```
**********************************************************
   36524 Tue Oct 30 20:22:47 2018
new/usr/src/Makefile.master
9939 Need to stop GCC reordering functions
**********************************************************
#
# 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.
#

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

```
GNUC_ROOT=      /opt/gcc/4.4.4
GCCLIBDIR=      $(GNUC_ROOT)/lib
GCCLIBDIR64=    $(GNUC_ROOT)/lib/$(MACH64)

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

RPCGEN=         /usr/bin/rpcgen
STABS=          $(ONBLD_TOOLS)/bin/$(MACH)/stabs
ELFEXTRACT=     $(ONBLD_TOOLS)/bin/$(MACH)/elfextract
MBH_PATCH=      $(ONBLD_TOOLS)/bin/$(MACH)/mbh_patch
BTXLD=          $(ONBLD_TOOLS)/bin/$(MACH)/btxld
VTFONTCVT=      $(ONBLD_TOOLS)/bin/$(MACH)/vtfontcvt
# echo(1) and true(1) are specified without absolute paths, so that the shell
# spawned by make(1) may use the built-in versions.  This is minimally
# problematic, as the shell spawned by make(1) is known and under control, the
# only risk being if the shell falls back to $PATH.
#
# We specifically want an echo(1) that does interpolation of escape sequences,
# which ksh93, /bin/sh, and bash will all provide.
ECHO=           echo
TRUE=           true
INS=            $(ONBLD_TOOLS)/bin/$(MACH)/install
SYMLINK=        /usr/bin/ln -s
LN=             /usr/bin/ln
MKDIR=          /usr/bin/mkdir
CHMOD=          /usr/bin/chmod
MV=             /usr/bin/mv -f
RM=             /usr/bin/rm -f
CUT=            /usr/bin/cut
NM=             /usr/ccs/bin/nm
DIFF=           /usr/bin/diff
GREP=           /usr/bin/grep
EGREP=          /usr/bin/egrep
ELFWRAP=        /usr/bin/elfwrap
KSH93=          /usr/bin/ksh93
SED=            /usr/bin/sed
AWK=            /usr/bin/nawk
CP=             /usr/bin/cp -f
MCS=            /usr/ccs/bin/mcs
CAT=            /usr/bin/cat
ELFDUMP=        /usr/ccs/bin/elfdump
M4=             /usr/bin/m4
STRIP=          /usr/ccs/bin/strip
LEX=            /usr/ccs/bin/lex
FLEX=           /usr/bin/flex
YACC=           /usr/ccs/bin/yacc
CPP=            /usr/lib/cpp
ANSI_CPP=       $(GNUC_ROOT)/bin/cpp
JAVAC=          $(JAVA_ROOT)/bin/javac
JAVAH=          $(JAVA_ROOT)/bin/javah
JAVADOC=        $(JAVA_ROOT)/bin/javadoc
RMIC=           $(JAVA_ROOT)/bin/rmic
JAR=            $(JAVA_ROOT)/bin/jar
CTFCONVERT=     $(ONBLD_TOOLS)/bin/$(MACH)/ctfconvert
CTFMERGE=       $(ONBLD_TOOLS)/bin/$(MACH)/ctfmerge
CTFSTABS=       $(ONBLD_TOOLS)/bin/$(MACH)/ctfstabs
CTFSTRIP=       $(ONBLD_TOOLS)/bin/$(MACH)/ctfstrip
NDRGEN=         $(ONBLD_TOOLS)/bin/$(MACH)/ndrgen
GENOFFSETS=     $(ONBLD_TOOLS)/bin/genoffsets
XREF=           $(ONBLD_TOOLS)/bin/xref
FIND=           /usr/bin/find
PERL=           /usr/bin/perl
PERL_VERSION=   5.10.0
PERL_PKGVERS=   -510
PERL_ARCH =             i86pc-solaris-64int
$(SPARC_BLD)PERL_ARCH = sun4-solaris-64int
```

```
PYTHON_VERSION= 2.7
PYTHON_PKGVERS= -27
PYTHON_SUFFIX=
PYTHON=         /usr/bin/python$(PYTHON_VERSION)
PYTHON3_VERSION=        3.5
PYTHON3_PKGVERS=        -35
PYTHON3_SUFFIX=         m
PYTHON3=                /usr/bin/python$(PYTHON3_VERSION)
SORT=           /usr/bin/sort
TOUCH=          /usr/bin/touch
WC=             /usr/bin/wc
XARGS=          /usr/bin/xargs
ELFEDIT=        /usr/bin/elfedit
DTRACE=         /usr/sbin/dtrace -xnolibs
UNIQ=           /usr/bin/uniq
TAR=            /usr/bin/tar
ASTBINDIR=      /usr/ast/bin
MSGCC=          $(ASTBINDIR)/msgcc
MSGFMT=         /usr/bin/msgfmt -s
LCDEF=          $(ONBLD_TOOLS)/bin/$(MACH)/localedef
TIC=            $(ONBLD_TOOLS)/bin/$(MACH)/tic
ZIC=            $(ONBLD_TOOLS)/bin/$(MACH)/zic
OPENSSL=        /usr/bin/openssl

