0x3f, \
198         0x40,   'a',    'b',    'c',    'd',    'e',    'f',    'g', \
199         'h',    'i',    'j',    'k',    'l',    'm',    'n',    'o', \
200         'p',    'q',    'r',    's',    't',    'u',    'v',    'w', \
201         'x',    'y',    'z',    0x5b,   0x5c,   0x5d,   0x5e,   0x5f, \
202         0x60,   'a',    'b',    'c',    'd',    'e',    'f',    'g', \
203         'h',    'i',    'j',    'k',    'l',    'm',    'n',    'o', \
204         'p',    'q',    'r',    's',    't',    'u',    'v',    'w', \
205         'x',    'y',    'z',    0x7b,   0x7c,   0x7d,   0x7e,   0x7f, \
206         0x80,   0x81,   0x82,   0x83,   0x84,   0x85,   0x86,   0x87, \
207         0x88,   0x89,   0x8a,   0x8b,   0x8c,   0x8d,   0x8e,   0x8f, \
208         0x90,   0x91,   0x92,   0x93,   0x94,   0x95,   0x96,   0x97, \
209         0x98,   0x99,   0x9a,   0x9b,   0x9c,   0x9d,   0x9e,   0x9f, \
210         0xa0,   0xa1,   0xa2,   0xa3,   0xa4,   0xa5,   0xa6,   0xa7, \
211         0xa8,   0xa9,   0xaa,   0xab,   0xac,   0xad,   0xae,   0xaf, \
212         0xb0,   0xb1,   0xb2,   0xb3,   0xb4,   0xb5,   0xb6,   0xb7, \
213         0xb8,   0xb9,   0xba,   0xbb,   0xbc,   0xbd,   0xbe,   0xbf, \
214         0xc0,   0xc1,   0xc2,   0xc3,   0xc4,   0xc5,   0xc6,   0xc7, \
215         0xc8,   0xc9,   0xca,   0xcb,   0xcc,   0xcd,   0xce,   0xcf, \
216         0xd0,   0xd1,   0xd2,   0xd3,   0xd4,   0xd5,   0xd6,   0xd7, \
217         0xd8,   0xd9,   0xda,   0xdb,   0xdc,   0xdd,   0xde,   0xdf, \
218         0xe0,   0xe1,   0xe2,   0xe3,   0xe4,   0xe5,   0xe6,   0xe7, \
219         0xe8,   0xe9,   0xea,   0xeb,   0xec,   0xed,   0xee,   0xef, \
220         0xf0,   0xf1,   0xf2,   0xf3,   0xf4,   0xf5,   0xf6,   0xf7, \
221         0xf8,   0xf9,   0xfa,   0xfb,   0xfc,   0xfd,   0xfe,   0xff
221         0xf8,   0xf9,   0xfa,   0xfb,   0xfc,   0xfd,   0xfe,   0xff, \
222 }

223 #define _DEFMAPUPPER \
224 #define _DEFMAPUPPER { \
224         0x00,   0x01,   0x02,   0x03,   0x04,   0x05,   0x06,   0x07, \
225         0x08,   0x09,   0x0a,   0x0b,   0x0c,   0x0d,   0x0e,   0x0f, \
226         0x10,   0x11,   0x12,   0x13,   0x14,   0x15,   0x16,   0x17, \
227         0x18,   0x19,   0x1a,   0x1b,   0x1c,   0x1d,   0x1e,   0x1f, \
228         0x20,   0x21,   0x22,   0x23,   0x24,   0x25,   0x26,   0x27, \
229         0x28,   0x29,   0x2a,   0x2b,   0x2c,   0x2d,   0x2e,   0x2f, \
230         0x30,   0x31,   0x32,   0x33,   0x34,   0x35,   0x36,   0x37, \
231         0x38,   0x39,   0x3a,   0x3b,   0x3c,   0x3d,   0x3e,   0x3f, \
232         0x40,   'A',    'B',    'C',    'D',    'E',    'F',    'G', \
233         'H',    'I',    'J',    'K',    'L',    'M',    'N',    'O', \
234         'P',    'Q',    'R',    'S',    'T',    'U',    'V',    'W', \
235         'X',    'Y',    'Z',    0x5b,   0x5c,   0x5d,   0x5e,   0x5f, \
236         0x60,   'A',    'B',    'C',    'D',    'E',    'F',    'G', \
237         'H',    'I',    'J',    'K',    'L',    'M',    'N',    'O', \
238         'P',    'Q',    'R',    'S',    'T',    'U',    'V',    'W', \
239         'X',    'Y',    'Z',    0x7b,   0x7c,   0x7d,   0x7e,   0x7f, \
240         0x80,   0x81,   0x82,   0x83,   0x84,   0x85,   0x86,   0x87, \
241         0x88,   0x89,   0x8a,   0x8b,   0x8c,   0x8d,   0x8e,   0x8f, \
242         0x90,   0x91,   0x92,   0x93,   0x94,   0x95,   0x96,   0x97, \
243         0x98,   0x99,   0x9a,   0x9b,   0x9c,   0x9d,   0x9e,   0x9f, \
244         0xa0,   0xa1,   0xa2,   0xa3,   0xa4,   0xa5,   0xa6,   0xa7, \
245         0xa8,   0xa9,   0xaa,   0xab,   0xac,   0xad,   0xae,   0xaf, \
246         0xb0,   0xb1,   0xb2,   0xb3,   0xb4,   0xb5,   0xb6,   0xb7, \
247         0xb8,   0xb9,   0xba,   0xbb,   0xbc,   0xbd,   0xbe,   0xbf, \
248         0xc0,   0xc1,   0xc2,   0xc3,   0xc4,   0xc5,   0xc6,   0xc7, \
```

```
249         0xc8,   0xc9,   0xca,   0xcb,   0xcc,   0xcd,   0xce,   0xcf, \
250         0xd0,   0xd1,   0xd2,   0xd3,   0xd4,   0xd5,   0xd6,   0xd7, \
251         0xd8,   0xd9,   0xda,   0xdb,   0xdc,   0xdd,   0xde,   0xdf, \
252         0xe0,   0xe1,   0xe2,   0xe3,   0xe4,   0xe5,   0xe6,   0xe7, \
253         0xe8,   0xe9,   0xea,   0xeb,   0xec,   0xed,   0xee,   0xef, \
254         0xf0,   0xf1,   0xf2,   0xf3,   0xf4,   0xf5,   0xf6,   0xf7, \
255         0xf8,   0xf9,   0xfa,   0xfb,   0xfc,   0xfd,   0xfe,   0xff,
256         0xf8,   0xf9,   0xfa,   0xfb,   0xfc,   0xfd,   0xfe,   0xff, \
257 }

257 _RuneLocale _DefaultRuneLocale = {
258         _RUNE_MAGIC_1,
259         "NONE",
260         { _DEFRUNETYPE },
261         { _DEFMAPLOWER },
262         { _DEFMAPUPPER },
262         _DEFRUNETYPE,
263         _DEFMAPLOWER,
264         _DEFMAPUPPER,
263 };