FILEMODE=       644
DIRMODE=        755

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

# For stylistic checks
#
# Note that the X and C checks are not used at this time and may need
# modification when they are actually used.
#
CSTYLE=         $(ONBLD_TOOLS)/bin/cstyle
CSTYLE_TAIL=
HDRCHK=         $(ONBLD_TOOLS)/bin/hdrchk
HDRCHK_TAIL=
JSTYLE=         $(ONBLD_TOOLS)/bin/jstyle

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

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

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

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

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

```
# install a link
INSLINKTARGET=  $<
INS.link=       $(RM) $@; $(LN) $(INSLINKTARGET) $@
INS.symlink=    $(RM) $@; $(SYMLINK) $(INSLINKTARGET) $@

#
# Python bakes the mtime of the .py file into the compiled .pyc and
# rebuilds if the baked-in mtime != the mtime of the source file
# (rather than only if it's less than), thus when installing python
# files we must make certain to not adjust the mtime of the source
# (.py) file.
#
INS.pyfile=     $(RM) $@; $(SED) -e "1s:^\#!@PYTHON@:\#!$(PYTHON):" < $< > $@; $

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

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

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

sparc_BUILD64=
i386_BUILD64=
BUILD64=        $($(MACH)_BUILD64)

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

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

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

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

# Prevent the removal of static symbols by the SPARC code generator (cg).
# The x86 code generator (ube) does not remove such symbols and as such
# using this workaround is not applicable for x86.
#
```

```
CCSTATICSYM=            -Wc,-Qassembler-ounrefsym=0
#
# generate 32-bit addresses in the v9 kernel. Saves memory.
CCABS32=                -Wc,-xcode=abs32
#
# generate v9 code which tolerates callers using the v7 ABI, for the sake of
# system calls.
CC32BITCALLERS=         -_gcc=-massume-32bit-callers

# GCC, especially, is increasingly beginning to auto-inline functions and
# sadly does so separately not under the general -fno-inline-functions
# Additionally, we wish to prevent optimisations which cause GCC to clone
# functions -- in particular, these may cause unhelpful symbols to be
# emitted instead of function names
CCNOAUTOINLINE= \
        -_gcc=-fno-inline-small-functions \
        -_gcc=-fno-inline-functions-called-once \
        -_gcc=-fno-ipa-cp \
        -_gcc6=-fno-ipa-icf \
        -_gcc7=-fno-ipa-icf \
        -_gcc8=-fno-ipa-icf \
        -_gcc6=-fno-clone-functions \
        -_gcc7=-fno-clone-functions \
        -_gcc8=-fno-clone-functions \

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

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

#
# compiler '-xarch' flag. This is here to centralize it and make it
# overridable for testing.
sparc_XARCH=    -m32
sparcv9_XARCH=  -m64
i386_XARCH=     -m32
amd64_XARCH=    -m64 -Ui386 -U__i386

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

#
```

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

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

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

#
# disable the incremental linker
ILDOFF=                 -xildoff
#
XFFLAG=                 -xF=%all
XESS=                   -xs
XSTRCONST=              -xstrconst

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

CERRWARN += -_gcc=-Wno-missing-braces
CERRWARN += -_gcc=-Wno-sign-compare
CERRWARN += -_gcc=-Wno-unknown-pragmas
CERRWARN += -_gcc=-Wno-unused-parameter
CERRWARN += -_gcc=-Wno-missing-field-initializers