265 /*
266  * __ctype_mask, __trans_lower, and __trans_upper come from former _ctype.c and
267  * have to stay pointers for binary compatibility, so we provide separate
268  * storage for them, initialized to "C" locale contents by default.  Note that
269  * legacy code may dereference __ctype_mask[-1] when checking against EOF,
270  * relying on that value to be 0.  To allow this, ___ctype_mask is expanded by
271  * one value and prepended with a leading 0, with __ctype_mask being set to
272  * point to ___ctype_mask[1].  (__trans_lower and __trans_upper do not suffer
273  * from this as EOF access was prevented in legacy code by a check against
274  * isascii(), which always returned 0 for EOF.)
270  * storage for them, initialized to "C" locale contents by default.
275  */
276 static unsigned int ___ctype_mask[_CACHED_RUNES + 1] = { 0, _DEFRUNETYPE };
277 unsigned int *__ctype_mask = &___ctype_mask[1];
272 static unsigned int ___ctype_mask[_CACHED_RUNES] = _DEFRUNETYPE;
273 unsigned int *__ctype_mask = ___ctype_mask;

279 static int ___trans_lower[_CACHED_RUNES] = { _DEFMAPLOWER };
275 static int ___trans_lower[_CACHED_RUNES] = _DEFMAPLOWER;
280 int *__trans_lower = ___trans_lower;

282 static int ___trans_upper[_CACHED_RUNES] = { _DEFMAPUPPER };
278 static int ___trans_upper[_CACHED_RUNES] = _DEFMAPUPPER;
283 int *__trans_upper = ___trans_upper;
```

```
**********************************************************
    3098 Wed Feb 27 14:11:45 2019
new/usr/src/lib/libsaveargs/Makefile.com
10468 __ctype_mask[EOF] has been working by accident
10469 GCC's -faggressive-loop-optimizations is too aggressive
10470 array over-read in has_saved_fp()
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: John Levon <john.levon@joyent.com>
**********************************************************
#
# 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 2007 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#
# Copyright (c) 2019, Joyent, Inc.
# Copyright (c) 2018, Joyent, Inc.

#
# The build process for libsaveargs is sightly different from that used by other
# libraries, because libsaveargs must be built in two flavors - as a standalone
# for use by kmdb and as a normal library.  We use $(CURTYPE) to indicate the
# current flavor being built.
#

LIBRARY=        libsaveargs.a
STANDLIBRARY=   libstandsaveargs.so
VERS=           .1

# By default, we build the shared library.  Construction of the standalone
# is specifically requested by architecture-specific Makefiles.
TYPES=          library
CURTYPE=        library

COMDIR=         $(SRC)/lib/libsaveargs/common

OBJECTS_common_amd64    = saveargs.o
SRCS_common_amd64       = $(OBJECTS_common_amd64:%.o=../amd64/%.c)

OBJECTS= $(OBJECTS_common_$(MACH)) $(OBJECTS_common_$(MACH64)) $(OBJECTS_common_

include $(SRC)/lib/Makefile.lib

SRCS=   $(SRCS_common_$(MACH)) $(SRCS_common_$(MACH64)) $(SRC_common_common)

#
# Used to verify that the standalone doesn't have any unexpected external
# dependencies.
```



```
#
LINKTEST_OBJ = objs/linktest_stand.o

CLOBBERFILES_standalone = $(LINKTEST_OBJ)
CLOBBERFILES += $(CLOBBERFILES_$(CURTYPE))

LIBS_standalone = $(STANDLIBRARY)
LIBS_library = $(DYNLIB) $(LINTLIB)
LIBS = $(LIBS_$(CURTYPE))

MAPFILES =      $(COMDIR)/mapfile-vers

LDLIBS +=       -lc -ldisasm

LDFLAGS_standalone = $(ZNOVERSION) $(BREDUCE) -dy -r
LDFLAGS = $(LDFLAGS_$(CURTYPE))

ASFLAGS_standalone = -DDIS_STANDALONE
ASFLAGS_library =
ASFLAGS += -P $(ASFLAGS_$(CURTYPE)) -D_ASM

$(LINTLIB) := SRCS = $(COMDIR)/$(LINTSRC)

# We want the thread-specific errno in the library, but we don't want it in
# the standalone.  $(DTS_ERRNO) is designed to add -D_TS_ERRNO to $(CPPFLAGS),
# in order to enable this feature.  Conveniently, -D_REENTRANT does the same
# thing.  As such, we null out $(DTS_ERRNO) to ensure that the standalone
# doesn't get it.
DTS_ERRNO=

CPPFLAGS_standalone = -DDIS_STANDALONE
CPPFLAGS_library = -D_REENTRANT
CPPFLAGS +=     -I$(COMDIR) $(CPPFLAGS_$(CURTYPE))

CFLAGS_standalone = $(STAND_FLAGS_32)
CFLAGS_common =
CFLAGS += $(CFLAGS_$(CURTYPE)) $(CFLAGS_common)

CFLAGS64_standalone = $(STAND_FLAGS_64)
CFLAGS64 += $(CCVERBOSE) $(CFLAGS64_$(CURTYPE)) $(CFLAGS64_common)

# not linted
SMATCH=off

DYNFLAGS +=     $(ZINTERPOSE)

.KEEP_STATE:
```

```
**********************************************************
    8842 Wed Feb 27 14:11:45 2019
new/usr/src/lib/libsaveargs/amd64/saveargs.c
10468 __ctype_mask[EOF] has been working by accident
10469 GCC's -faggressive-loop-optimizations is too aggressive
10470 array over-read in has_saved_fp()
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: John Levon <john.levon@joyent.com>
**********************************************************
```

```
/*
 * 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 2007 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

/*
 * Copyright 2019 Joyent, Inc.
 */

/*
 * The Sun Studio and GCC (patched for opensolaris/illumos) compilers
 * implement a argument saving scheme on amd64 via the -Wu,save-args or
 * options.  When the option is specified, INTEGER type function arguments
 * passed via registers will be saved on the stack immediately after %rbp, and
 * will not be modified through out the life of the routine.
 *
 *                              +--------+
 *              %rbp    -->     |  %rbp  |
 *                              +--------+
 *              -0x8(%rbp)      |  %rdi  |
 *                              +--------+
 *              -0x10(%rbp)     |  %rsi  |
 *                              +--------+
 *              -0x18(%rbp)     |  %rdx  |
 *                              +--------+
 *              -0x20(%rbp)     |  %rcx  |
 *                              +--------+
 *              -0x28(%rbp)     |  %r8   |
 *                              +--------+
 *              -0x30(%rbp)     |  %r9   |
 *                              +--------+
 *
 *
 * For example, for the following function,
 *
 * void
 * foo(int a1, int a2, int a3, int a4, int a5, int a6, int a7)
```