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

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

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

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

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

```
sparc_ASFLAGS=  $(sparc_AS_XARCH)
sparcv9_ASFLAGS=$(sparcv9_AS_XARCH)
i386_ASFLAGS=   $(i386_AS_XARCH)
amd64_ASFLAGS=  $(amd64_AS_XARCH)

#
sparc_COPTFLAG=         -xO3
sparcv9_COPTFLAG=       -xO3
i386_COPTFLAG=          -O
amd64_COPTFLAG=         -xO3

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

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

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

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

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

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

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

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

CTF_FLAGS_32    = $(CTF_FLAGS_$(MACH)) $(DEBUGFORMAT)
CTF_FLAGS_64    = $(CTF_FLAGS_$(MACH64)) $(DEBUGFORMAT)
CTF_FLAGS       = $(CTF_FLAGS_32)

#
# Flags used with genoffsets
#
GOFLAGS = $(CALLSYMS) $(CDWARFSTR)

OFFSETS_CREATE = $(GENOFFSETS) -s $(CTFSTABS) -r $(CTFCONVERT) \
        $(CW) --noecho $(CW_CC_COMPILERS) -- $(GOFLAGS) $(CFLAGS) $(CPPFLAGS)

OFFSETS_CREATE64 = $(GENOFFSETS) -s $(CTFSTABS) -r $(CTFCONVERT) \
        $(CW) --noecho $(CW_CC_COMPILERS) -- $(GOFLAGS) $(CFLAGS64) $(CPPFLAGS)

#
# tradeoff time for space (smaller is better)
```

```
#
sparc_SPACEFLAG         = -xspace -W0,-Lt
sparcv9_SPACEFLAG       = -xspace -W0,-Lt
i386_SPACEFLAG          = -xspace
amd64_SPACEFLAG         =

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

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

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

sparc_XREGSFLAG         = -xregs=no%appl
sparcv9_XREGSFLAG       = -xregs=no%appl
i386_XREGSFLAG          =
amd64_XREGSFLAG         =

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

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

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

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

CFLAGS=         $(COPTFLAG) $($(MACH)_CFLAGS) $(SPACEFLAG) $(CCMODE) \
                $(ILDOFF) $(CERRWARN) $(CSTD) $(CCUNBOUND) $(IROPTFLAG) \
                $(CGLOBALSTATIC) $(CCNOAUTOINLINE) $(CCNOREORDER) \
                $(CSOURCEDEBUGFLAGS) $(CUSERFLAGS)
                $(CGLOBALSTATIC) $(CCNOAUTOINLINE) $(CSOURCEDEBUGFLAGS) \
                $(CUSERFLAGS)
CFLAGS64=       $(COPTFLAG64) $($(MACH64)_CFLAGS) $(SPACEFLAG64) $(CCMODE64) \
                $(ILDOFF) $(CERRWARN) $(CSTD) $(CCUNBOUND) $(IROPTFLAG64) \
                $(CGLOBALSTATIC) $(CCNOAUTOINLINE) $(CCNOREORDER) \
                $(CSOURCEDEBUGFLAGS) $(CUSERFLAGS64)
                $(CGLOBALSTATIC) $(CCNOAUTOINLINE) $(CSOURCEDEBUGFLAGS) \
                $(CUSERFLAGS64)
```

```
#
# Flags that are used to build parts of the code that are subsequently
# run on the build machine (also known as the NATIVE_BUILD).
#
NATIVE_CFLAGS=  $(COPTFLAG) $($(NATIVE_MACH)_CFLAGS) $(CCMODE) \
                $(ILDOFF) $(CERRWARN) $(CSTD) $($(NATIVE_MACH)_CCUNBOUND) \
                $(IROPTFLAG) $(CGLOBALSTATIC) $(CCNOAUTOINLINE) \
                $(CCNOREORDER) $(CSOURCEDEBUGFLAGS) $(CUSERFLAGS)
                $(CSOURCEDEBUGFLAGS) $(CUSERFLAGS)

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

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

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

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

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

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

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

#
#
#
ELFWRAP_FLAGS   =
ELFWRAP_FLAGS64 =       -64

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

#
# Generated mapfiles that are compiler specific, and used throughout the
# build.  These mapfiles are not delivered in /usr/lib/ld.
#
MAPFILE.NGB_sparc=      $(SRC)/common/mapfiles/gen/sparc_cc_map.noexeglobs
$(__GNUC64)MAPFILE.NGB_sparc= \
                        $(SRC)/common/mapfiles/gen/sparc_gcc_map.noexeglobs
MAPFILE.NGB_sparcv9=    $(SRC)/common/mapfiles/gen/sparcv9_cc_map.noexeglobs
$(__GNUC64)MAPFILE.NGB_sparcv9= \
                        $(SRC)/common/mapfiles/gen/sparcv9_gcc_map.noexeglobs
MAPFILE.NGB_i386=       $(SRC)/common/mapfiles/gen/i386_cc_map.noexeglobs
$(__GNUC64)MAPFILE.NGB_i386= \
                        $(SRC)/common/mapfiles/gen/i386_gcc_map.noexeglobs
MAPFILE.NGB_amd64=      $(SRC)/common/mapfiles/gen/amd64_cc_map.noexeglobs
$(__GNUC64)MAPFILE.NGB_amd64= \
                        $(SRC)/common/mapfiles/gen/amd64_gcc_map.noexeglobs
MAPFILE.NGB =           $(MAPFILE.NGB_$(MACH))

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

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

```
LDLIBS.lib =    $(LDLIBS32)

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

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

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

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

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

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

# This forces lint to pick up note.h and sys/note.h from Devpro rather than
# from the proto area.  The note.h that ON delivers would disable NOTE().
```