```
 * {
 * ...
 * }
 *
 * Disassembled code will look something like the following:
 *
 *      pushq    %rbp
 *      movq     %rsp, %rbp
 *      subq     $imm8, %rsp                        **
 *      movq     %rdi, -0x8(%rbp)
 *      movq     %rsi, -0x10(%rbp)
 *      movq     %rdx, -0x18(%rbp)
 *      movq     %rcx, -0x20(%rbp)
 *      movq     %r8, -0x28(%rbp)
 *      movq     %r9, -0x30(%rbp)
 *      ...
 * or
 *      pushq    %rbp
 *      movq     %rsp, %rbp
 *      subq     $imm8, %rsp                        **
 *      movq     %r9, -0x30(%rbp)
 *      movq     %r8, -0x28(%rbp)
 *      movq     %rcx, -0x20(%rbp)
 *      movq     %rdx, -0x18(%rbp)
 *      movq     %rsi, -0x10(%rbp)
 *      movq     %rdi, -0x8(%rbp)
 *      ...
 * or
 *      pushq    %rbp
 *      movq     %rsp, %rbp
 *      pushq    %rdi
 *      pushq    %rsi
 *      pushq    %rdx
 *      pushq    %rcx
 *      pushq    %r8
 *      pushq    %r9
 *
 * **: The space being reserved is in addition to what the current
 *     function prolog already reserves.
 *
 * We loop through the first SAVEARGS_INSN_SEQ_LEN bytes of the function
 * looking for each argument saving instruction we would expect to see.
 *
 * If there are odd number of arguments to a function, additional space is
 * reserved on the stack to maintain 16-byte alignment.  For example,
 *
 *      argc == 0: no argument saving.
 *      argc == 3: save 3, but space for 4 is reserved
 *      argc == 7: save 6.
 */

#include <sys/sysmacros.h>
#include <sys/types.h>
#include <libdisasm.h>
#include <string.h>

#include <saveargs.h>

/*
 * Size of the instruction sequence arrays.  It should correspond to
 * the maximum number of arguments passed via registers.
 */
#define INSTR_ARRAY_SIZE        6

#define INSTR1(ins, off) (ins[(off)])
#define INSTR2(ins, off) (ins[(off)] + (ins[(off) + 1] << 8))
```

```
#define INSTR3(ins, off)        \
        (ins[(off)] + (ins[(off) + 1] << 8) + (ins[(off + 2)] << 16))
#define INSTR4(ins, off)        \
        (ins[(off)] + (ins[(off) + 1] << 8) + (ins[(off + 2)] << 16) + \
        (ins[(off) + 3] << 24))

/*
 * Sun Studio 10 patch implementation saves %rdi first;
 * GCC 3.4.3 Sun branch implementation saves them in reverse order.
 */
static const uint32_t save_instr[INSTR_ARRAY_SIZE] = {
        0xf87d8948,     /* movq %rdi, -0x8(%rbp) */
        0xf0758948,     /* movq %rsi, -0x10(%rbp) */
        0xe8558948,     /* movq %rdx, -0x18(%rbp) */
        0xe04d8948,     /* movq %rcx, -0x20(%rbp) */
        0xd845894c,     /* movq %r8, -0x28(%rbp) */
        0xd04d894c      /* movq %r9, -0x30(%rbp) */
};
```

*unchanged_portion_omitted_*

```
static boolean_t
has_saved_fp(dis_handle_t *dhp, uint8_t *ins, int size)
{
        int             i, j;
        uint32_t        n;
        boolean_t       found_push = B_FALSE;
        ssize_t         sz = 0;

        for (i = 0; i < size; i += sz) {
                if ((sz = instr_size(dhp, ins, i, size)) < 1)
                        return (B_FALSE);

                if (found_push == B_FALSE) {
                        if (sz != 1)
                                continue;

                        n = INSTR1(ins, i);
                        for (j = 0; j < NUM_FP_PUSHES; j++)
                        for (j = 0; j <= NUM_FP_PUSHES; j++)
                                if (save_fp_pushes[j] == n) {
                                        found_push = B_TRUE;
                                        break;
                                }
                } else {
                        if (sz != 3)
                                continue;
                        n = INSTR3(ins, i);
                        for (j = 0; j < NUM_FP_MOVS; j++)
                        for (j = 0; j <= NUM_FP_MOVS; j++)
                                if (save_fp_movs[j] == n)
                                        return (B_TRUE);
                }
        }

        return (B_FALSE);
}
```

*unchanged_portion_omitted_*

```
**********************************************************
   21739 Wed Feb 27 14:11:45 2019
new/usr/src/uts/Makefile.uts
10468 __ctype_mask[EOF] has been working by accident
10469 GCC's -faggressive-loop-optimizations is too aggressive
10470 array over-read in has_saved_fp()
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: John Levon <john.levon@joyent.com>
**********************************************************
#
# 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) 1991, 2010, Oracle and/or its affiliates. All rights reserved.
# Copyright (c) 2011 Bayard G. Bell. All rights reserved.
# Copyright (c) 2011 by Delphix. All rights reserved.
# Copyright (c) 2013 Andrew Stormont.  All rights reserved.
# Copyright 2016 Hans Rosenfeld <rosenfeld@grumpf.hope-2000.org>
# Copyright (c) 2019, Joyent, Inc.
#

#
#       This Makefile contains the common targets and definitions for
#       all kernels. It is to be included in the Makefiles for specific
#       implementation architectures and processor architecture dependent
#       modules: i.e.: all driving kernel Makefiles.
#
#       Include global definitions:
#
include $(SRC)/Makefile.master

#
#       No text domain in the kernel.
#
DTEXTDOM =

#
#       Keep references to $(SRC)/common relative.
COMMONBASE=     $(UTSBASE)/../common

#
#       Setup build-specific vars
#       To add a build type:
#               add name to ALL_BUILDS32 & ALL_BUILDS64
#               set CLASS_name and OBJ_DIR_name
#               add targets to Makefile.targ
#
```



```
#
#       DEF_BUILDS is for def, lint, sischeck, and install
#       ALL_BUILDS is for everything else (all, clean, ...)
#
# The NOT_RELEASE_BUILD noise is to maintain compatibility with the
# gatekeeper's nightly build script.
#
DEF_BUILDS32                            = obj32
DEF_BUILDS64                            = obj64
DEF_BUILDSONLY64                        = obj64
$(NOT_RELEASE_BUILD)DEF_BUILDS32        = debug32
$(NOT_RELEASE_BUILD)DEF_BUILDS64        = debug64
$(NOT_RELEASE_BUILD)DEF_BUILDSONLY64    = debug64
ALL_BUILDS32                            = obj32 debug32
ALL_BUILDS64                            = obj64 debug64
ALL_BUILDSONLY64                        = obj64 debug64

#
#       For modules in 64b dirs that aren't built 64b
#       or modules in 64b dirs that aren't built 32b we
#       need to create empty modlintlib files so global lint works
#
LINT32_BUILDS   = debug32
LINT64_BUILDS   = debug64