```
ONLY_LINT_DEFS =        -I$(SPRO_VROOT)/prod/include/lint

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

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

# Define native compilation macros
#

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

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

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


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

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

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

$(MACH)_CC=             $(BUILD_CC)
$(MACH)_CCC=            $(BUILD_CCC)
$(MACH)_CPP=            $(BUILD_CPP)
$(MACH)_LD=             $(BUILD_LD)
$(MACH)_LINT=           $(BUILD_LINT)
$(MACH64)_CC=           $(BUILD_CC)
$(MACH64)_CCC=          $(BUILD_CCC)
$(MACH64)_CPP=          $(BUILD_CPP)
$(MACH64)_LD=           $(BUILD_LD)
$(MACH64)_LINT=         $(BUILD_LINT)

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

```

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

NATIVECC=               $($(NATIVE_MACH)_CC)
NATIVECCC=              $($(NATIVE_MACH)_CCC)
NATIVECPP=              $($(NATIVE_MACH)_CPP)
NATIVEAS=               $($(NATIVE_MACH)_AS)
NATIVELD=               $($(NATIVE_MACH)_LD)
NATIVELINT=             $($(NATIVE_MACH)_LINT)

#
# Makefile.master.64 overrides these settings
#
CC=                     $(NATIVECC)
CCC=                    $(NATIVECCC)
CPP=                    $(NATIVECPP)
AS=                     $(NATIVEAS)
LD=                     $(NATIVELD)
LINT=                   $(NATIVELINT)

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

BDIRECT=        -Bdirect
BDYNAMIC=       -Bdynamic
BLOCAL=         -Blocal
BNODIRECT=      -Bnodirect
BREDUCE=        -Breduce
BSTATIC=        -Bstatic

ZDEFS=          -zdefs
ZDIRECT=        -zdirect
ZIGNORE=        -zignore
ZINITFIRST=     -zinitfirst
ZINTERPOSE=     -zinterpose
ZLAZYLOAD=      -zlazyload
ZLOADFLTR=      -zloadfltr
ZMULDEFS=       -zmuldefs
ZNODEFAULTLIB=  -znodefaultlib
ZNODEFS=        -znodefs
ZNODELETE=      -znodelete
ZNODLOPEN=      -znodlopen
ZNODUMP=        -znodump
ZNOLAZYLOAD=    -znolazyload
ZNOLDYNSYM=     -znoldynsym
ZNORELOC=       -znoreloc
ZNOVERSION=     -znoversion
ZRECORD=        -zrecord
ZREDLOCSYM=     -zredlocsym
ZTEXT=          -ztext
ZVERBOSE=       -zverbose

GSHARED=        -G
CCMT=           -mt

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

sparc_C_PICFLAGS =      -fpic
sparcv9_C_PICFLAGS =    -fpic
i386_C_PICFLAGS =       -fpic
amd64_C_PICFLAGS =      -fpic
C_PICFLAGS =            $($(MACH)_C_PICFLAGS)
C_PICFLAGS64 =          $($(MACH64)_C_PICFLAGS)
```

```
sparc_C_BIGPICFLAGS =   -fPIC
sparcv9_C_BIGPICFLAGS = -fPIC
i386_C_BIGPICFLAGS =    -fPIC
amd64_C_BIGPICFLAGS =   -fPIC
C_BIGPICFLAGS =         $($(MACH)_C_BIGPICFLAGS)
C_BIGPICFLAGS64 =       $($(MACH64)_C_BIGPICFLAGS)

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

AS_PICFLAGS=            -K pic
AS_BIGPICFLAGS=         -K PIC

#
# Default label for CTF sections
#
CTFCVTFLAGS=            -i -L VERSION

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

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

# Rules (normally from make.rules) and macros which are used for post
# processing files. Normally, these do stripping of the comment section
# automatically.
#    RELEASE_CM:        Should be editted to reflect the release.
#    POST_PROCESS_O:    Post-processing for '.o' files.
#    POST_PROCESS_A:    Post-processing for '.a' files (currently null).
#    POST_PROCESS_SO:   Post-processing for '.so' files.
#    POST_PROCESS:      Post-processing for executable files (no suffix).
# Note that these macros are not completely generalized as they are to be
# used with the file name to be processed following.
#
# It is left as an exercise to Release Engineering to embellish the generation
# of the release comment string.
#
#       If this is a standard development build:
#               compress the comment section (mcs -c)
#               add the standard comment (mcs -a $(RELEASE_CM))
#               add the development specific comment (mcs -a $(DEV_CM))
#
#       If this is an installation build:
#               delete the comment section (mcs -d)
#               add the standard comment (mcs -a $(RELEASE_CM))
#               add the development specific comment (mcs -a $(DEV_CM))
#
#       If this is an release build:
#               delete the comment section (mcs -d)
#               add the standard comment (mcs -a $(RELEASE_CM))
#
# The following list of macros are used in the definition of RELEASE_CM
# which is used to label all binaries in the build:
#
#       RELEASE         Specific release of the build, eg: 5.2
```

```
#       RELEASE_MAJOR   Major version number part of $(RELEASE)
#       RELEASE_MINOR   Minor version number part of $(RELEASE)
#       VERSION         Version of the build (alpha, beta, Generic)
#       PATCHID         If this is a patch this value should contain
#                       the patchid value (eg: "Generic 100832-01"), otherwise
#                       it will be set to $(VERSION)
#       RELEASE_DATE    Date of the Release Build
#       PATCH_DATE      Date the patch was created, if this is blank it
#                       will default to the RELEASE_DATE
#
RELEASE_MAJOR=  5
RELEASE_MINOR=  11
RELEASE=        $(RELEASE_MAJOR).$(RELEASE_MINOR)
VERSION=        SunOS Development
PATCHID=        $(VERSION)
RELEASE_DATE=   release date not set
PATCH_DATE=     $(RELEASE_DATE)
RELEASE_CM=     "@($(POUND_SIGN))SunOS $(RELEASE) $(PATCHID) $(PATCH_DATE)"
DEV_CM=         "@($(POUND_SIGN))SunOS Internal Development: non-nightly build"

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

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

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

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

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

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

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

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

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

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

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