#
#       Build class (32b or 64b)
#
CLASS_OBJ32     = 32
CLASS_DBG32     = 32
CLASS_OBJ64     = 64
CLASS_DBG64     = 64
CLASS           = $(CLASS_$(BUILD_TYPE))

#
#       Build subdirectory
#
OBJS_DIR_OBJ32  = obj32
OBJS_DIR_DBG32  = debug32
OBJS_DIR_OBJ64  = obj64
OBJS_DIR_DBG64  = debug64
OBJS_DIR        = $(OBJS_DIR_$(BUILD_TYPE))

#
#       Create defaults so empty rules don't
#       confuse make
#
CLASS_          = 64
OBJS_DIR_       = debug64

#
#       Build tools
#
CC_sparc_32     = $(sparc_CC)
CC_sparc_64     = $(sparcv9_CC)

CC_i386_32      = $(i386_CC)
CC_i386_64      = $(amd64_CC)
CC_amd64_64     = $(amd64_CC)

CC              = $(CC_$(MACH)_$(CLASS))

AS_sparc_32     = $(sparc_AS)
AS_sparc_64     = $(sparcv9_AS)

AS_i386_32      = $(i386_AS)
```

```
124 AS_i386_64      = $(amd64_AS)
125 AS_amd64_64     = $(amd64_AS)

127 AS              = $(AS_$(MACH)_$(CLASS))

129 LD_sparc_32     = $(sparc_LD)
130 LD_sparc_64     = $(sparcv9_LD)

132 LD_i386_32      = $(i386_LD)
133 LD_i386_64      = $(amd64_LD)
134 LD_amd64_64     = $(amd64_LD)

136 LD              = $(LD_$(MACH)_$(CLASS))

138 LINT_sparc_32   = $(sparc_LINT)
139 LINT_sparc_64   = $(sparcv9_LINT)

141 LINT_i386_32    = $(i386_LINT)
142 LINT_i386_64    = $(amd64_LINT)
143 LINT_amd64_64   = $(amd64_LINT)

145 LINT            = $(LINT_$(MACH)_$(CLASS))

147 MODEL_32        = ilp32
148 MODEL_64        = lp64
149 MODEL           = $(MODEL_$(CLASS))

151 #
152 #       Build rules for linting the kernel.
153 #
154 LHEAD = $(ECHO) "\n$@";

156 # Note: egrep returns "failure" if there are no matches, which is
157 # exactly the opposite of what we need.
158 LGREP.2 =       if egrep -v ' (_init|_fini|_info) '; then false; else true; fi

160 LTAIL =

162 LINT.c =        $(LINT) -c -dirout=$(LINTS_DIR) $(LINTFLAGS) $(LINT_DEFS) $(CPPF

164 # Please do not add new erroff directives here.  If you need to disable
165 # lint warnings in your module for things that cannot be fixed in any
166 # reasonable manner, please augment LINTTAGS in your module Makefile
167 # instead.
168 LINTTAGS        = -erroff=E_INCONS_ARG_DECL2
169 LINTTAGS        += -erroff=E_INCONS_VAL_TYPE_DECL2

171 LINTFLAGS_sparc_32      = $(LINTCCMODE) -nsxmuF -errtags=yes
172 LINTFLAGS_sparc_64      = $(LINTFLAGS_sparc_32) -m64
173 LINTFLAGS_i386_32       = $(LINTCCMODE) -nsxmuF -errtags=yes
174 LINTFLAGS_i386_64       = $(LINTFLAGS_i386_32) -m64

176 LINTFLAGS       = $(LINTFLAGS_$(MACH)_$(CLASS)) $(LINTTAGS)
177 LINTFLAGS       += $(C99LMODE)

179 #
180 #       Override this variable to modify the name of the lint target.
181 #
182 LINT_MODULE=    $(MODULE)

184 #
185 #       Build the compile/assemble lines:
186 #
187 EXTRA_OPTIONS           =
188 AS_DEFS                 = -D_ASM -D__STDC__=0
```



```
190 ALWAYS_DEFS_32          = -D_KERNEL -D_SYSCALL32 -D_DDI_STRICT
191 ALWAYS_DEFS_64          = -D_KERNEL -D_SYSCALL32 -D_SYSCALL32_IMPL -D_ELF64 \
192                         -D_DDI_STRICT
193 #
194 # XX64  This should be defined by the compiler!
195 #
196 ALWAYS_DEFS_64          += -Dsun -D__sun -D__SVR4
197 ALWAYS_DEFS             = $(ALWAYS_DEFS_$(CLASS))

199 #
200 #       CPPFLAGS is deliberatly set with a "=" and not a "+=".  For the kernel
201 #       the header include path should not look for header files outside of
202 #       the kernel code.  This "=" removes the search path built in
203 #       Makefile.master inside CPPFLAGS.  Ditto for AS_CPPFLAGS.
204 #
205 CPPFLAGS        = $(ALWAYS_DEFS) $(ALL_DEFS) $(CONFIG_DEFS) \
206                     $(INCLUDE_PATH) $(EXTRA_OPTIONS)
207 ASFLAGS         += -P
208 AS_CPPFLAGS     = $(ALWAYS_DEFS) $(ALL_DEFS) $(CONFIG_DEFS) $(AS_DEFS) \
209                     $(AS_INC_PATH) $(EXTRA_OPTIONS)

211 #
212 #       Make it (relatively) easy to share compilation options between
213 #       all kernel implementations.
214 #

216 # Override the default, the kernel is squeaky clean
217 CERRWARN = -errtags=yes -errwarn=%all

219 CERRWARN += -_gcc=-Wno-missing-braces
220 CERRWARN += -_gcc=-Wno-sign-compare
221 CERRWARN += -_gcc=-Wno-unknown-pragmas
222 CERRWARN += -_gcc=-Wno-unused-parameter
223 CERRWARN += -_gcc=-Wno-missing-field-initializers

225 # DEBUG v. -nd make for frequent unused variables, empty conditions, etc. in
226 # -nd builds
227 $(RELEASE_BUILD)CERRWARN += -_gcc=-Wno-unused
228 $(RELEASE_BUILD)CERRWARN += -_gcc=-Wno-empty-body

230 CERRWARN += -_smatch=-p=illumos_kernel
231 include $(SRC)/Makefile.smatch

233 #
234 # Unfortunately, _IOWR() is regularly used with a third argument of 0,
235 # so we have to disable all these smatch checks.
236 #
237 SMOFF += sizeof

239 CSTD = $(CSTD_GNU99)

241 CFLAGS_uts              =
242 CFLAGS_uts              += $(STAND_FLAGS_$(CLASS))
243 CFLAGS_uts              += $(CCVERBOSE)
244 CFLAGS_uts              += $(ILDOFF)
245 CFLAGS_uts              += $(XAOPT)
246 CFLAGS_uts              += $(CTF_FLAGS_$(CLASS))
247 CFLAGS_uts              += $(CERRWARN)
248 CFLAGS_uts              += $(CCNOAUTOINLINE)
249 CFLAGS_uts              += $(CCNOREORDER)
250 CFLAGS_uts              += $(CCNOAGGRESSIVELOOPS)
251 CFLAGS_uts              += $(CGLOBALSTATIC)
252 CFLAGS_uts              += $(EXTRA_CFLAGS)
253 CFLAGS_uts              += $(CSOURCEDEBUGFLAGS)
254 CFLAGS_uts              += $(CUSERFLAGS)
```