```

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

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

.py:
	$(RM) $@; $(SED) -e "1s:^\#!@PYTHON@:\#!$(PYTHON):" < $< > $@; $(CHMOD)

.py.pyc:
	$(RM) $@
	$(PYTHON) -mpy_compile $<
	@[ $(<)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
```

****
   21217 Tue Oct 30 20:22:48 2018
new/usr/src/uts/Makefile.uts
9939 Need to stop GCC reordering functions
****

```
#
# 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>
#

#
#       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)
AS_i386_64      = $(amd64_AS)
AS_amd64_64     = $(amd64_AS)

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

```
LD_sparc_32     = $(sparc_LD)
LD_sparc_64     = $(sparcv9_LD)

LD_i386_32      = $(i386_LD)
LD_i386_64      = $(amd64_LD)
LD_amd64_64     = $(amd64_LD)

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

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

LINT_i386_32    = $(i386_LINT)
LINT_i386_64    = $(amd64_LINT)
LINT_amd64_64   = $(amd64_LINT)

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

MODEL_32        = ilp32
MODEL_64        = lp64
MODEL           = $(MODEL_$(CLASS))

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

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

LTAIL =

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

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

LINTFLAGS_sparc_32      = $(LINTCCMODE) -nsxmuF -errtags=yes
LINTFLAGS_sparc_64      = $(LINTFLAGS_sparc_32) -m64
LINTFLAGS_i386_32       = $(LINTCCMODE) -nsxmuF -errtags=yes
LINTFLAGS_i386_64       = $(LINTFLAGS_i386_32) -m64

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

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

#
#       Build the compile/assemble lines:
#
EXTRA_OPTIONS           =
AS_DEFS                 = -D_ASM -D__STDC__=0

ALWAYS_DEFS_32          = -D_KERNEL -D_SYSCALL32 -D_DDI_STRICT
ALWAYS_DEFS_64          = -D_KERNEL -D_SYSCALL32 -D_SYSCALL32_IMPL -D_ELF64 \
                        -D_DDI_STRICT
#
# XX64  This should be defined by the compiler!
```

```
#
ALWAYS_DEFS_64          += -Dsun -D__sun -D__SVR4
ALWAYS_DEFS             = $(ALWAYS_DEFS_$(CLASS))

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

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

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

CERRWARN += -_gcc=-Wno-missing-braces
CERRWARN += -_gcc=-Wno-sign-compare
CERRWARN += -_gcc=-Wno-unknown-pragmas
CERRWARN += -_gcc=-Wno-unused-parameter
CERRWARN += -_gcc=-Wno-missing-field-initializers

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

CSTD = $(CSTD_GNU99)

CFLAGS_uts              =
CFLAGS_uts              += $(STAND_FLAGS_$(CLASS))
CFLAGS_uts              += $(CCVERBOSE)
CFLAGS_uts              += $(ILDOFF)
CFLAGS_uts              += $(XAOPT)
CFLAGS_uts              += $(CTF_FLAGS_$(CLASS))
CFLAGS_uts              += $(CERRWARN)
CFLAGS_uts              += $(CCNOAUTOINLINE)
CFLAGS_uts              += $(CCNOREORDER)
#endif /* ! codereview */
CFLAGS_uts              += $(CGLOBALSTATIC)
CFLAGS_uts              += $(EXTRA_CFLAGS)
CFLAGS_uts              += $(CSOURCEDEBUGFLAGS)
CFLAGS_uts              += $(CUSERFLAGS)

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

#
#       Expanded dependencies
#
DEF_DEPS        = $(DEF_BUILDS:%=def.%)
ALL_DEPS        = $(ALL_BUILDS:%=all.%)
CLEAN_DEPS      = $(ALL_BUILDS:%=clean.%)
CLOBBER_DEPS    = $(ALL_BUILDS:%=clobber.%)
LINT_DEPS       = $(DEF_BUILDS:%=lint.%)
```

```
MODLINTLIB_DEPS = $(DEF_BUILDS:%=modlintlib.%)
MODLIST_DEPS    = $(DEF_BUILDS:%=modlist.%)
CLEAN_LINT_DEPS = $(ALL_BUILDS:%=clean.lint.%)
INSTALL_DEPS    = $(DEF_BUILDS:%=install.%)
SYM_DEPS        = $(SYM_BUILDS:%=symcheck.%)
SISCHECK_DEPS   = $(DEF_BUILDS:%=sischeck.%)
SISCLEAN_DEPS   = $(ALL_BUILDS:%=sisclean.%)

#
#       Default module name
#
BINARY          = $(OBJS_DIR)/$(MODULE)

#
#       Default cleanup definitions
#
CLEANLINTFILES  = $(LINTS) $(MOD_LINT_LIB)
CLEANFILES      = $(OBJECTS) $(CLEANLINTFILES)
CLOBBERFILES    = $(BINARY) $(CLEANFILES)

#
#       Installation constants:
#
#               FILEMODE is the mode given to the kernel modules
#               CFILEMODE is the mode given to the '.conf' files
#
FILEMODE         = 755
DIRMODE          = 755
CFILEMODE        = 644

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


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

#
# The CTF merge of child kernel modules is performed against one of the genunix
# modules.  For Intel builds, all modules will be used with a single genunix:
# the one built in intel/genunix.  For SPARC builds, a given
# module may be
# used with one of a number of genunix files, depending on what platform the
# module is deployed on.  We merge against the sun4u genunix to optimize for
# the common case.  We also merge against the ip driver since networking is
# typically loaded and types defined therein are shared between many modules.
#
CTFMERGE_GUDIR_sparc    = sun4u
```

```
CTFMERGE_GUDIR_i386     = intel
CTFMERGE_GUDIR          = $(CTFMERGE_GUDIR_$(MACH))

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

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

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

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

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

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

#
# SONAME defaults for common fake shared libraries.
#
$(LIBGEN)               := SONAME = $(MODULE)
$(PLATLIB)              := SONAME = misc/platmod
$(CPULIB)               := SONAME = 'cpu/$$CPU'
$(DTRACESTUBS)          := SONAME = dtracestubs

#
#       Installation directories
#

#
#       For now, 64b modules install into a subdirectory
#       of their 32b brethren.
#
SUBDIR64_sparc          = sparcv9
SUBDIR64_i386           = amd64
SUBDIR64                = $(SUBDIR64_$(MACH))

ROOT_MOD_DIR            = $(ROOT)/kernel
```

```
ROOT_KERN_DIR_32        = $(ROOT_MOD_DIR)
ROOT_BRAND_DIR_32       = $(ROOT_MOD_DIR)/brand
ROOT_DRV_DIR_32         = $(ROOT_MOD_DIR)/drv
ROOT_DTRACE_DIR_32      = $(ROOT_MOD_DIR)/dtrace
ROOT_EXEC_DIR_32        = $(ROOT_MOD_DIR)/exec