```
256 #
257 #       Declare that $(OBJECTS) and $(LINTS) can be compiled in parallel.
258 #       The DUMMY target is for those instances where OBJECTS and LINTS
259 #       are empty (to avoid an unconditional .PARALLEL).
260 .PARALLEL:      $(OBJECTS) $(LINTS) DUMMY

262 #
263 #       Expanded dependencies
264 #
265 DEF_DEPS        = $(DEF_BUILDS:%=def.%)
266 ALL_DEPS        = $(ALL_BUILDS:%=all.%)
267 CLEAN_DEPS      = $(ALL_BUILDS:%=clean.%)
268 CLOBBER_DEPS    = $(ALL_BUILDS:%=clobber.%)
269 LINT_DEPS       = $(DEF_BUILDS:%=lint.%)
270 MODLINTLIB_DEPS = $(DEF_BUILDS:%=modlintlib.%)
271 MODLIST_DEPS    = $(DEF_BUILDS:%=modlist.%)
272 CLEAN_LINT_DEPS = $(ALL_BUILDS:%=clean.lint.%)
273 INSTALL_DEPS    = $(DEF_BUILDS:%=install.%)
274 SYM_DEPS        = $(SYM_BUILDS:%=symcheck.%)
275 SISCHECK_DEPS   = $(DEF_BUILDS:%=sischeck.%)
276 SISCLEAN_DEPS   = $(ALL_BUILDS:%=sisclean.%)

278 #
279 #       Default module name
280 #
281 BINARY          = $(OBJS_DIR)/$(MODULE)

283 #
284 #       Default cleanup definitions
285 #
286 CLEANLINTFILES  = $(LINTS) $(MOD_LINT_LIB)
287 CLEANFILES      = $(OBJECTS) $(CLEANLINTFILES)
288 CLOBBERFILES    = $(BINARY) $(CLEANFILES)

290 #
291 #       Installation constants:
292 #
293 #               FILEMODE is the mode given to the kernel modules
294 #               CFILEMODE is the mode given to the '.conf' files
295 #
296 FILEMODE         = 755
297 DIRMODE          = 755
298 CFILEMODE        = 644

300 #
301 #       Special Installation Macros for the installation of '.conf' files.
302 #
303 #       These are unique because they are not installed from the current
304 #       working directory.
305 #
306 # Sigh.  Apparently at some time in the past there was a confusion on
307 # whether the name is SRC_CONFFILE or SRC_CONFILE.  Consistency with the
308 # other names would indicate SRC_CONFFILE, but the voting is >180 Makefiles
309 # with SRC_CONFILE and about 11 with SRC_CONFFILE.  Software development
310 # isn't a popularity contest, though, and so my inclination is to define
311 # both names for now and incrementally convert to SRC_CONFFILE to be consistent
312 # with the other names.
313 #
314 CONFFILE                = $(MODULE).conf
315 SRC_CONFFILE            = $(CONF_SRCDIR)/$(CONFFILE)
316 SRC_CONFILE             = $(SRC_CONFFILE)
317 ROOT_CONFFILE_32        = $(ROOTMODULE).conf
318 ROOT_CONFFILE_64        = $(ROOTMODULE:%/$(SUBDIR64)/$(MODULE)=%/$(MODULE)).conf
319 ROOT_CONFFILE           = $(ROOT_CONFFILE_$(CLASS))
```



```
322 INS.conffile= \
323         $(RM) $@; $(INS) -s -m $(CFILEMODE) -f $(@D) $(SRC_CONFFILE)

325 #
326 # The CTF merge of child kernel modules is performed against one of the genunix
327 # modules.  For Intel builds, all modules will be used with a single genunix:
328 # the one built in intel/genunix.  For SPARC builds, a given
329 # module may be
330 # used with one of a number of genunix files, depending on what platform the
331 # module is deployed on.  We merge against the sun4u genunix to optimize for
332 # the common case.  We also merge against the ip driver since networking is
333 # typically loaded and types defined therein are shared between many modules.
334 #
335 CTFMERGE_GUDIR_sparc    = sun4u
336 CTFMERGE_GUDIR_i386     = intel
337 CTFMERGE_GUDIR          = $(CTFMERGE_GUDIR_$(MACH))

339 CTFMERGE_GENUNIX        = \
340         $(UTSBASE)/$(CTFMERGE_GUDIR)/genunix/$(OBJS_DIR)/genunix

342 #
343 # Used to uniquify a non-genunix module against genunix. $VERSION is used
344 # for the label.
345 #
346 # For the ease of developers dropping modules onto possibly unrelated systems,
347 # you can set NO_GENUNIX_UNIQUIFY= in the environment to skip uniquifying
348 # against genunix.
349 #
350 NO_GENUNIX_UNIQUIFY=$(POUND_SIGN)
351 CTFMERGE_GENUNIX_DFLAG=-d $(CTFMERGE_GENUNIX)
352 $(NO_GENUNIX_UNIQUIFY)CTF_GENUNIX_DFLAG=

354 CTFMERGE_UNIQUIFY_AGAINST_GENUNIX       = \
355         $(CTFMERGE) $(CTFMRGFLAGS) -L VERSION \
356         $(CTFMERGE_GENUNIX_DFLAG) -o $@ $(OBJECTS) $(CTFEXTRAOBJS)

358 #
359 # Used to merge the genunix module.
360 #
361 CTFMERGE_GENUNIX_MERGE          = \
362         $(CTFMERGE) $(CTFMRGFLAGS) -L VERSION -o $@ \
363         $(OBJECTS) $(CTFEXTRAOBJS) $(IPCTF_TARGET)

365 #
366 # We ctfmerge the ip objects into genunix to maximize the number of common types
367 # found there, thus maximizing the effectiveness of uniquification.  We don't
368 # want the genunix build to have to know about the individual ip objects, so we
369 # put them in an archive.  The genunix ctfmerge then includes this archive.
370 #
371 IPCTF           = $(IPDRV_DIR)/$(OBJS_DIR)/ipctf.a

373 #
374 # Rule for building fake shared libraries used for symbol resolution
375 # when building other modules.  -znoreloc is needed here to avoid
376 # tripping over code that isn't really suitable for shared libraries.
377 #
378 BUILD.SO                = \
379         $(LD) -o $@ $(GSHARED) $(ZNORELOC) -h $(SONAME)

381 #
382 # SONAME defaults for common fake shared libraries.
383 #
384 $(LIBGEN)               := SONAME = $(MODULE)
385 $(PLATLIB)              := SONAME = misc/platmod
386 $(CPULIB)               := SONAME = 'cpu/$$CPU'
387 $(DTRACESTUBS)          := SONAME = dtracestubs
```

```
389 #
390 #       Installation directories
391 #

393 #
394 #       For now, 64b modules install into a subdirectory
395 #       of their 32b brethren.
396 #
397 SUBDIR64_sparc          = sparcv9
398 SUBDIR64_i386           = amd64
399 SUBDIR64                = $(SUBDIR64_$(MACH))

401 ROOT_MOD_DIR            = $(ROOT)/kernel

403 ROOT_KERN_DIR_32        = $(ROOT_MOD_DIR)
404 ROOT_BRAND_DIR_32       = $(ROOT_MOD_DIR)/brand
405 ROOT_DRV_DIR_32         = $(ROOT_MOD_DIR)/drv
406 ROOT_DTRACE_DIR_32      = $(ROOT_MOD_DIR)/dtrace
407 ROOT_EXEC_DIR_32        = $(ROOT_MOD_DIR)/exec
408 ROOT_FS_DIR_32          = $(ROOT_MOD_DIR)/fs
409 ROOT_SCHED_DIR_32       = $(ROOT_MOD_DIR)/sched
410 ROOT_SOCK_DIR_32        = $(ROOT_MOD_DIR)/socketmod
411 ROOT_STRMOD_DIR_32      = $(ROOT_MOD_DIR)/strmod
412 ROOT_IPP_DIR_32         = $(ROOT_MOD_DIR)/ipp
413 ROOT_SYS_DIR_32         = $(ROOT_MOD_DIR)/sys
414 ROOT_MISC_DIR_32        = $(ROOT_MOD_DIR)/misc
415 ROOT_KGSS_DIR_32        = $(ROOT_MOD_DIR)/misc/kgss
416 ROOT_SCSI_VHCI_DIR_32   = $(ROOT_MOD_DIR)/misc/scsi_vhci
417 ROOT_PMCS_FW_DIR_32     = $(ROOT_MOD_DIR)/misc/pmcs
418 ROOT_QLC_FW_DIR_32      = $(ROOT_MOD_DIR)/misc/qlc
419 ROOT_EMLXS_FW_DIR_32    = $(ROOT_MOD_DIR)/misc/emlxs
420 ROOT_NLMISC_DIR_32      = $(ROOT_MOD_DIR)/misc
421 ROOT_MACH_DIR_32        = $(ROOT_MOD_DIR)/mach
422 ROOT_CPU_DIR_32         = $(ROOT_MOD_DIR)/cpu
423 ROOT_TOD_DIR_32         = $(ROOT_MOD_DIR)/tod
424 ROOT_FONT_DIR_32        = $(ROOT_MOD_DIR)/fonts
425 ROOT_DACF_DIR_32        = $(ROOT_MOD_DIR)/dacf
426 ROOT_CRYPTO_DIR_32      = $(ROOT_MOD_DIR)/crypto
427 ROOT_MAC_DIR_32         = $(ROOT_MOD_DIR)/mac
428 ROOT_KICONV_DIR_32      = $(ROOT_MOD_DIR)/kiconv

430 ROOT_KERN_DIR_64        = $(ROOT_MOD_DIR)/$(SUBDIR64)
431 ROOT_BRAND_DIR_64       = $(ROOT_MOD_DIR)/brand/$(SUBDIR64)
432 ROOT_DRV_DIR_64         = $(ROOT_MOD_DIR)/drv/$(SUBDIR64)
433 ROOT_DTRACE_DIR_64      = $(ROOT_MOD_DIR)/dtrace/$(SUBDIR64)
434 ROOT_EXEC_DIR_64        = $(ROOT_MOD_DIR)/exec/$(SUBDIR64)
435 ROOT_FS_DIR_64          = $(ROOT_MOD_DIR)/fs/$(SUBDIR64)
436 ROOT_SCHED_DIR_64       = $(ROOT_MOD_DIR)/sched/$(SUBDIR64)
437 ROOT_SOCK_DIR_64        = $(ROOT_MOD_DIR)/socketmod/$(SUBDIR64)
438 ROOT_STRMOD_DIR_64      = $(ROOT_MOD_DIR)/strmod/$(SUBDIR64)
439 ROOT_IPP_DIR_64         = $(ROOT_MOD_DIR)/ipp/$(SUBDIR64)
440 ROOT_SYS_DIR_64         = $(ROOT_MOD_DIR)/sys/$(SUBDIR64)
441 ROOT_MISC_DIR_64        = $(ROOT_MOD_DIR)/misc/$(SUBDIR64)
442 ROOT_KGSS_DIR_64        = $(ROOT_MOD_DIR)/misc/kgss/$(SUBDIR64)
443 ROOT_SCSI_VHCI_DIR_64   = $(ROOT_MOD_DIR)/misc/scsi_vhci/$(SUBDIR64)
444 ROOT_PMCS_FW_DIR_64     = $(ROOT_MOD_DIR)/misc/pmcs/$(SUBDIR64)
445 ROOT_QLC_FW_DIR_64      = $(ROOT_MOD_DIR)/misc/qlc/$(SUBDIR64)
446 ROOT_EMLXS_FW_DIR_64    = $(ROOT_MOD_DIR)/misc/emlxs/$(SUBDIR64)
447 ROOT_NLMISC_DIR_64      = $(ROOT_MOD_DIR)/misc/$(SUBDIR64)
448 ROOT_MACH_DIR_64        = $(ROOT_MOD_DIR)/mach/$(SUBDIR64)
449 ROOT_CPU_DIR_64         = $(ROOT_MOD_DIR)/cpu/$(SUBDIR64)
450 ROOT_TOD_DIR_64         = $(ROOT_MOD_DIR)/tod/$(SUBDIR64)
451 ROOT_FONT_DIR_64        = $(ROOT_MOD_DIR)/fonts/$(SUBDIR64)
452 ROOT_DACF_DIR_64        = $(ROOT_MOD_DIR)/dacf/$(SUBDIR64)
453 ROOT_CRYPTO_DIR_64      = $(ROOT_MOD_DIR)/crypto/$(SUBDIR64)
```



```
454 ROOT_MAC_DIR_64         = $(ROOT_MOD_DIR)/mac/$(SUBDIR64)
455 ROOT_KICONV_DIR_64      = $(ROOT_MOD_DIR)/kiconv/$(SUBDIR64)