ROOT_FS_DIR_32          = $(ROOT_MOD_DIR)/fs
ROOT_SCHED_DIR_32       = $(ROOT_MOD_DIR)/sched
ROOT_SOCK_DIR_32        = $(ROOT_MOD_DIR)/socketmod
ROOT_STRMOD_DIR_32      = $(ROOT_MOD_DIR)/strmod
ROOT_IPP_DIR_32         = $(ROOT_MOD_DIR)/ipp
ROOT_SYS_DIR_32         = $(ROOT_MOD_DIR)/sys
ROOT_MISC_DIR_32        = $(ROOT_MOD_DIR)/misc
ROOT_KGSS_DIR_32        = $(ROOT_MOD_DIR)/misc/kgss
ROOT_SCSI_VHCI_DIR_32   = $(ROOT_MOD_DIR)/misc/scsi_vhci
ROOT_PMCS_FW_DIR_32     = $(ROOT_MOD_DIR)/misc/pmcs
ROOT_QLC_FW_DIR_32      = $(ROOT_MOD_DIR)/misc/qlc
ROOT_EMLXS_FW_DIR_32    = $(ROOT_MOD_DIR)/misc/emlxs
ROOT_NLMISC_DIR_32      = $(ROOT_MOD_DIR)/misc
ROOT_MACH_DIR_32        = $(ROOT_MOD_DIR)/mach
ROOT_CPU_DIR_32         = $(ROOT_MOD_DIR)/cpu
ROOT_TOD_DIR_32         = $(ROOT_MOD_DIR)/tod
ROOT_FONT_DIR_32        = $(ROOT_MOD_DIR)/fonts
ROOT_DACF_DIR_32        = $(ROOT_MOD_DIR)/dacf
ROOT_CRYPTO_DIR_32      = $(ROOT_MOD_DIR)/crypto
ROOT_MAC_DIR_32         = $(ROOT_MOD_DIR)/mac
ROOT_KICONV_DIR_32      = $(ROOT_MOD_DIR)/kiconv

ROOT_KERN_DIR_64        = $(ROOT_MOD_DIR)/$(SUBDIR64)
ROOT_BRAND_DIR_64       = $(ROOT_MOD_DIR)/brand/$(SUBDIR64)
ROOT_DRV_DIR_64         = $(ROOT_MOD_DIR)/drv/$(SUBDIR64)
ROOT_DTRACE_DIR_64      = $(ROOT_MOD_DIR)/dtrace/$(SUBDIR64)
ROOT_EXEC_DIR_64        = $(ROOT_MOD_DIR)/exec/$(SUBDIR64)
ROOT_FS_DIR_64          = $(ROOT_MOD_DIR)/fs/$(SUBDIR64)
ROOT_SCHED_DIR_64       = $(ROOT_MOD_DIR)/sched/$(SUBDIR64)
ROOT_SOCK_DIR_64        = $(ROOT_MOD_DIR)/socketmod/$(SUBDIR64)
ROOT_STRMOD_DIR_64      = $(ROOT_MOD_DIR)/strmod/$(SUBDIR64)
ROOT_IPP_DIR_64         = $(ROOT_MOD_DIR)/ipp/$(SUBDIR64)
ROOT_SYS_DIR_64         = $(ROOT_MOD_DIR)/sys/$(SUBDIR64)
ROOT_MISC_DIR_64        = $(ROOT_MOD_DIR)/misc/$(SUBDIR64)
ROOT_KGSS_DIR_64        = $(ROOT_MOD_DIR)/misc/kgss/$(SUBDIR64)
ROOT_SCSI_VHCI_DIR_64   = $(ROOT_MOD_DIR)/misc/scsi_vhci/$(SUBDIR64)
ROOT_PMCS_FW_DIR_64     = $(ROOT_MOD_DIR)/misc/pmcs/$(SUBDIR64)
ROOT_QLC_FW_DIR_64      = $(ROOT_MOD_DIR)/misc/qlc/$(SUBDIR64)
ROOT_EMLXS_FW_DIR_64    = $(ROOT_MOD_DIR)/misc/emlxs/$(SUBDIR64)
ROOT_NLMISC_DIR_64      = $(ROOT_MOD_DIR)/misc/$(SUBDIR64)
ROOT_MACH_DIR_64        = $(ROOT_MOD_DIR)/mach/$(SUBDIR64)
ROOT_CPU_DIR_64         = $(ROOT_MOD_DIR)/cpu/$(SUBDIR64)
ROOT_TOD_DIR_64         = $(ROOT_MOD_DIR)/tod/$(SUBDIR64)
ROOT_FONT_DIR_64        = $(ROOT_MOD_DIR)/fonts/$(SUBDIR64)
ROOT_DACF_DIR_64        = $(ROOT_MOD_DIR)/dacf/$(SUBDIR64)
ROOT_CRYPTO_DIR_64      = $(ROOT_MOD_DIR)/crypto/$(SUBDIR64)
ROOT_MAC_DIR_64         = $(ROOT_MOD_DIR)/mac/$(SUBDIR64)
ROOT_KICONV_DIR_64      = $(ROOT_MOD_DIR)/kiconv/$(SUBDIR64)

ROOT_KERN_DIR           = $(ROOT_KERN_DIR_$(CLASS))
ROOT_BRAND_DIR          = $(ROOT_BRAND_DIR_$(CLASS))
ROOT_DRV_DIR            = $(ROOT_DRV_DIR_$(CLASS))
ROOT_DTRACE_DIR         = $(ROOT_DTRACE_DIR_$(CLASS))
ROOT_EXEC_DIR           = $(ROOT_EXEC_DIR_$(CLASS))
ROOT_FS_DIR             = $(ROOT_FS_DIR_$(CLASS))
ROOT_SCHED_DIR          = $(ROOT_SCHED_DIR_$(CLASS))
ROOT_SOCK_DIR           = $(ROOT_SOCK_DIR_$(CLASS))
ROOT_STRMOD_DIR         = $(ROOT_STRMOD_DIR_$(CLASS))
ROOT_IPP_DIR            = $(ROOT_IPP_DIR_$(CLASS))