457 ROOT_KERN_DIR           = $(ROOT_KERN_DIR_$(CLASS))
458 ROOT_BRAND_DIR          = $(ROOT_BRAND_DIR_$(CLASS))
459 ROOT_DRV_DIR            = $(ROOT_DRV_DIR_$(CLASS))
460 ROOT_DTRACE_DIR         = $(ROOT_DTRACE_DIR_$(CLASS))
461 ROOT_EXEC_DIR           = $(ROOT_EXEC_DIR_$(CLASS))
462 ROOT_FS_DIR             = $(ROOT_FS_DIR_$(CLASS))
463 ROOT_SCHED_DIR          = $(ROOT_SCHED_DIR_$(CLASS))
464 ROOT_SOCK_DIR           = $(ROOT_SOCK_DIR_$(CLASS))
465 ROOT_STRMOD_DIR         = $(ROOT_STRMOD_DIR_$(CLASS))
466 ROOT_IPP_DIR            = $(ROOT_IPP_DIR_$(CLASS))
467 ROOT_SYS_DIR            = $(ROOT_SYS_DIR_$(CLASS))
468 ROOT_MISC_DIR           = $(ROOT_MISC_DIR_$(CLASS))
469 ROOT_KGSS_DIR           = $(ROOT_KGSS_DIR_$(CLASS))
470 ROOT_SCSI_VHCI_DIR      = $(ROOT_SCSI_VHCI_DIR_$(CLASS))
471 ROOT_PMCS_FW_DIR        = $(ROOT_PMCS_FW_DIR_$(CLASS))
472 ROOT_QLC_FW_DIR         = $(ROOT_QLC_FW_DIR_$(CLASS))
473 ROOT_EMLXS_FW_DIR       = $(ROOT_EMLXS_FW_DIR_$(CLASS))
474 ROOT_NLMISC_DIR         = $(ROOT_NLMISC_DIR_$(CLASS))
475 ROOT_MACH_DIR           = $(ROOT_MACH_DIR_$(CLASS))
476 ROOT_CPU_DIR            = $(ROOT_CPU_DIR_$(CLASS))
477 ROOT_TOD_DIR            = $(ROOT_TOD_DIR_$(CLASS))
478 ROOT_FONT_DIR           = $(ROOT_FONT_DIR_$(CLASS))
479 ROOT_DACF_DIR           = $(ROOT_DACF_DIR_$(CLASS))
480 ROOT_CRYPTO_DIR         = $(ROOT_CRYPTO_DIR_$(CLASS))
481 ROOT_MAC_DIR            = $(ROOT_MAC_DIR_$(CLASS))
482 ROOT_KICONV_DIR         = $(ROOT_KICONV_DIR_$(CLASS))
483 ROOT_FIRMWARE_DIR       = $(ROOT_MOD_DIR)/firmware

485 ROOT_MOD_DIRS_32        = $(ROOT_BRAND_DIR_32) $(ROOT_DRV_DIR_32)
486 ROOT_MOD_DIRS_32        = $(ROOT_BRAND_DIR_32) $(ROOT_DRV_DIR_32)
487 ROOT_MOD_DIRS_32        += $(ROOT_EXEC_DIR_32) $(ROOT_DTRACE_DIR_32)
488 ROOT_MOD_DIRS_32        += $(ROOT_FS_DIR_32) $(ROOT_SCHED_DIR_32)
489 ROOT_MOD_DIRS_32        += $(ROOT_STRMOD_DIR_32) $(ROOT_SYS_DIR_32)
490 ROOT_MOD_DIRS_32        += $(ROOT_IPP_DIR_32) $(ROOT_SOCK_DIR_32)
491 ROOT_MOD_DIRS_32        += $(ROOT_MISC_DIR_32) $(ROOT_MACH_DIR_32)
492 ROOT_MOD_DIRS_32        += $(ROOT_KGSS_DIR_32)
493 ROOT_MOD_DIRS_32        += $(ROOT_SCSI_VHCI_DIR_32)
494 ROOT_MOD_DIRS_32        += $(ROOT_PMCS_FW_DIR_32)
495 ROOT_MOD_DIRS_32        += $(ROOT_QLC_FW_DIR_32)
496 ROOT_MOD_DIRS_32        += $(ROOT_EMLXS_FW_DIR_32)
497 ROOT_MOD_DIRS_32        += $(ROOT_CPU_DIR_32) $(ROOT_FONT_DIR_32)
498 ROOT_MOD_DIRS_32        += $(ROOT_TOD_DIR_32) $(ROOT_DACF_DIR_32)
499 ROOT_MOD_DIRS_32        += $(ROOT_CRYPTO_DIR_32) $(ROOT_MAC_DIR_32)
500 ROOT_MOD_DIRS_32        += $(ROOT_KICONV_DIR_32)
501 ROOT_MOD_DIRS_32        += $(ROOT_FIRMWARE_DIR)

503 USR_MOD_DIR            = $(ROOT)/usr/kernel

505 USR_DRV_DIR_32          = $(USR_MOD_DIR)/drv
506 USR_EXEC_DIR_32         = $(USR_MOD_DIR)/exec
507 USR_FS_DIR_32           = $(USR_MOD_DIR)/fs
508 USR_SCHED_DIR_32        = $(USR_MOD_DIR)/sched
509 USR_SOCK_DIR_32         = $(USR_MOD_DIR)/socketmod
510 USR_STRMOD_DIR_32       = $(USR_MOD_DIR)/strmod
511 USR_SYS_DIR_32          = $(USR_MOD_DIR)/sys
512 USR_MISC_DIR_32         = $(USR_MOD_DIR)/misc
513 USR_DACF_DIR_32         = $(USR_MOD_DIR)/dacf
514 USR_PCBE_DIR_32         = $(USR_MOD_DIR)/pcbe
515 USR_DTRACE_DIR_32       = $(USR_MOD_DIR)/dtrace
516 USR_BRAND_DIR_32        = $(USR_MOD_DIR)/brand

518 USR_DRV_DIR_64          = $(USR_MOD_DIR)/drv/$(SUBDIR64)
519 USR_EXEC_DIR_64         = $(USR_MOD_DIR)/exec/$(SUBDIR64)
```

```
USR_FS_DIR_64           = $(USR_MOD_DIR)/fs/$(SUBDIR64)
USR_SCHED_DIR_64        = $(USR_MOD_DIR)/sched/$(SUBDIR64)
USR_SOCK_DIR_64         = $(USR_MOD_DIR)/socketmod/$(SUBDIR64)
USR_STRMOD_DIR_64       = $(USR_MOD_DIR)/strmod/$(SUBDIR64)
USR_SYS_DIR_64          = $(USR_MOD_DIR)/sys/$(SUBDIR64)
USR_MISC_DIR_64         = $(USR_MOD_DIR)/misc/$(SUBDIR64)
USR_DACF_DIR_64         = $(USR_MOD_DIR)/dacf/$(SUBDIR64)
USR_PCBE_DIR_64         = $(USR_MOD_DIR)/pcbe/$(SUBDIR64)
USR_DTRACE_DIR_64       = $(USR_MOD_DIR)/dtrace/$(SUBDIR64)
USR_BRAND_DIR_64        = $(USR_MOD_DIR)/brand/$(SUBDIR64)

USR_DRV_DIR             = $(USR_DRV_DIR_$(CLASS))
USR_EXEC_DIR            = $(USR_EXEC_DIR_$(CLASS))
USR_FS_DIR              = $(USR_FS_DIR_$(CLASS))
USR_SCHED_DIR           = $(USR_SCHED_DIR_$(CLASS))
USR_SOCK_DIR            = $(USR_SOCK_DIR_$(CLASS))
USR_STRMOD_DIR          = $(USR_STRMOD_DIR_$(CLASS))
USR_SYS_DIR             = $(USR_SYS_DIR_$(CLASS))
USR_MISC_DIR            = $(USR_MISC_DIR_$(CLASS))
USR_DACF_DIR            = $(USR_DACF_DIR_$(CLASS))
USR_PCBE_DIR            = $(USR_PCBE_DIR_$(CLASS))
USR_DTRACE_DIR          = $(USR_DTRACE_DIR_$(CLASS))
USR_BRAND_DIR           = $(USR_BRAND_DIR_$(CLASS))

USR_MOD_DIRS_32         = $(USR_DRV_DIR_32) $(USR_EXEC_DIR_32)
USR_MOD_DIRS_32         += $(USR_FS_DIR_32) $(USR_SCHED_DIR_32)
USR_MOD_DIRS_32         += $(USR_STRMOD_DIR_32) $(USR_SYS_DIR_32)
USR_MOD_DIRS_32         += $(USR_MISC_DIR_32) $(USR_DACF_DIR_32)
USR_MOD_DIRS_32         += $(USR_PCBE_DIR_32)
USR_MOD_DIRS_32         += $(USR_DTRACE_DIR_32) $(USR_BRAND_DIR_32)
USR_MOD_DIRS_32         += $(USR_SOCK_DIR_32)

#
#
#
include $(SRC)/Makefile.psm

#
#       The "-r" on the remove may be considered temporary, but is required
#       while the replacement of the SUNW,SPARCstation-10,SX directory by
#       a symbolic link is being propagated.
#
INS.slink1= $(RM) -r $@; $(SYMLINK) $(PLATFORM) $@
INS.slink2= $(RM) -r $@; $(SYMLINK) ../$(PLATFORM)/$(@F) $@
INS.slink3= $(RM) -r $@; $(SYMLINK) $(IMPLEMENTED_PLATFORM) $@
INS.slink4= $(RM) -r $@; $(SYMLINK) ../$(PLATFORM)/include $@
INS.slink5= $(RM) -r $@; $(SYMLINK) ../$(PLATFORM)/sbin $@
INS.slink6= $(RM) -r $@; $(SYMLINK) ../../$(PLATFORM)/lib/$(MODULE) $@
INS.slink7= $(RM) -r $@; $(SYMLINK) ../../$(PLATFORM)/sbin/$(@F) $@

ROOT_PLAT_LINKS          = $(PLAT_LINKS:%=$(ROOT_PLAT_DIR)/%)
ROOT_PLAT_LINKS_2        = $(PLAT_LINKS_2:%=$(ROOT_PLAT_DIR)/%)
USR_PLAT_LINKS           = $(PLAT_LINKS:%=$(USR_PLAT_DIR)/%)
USR_PLAT_LINKS_2         = $(PLAT_LINKS_2:%=$(USR_PLAT_DIR)/%)

#
# Collection of all relevant, delivered kernel modules.
#
# Note that we insist on building genunix first, because everything else
# uniquifies against it.  When doing a 'make' from usr/src/uts/, we'll enter
# the platform directories first.  These will cd into the corresponding genunix
# directory and build it.  So genunix /shouldn't/ get rebuilt when we get to
# building all the kernel modules.  However, due to an as-yet-unexplained
# problem with dependencies, sometimes it does get rebuilt, which then messes
# up the other modules.  So we always force the issue here rather than try to
# build genunix in parallel with everything else.
```



```
#
PARALLEL_KMODS = $(DRV_KMODS) $(EXEC_KMODS) $(FS_KMODS) $(SCHED_KMODS) \
                 $(TOD_KMODS) $(STRMOD_KMODS) $(SYS_KMODS) $(MISC_KMODS) \
                 $(NLMISC_KMODS) $(MACH_KMODS) $(CPU_KMODS) $(GSS_KMODS) \
                 $(MMU_KMODS) $(DACF_KMODS) $(EXPORT_KMODS) $(IPP_KMODS) \
                 $(CRYPTO_KMODS) $(PCBE_KMODS) \
                 $(DRV_KMODS_$(CLASS)) $(MISC_KMODS_$(CLASS)) $(MAC_KMODS) \
                 $(BRAND_KMODS) $(KICONV_KMODS) \
                 $(SOCKET_KMODS)

KMODS = $(GENUNIX_KMODS) $(PARALLEL_KMODS)

$(PARALLEL_KMODS): $(GENUNIX_KMODS)

LINT_KMODS = $(DRV_KMODS) $(EXEC_KMODS) $(FS_KMODS) $(SCHED_KMODS) \
          $(TOD_KMODS) $(STRMOD_KMODS) $(SYS_KMODS) $(MISC_KMODS) \
          $(MACH_KMODS) $(GSS_KMODS) $(DACF_KMODS) $(IPP_KMODS) \
          $(CRYPTO_KMODS) $(PCBE_KMODS) \
          $(DRV_KMODS_$(CLASS)) $(MISC_KMODS_$(CLASS)) $(MAC_KMODS) \
          $(BRAND_KMODS) $(KICONV_KMODS) $(SOCKET_KMODS)

#
#       Files to be compiled with -xa, to generate basic block execution
#       count data.
#
#       There are several ways to compile parts of the kernel for kcov:
#               1)  Add targets to BB_FILES here or in other Makefiles
#                       (they must in the form of $(OBJS_DIR)/target.o)
#               2)  setenv BB_FILES '$(XXX_OBJS:%=$(OBJS_DIR)/%)'
#               3)  setenv BB_FILES '$(OBJECTS)'
#
#       Do NOT setenv CFLAGS -xa, as that will cause infinite recursion
#       in unix_bb.o
#
BB_FILES =
$(BB_FILES)     := XAOPT = -xa

#
#       The idea here is for unix_bb.o to be in all kernels except the
#       kernel which actually gets shipped to customers.  In practice,
#       $(RELEASE_BUILD) is on for a number of the late beta and fcs builds.
#
$(NOT_RELEASE_BUILD)$(OBJS_DIR)/unix_bb.o   := CPPFLAGS     += -DKCOV
$(NOT_RELEASE_BUILD)$(OBJS_DIR)/unix_bb.ln  := CPPFLAGS     += -DKCOV

#
#       Do not let unix_bb.o get compiled with -xa!
#
$(OBJS_DIR)/unix_bb.o   := XAOPT =

#
# Privilege files
#
PRIVS_AWK = $(SRC)/uts/common/os/privs.awk
PRIVS_DEF = $(SRC)/uts/common/os/priv_defs

#
# USB device data
#
USBDEVS_AWK =   $(SRC)/uts/common/io/usb/usbdevs2h.awk
USBDEVS_DATA =  $(SRC)/uts/common/io/usb/usbdevs


#
# If we're using the newer CTF tools, then we need to make sure that we
# are building with the private -X option to ctfconvert which allows us
```

```
652 # to fixup the struct cpu to account for machcpu.
653 #
654 $(BUILD_NEW_CTF_TOOLS)CTFCVTFLAGS += -X
```