ROOT_SYS_DIR            = $(ROOT_SYS_DIR_$(CLASS))
```

```
ROOT_MISC_DIR           = $(ROOT_MISC_DIR_$(CLASS))
ROOT_KGSS_DIR           = $(ROOT_KGSS_DIR_$(CLASS))
ROOT_SCSI_VHCI_DIR      = $(ROOT_SCSI_VHCI_DIR_$(CLASS))
ROOT_PMCS_FW_DIR        = $(ROOT_PMCS_FW_DIR_$(CLASS))
ROOT_QLC_FW_DIR         = $(ROOT_QLC_FW_DIR_$(CLASS))
ROOT_EMLXS_FW_DIR       = $(ROOT_EMLXS_FW_DIR_$(CLASS))
ROOT_NLMISC_DIR         = $(ROOT_NLMISC_DIR_$(CLASS))
ROOT_MACH_DIR           = $(ROOT_MACH_DIR_$(CLASS))
ROOT_CPU_DIR            = $(ROOT_CPU_DIR_$(CLASS))
ROOT_TOD_DIR            = $(ROOT_TOD_DIR_$(CLASS))
ROOT_FONT_DIR           = $(ROOT_FONT_DIR_$(CLASS))
ROOT_DACF_DIR           = $(ROOT_DACF_DIR_$(CLASS))
ROOT_CRYPTO_DIR         = $(ROOT_CRYPTO_DIR_$(CLASS))
ROOT_MAC_DIR            = $(ROOT_MAC_DIR_$(CLASS))
ROOT_KICONV_DIR         = $(ROOT_KICONV_DIR_$(CLASS))
ROOT_FIRMWARE_DIR       = $(ROOT_MOD_DIR)/firmware

ROOT_MOD_DIRS_32        = $(ROOT_BRAND_DIR_32) $(ROOT_DRV_DIR_32)
ROOT_MOD_DIRS_32        = $(ROOT_BRAND_DIR_32) $(ROOT_DRV_DIR_32)
ROOT_MOD_DIRS_32        += $(ROOT_EXEC_DIR_32) $(ROOT_DTRACE_DIR_32)
ROOT_MOD_DIRS_32        += $(ROOT_FS_DIR_32) $(ROOT_SCHED_DIR_32)
ROOT_MOD_DIRS_32        += $(ROOT_STRMOD_DIR_32) $(ROOT_SYS_DIR_32)
ROOT_MOD_DIRS_32        += $(ROOT_IPP_DIR_32) $(ROOT_SOCK_DIR_32)
ROOT_MOD_DIRS_32        += $(ROOT_MISC_DIR_32) $(ROOT_MACH_DIR_32)
ROOT_MOD_DIRS_32        += $(ROOT_KGSS_DIR_32)
ROOT_MOD_DIRS_32        += $(ROOT_SCSI_VHCI_DIR_32)
ROOT_MOD_DIRS_32        += $(ROOT_PMCS_FW_DIR_32)
ROOT_MOD_DIRS_32        += $(ROOT_QLC_FW_DIR_32)
ROOT_MOD_DIRS_32        += $(ROOT_EMLXS_FW_DIR_32)
ROOT_MOD_DIRS_32        += $(ROOT_CPU_DIR_32) $(ROOT_FONT_DIR_32)
ROOT_MOD_DIRS_32        += $(ROOT_TOD_DIR_32) $(ROOT_DACF_DIR_32)
ROOT_MOD_DIRS_32        += $(ROOT_CRYPTO_DIR_32) $(ROOT_MAC_DIR_32)
ROOT_MOD_DIRS_32        += $(ROOT_KICONV_DIR_32)
ROOT_MOD_DIRS_32        += $(ROOT_FIRMWARE_DIR)

USR_MOD_DIR             = $(ROOT)/usr/kernel

USR_DRV_DIR_32          = $(USR_MOD_DIR)/drv
USR_EXEC_DIR_32         = $(USR_MOD_DIR)/exec
USR_FS_DIR_32           = $(USR_MOD_DIR)/fs
USR_SCHED_DIR_32        = $(USR_MOD_DIR)/sched
USR_SOCK_DIR_32         = $(USR_MOD_DIR)/socketmod
USR_STRMOD_DIR_32       = $(USR_MOD_DIR)/strmod
USR_SYS_DIR_32          = $(USR_MOD_DIR)/sys
USR_MISC_DIR_32         = $(USR_MOD_DIR)/misc
USR_DACF_DIR_32         = $(USR_MOD_DIR)/dacf
USR_PCBE_DIR_32         = $(USR_MOD_DIR)/pcbe
USR_DTRACE_DIR_32       = $(USR_MOD_DIR)/dtrace
USR_BRAND_DIR_32        = $(USR_MOD_DIR)/brand

USR_DRV_DIR_64          = $(USR_MOD_DIR)/drv/$(SUBDIR64)
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
```

****
3361 Tue Oct 30 20:22:48 2018
new/usr/src/uts/intel/asm/cpu.h
9927 refetch_read_once() would like a p please bob
****
______***unchanged_portion_omitted_***

```
extern __GNU_INLINE void
prefetch_read_once(void *addr)
refetch_read_once(void *addr)
{
#if defined(__amd64)
        __asm__(
            "prefetchnta (%0);"
            "prefetchnta 32(%0);"
            : /* no output */
            : "r" (addr));
#endif  /* __amd64 */
}
```
______***unchanged_portion_omitted_***