```
**********************************************************
   34785 Tue Nov  4 17:02:07 2014
new/usr/src/Makefile.master
4457 we apparently change .comment of almost every userland object
**********************************************************
#
# 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>
#

#
# 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)

# 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

JAVA_ROOT=      /usr/java

SFW_ROOT=       /usr/sfw
SFWINCDIR=      $(SFW_ROOT)/include
SFWLIBDIR=      $(SFW_ROOT)/lib
SFWLIBDIR64=    $(SFW_ROOT)/lib/$(MACH64)

GCC_ROOT=       /opt/gcc/4.4.4
GCCLIBDIR=      $(GCC_ROOT)/lib
GCCLIBDIR64=    $(GCC_ROOT)/lib/$(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
ECHO=           echo
INS=            install
TRUE=           true
```

```
SYMLINK=        /usr/bin/ln -s
LN=             /usr/bin/ln
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
NAWK=           /usr/bin/nawk
CP=             /usr/bin/cp -f
MCS=            /usr/ccs/bin/mcs
CAT=            /usr/bin/cat
ELFDUMP=        /usr/ccs/bin/elfdump
M4=             /usr/ccs/bin/m4
STRIP=          /usr/ccs/bin/strip
LEX=            /usr/ccs/bin/lex
FLEX=           $(SFW_ROOT)/bin/flex
YACC=           /usr/ccs/bin/yacc
CPP=            /usr/lib/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
CTFCVTPTBL=     $(ONBLD_TOOLS)/bin/ctfcvtptbl
CTFFINDMOD=     $(ONBLD_TOOLS)/bin/ctffindmod
XREF=           $(ONBLD_TOOLS)/bin/xref
FIND=           /usr/bin/find
PERL=           /usr/bin/perl
PERL_VERSION=   5.10.0
PERL_PKGVERS=   -510
PYTHON_26=      /usr/bin/python2.6
PYTHON=         $(PYTHON_26)
SORT=           /usr/bin/sort
TOUCH=          /usr/bin/touch
WC=             /usr/bin/wc
XARGS=          /usr/bin/xargs
ELFEDIT=        /usr/bin/elfedit
ELFSIGN=        /usr/bin/elfsign
DTRACE=         /usr/sbin/dtrace -xnolibs
UNIQ=           /usr/bin/uniq
TAR=            /usr/bin/tar
ASTBINDIR=      /usr/ast/bin
MSGCC=          $(ASTBINDIR)/msgcc

FILEMODE=       644
DIRMODE=        755

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

# 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=     $(INS.file); $(TOUCH) -r $< $@

# 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

# 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=
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
#
XDEPEND=                -xdepend
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

# C99 mode
C99_ENABLE=     -xc99=%all
C99_DISABLE=    -xc99=%none
C99MODE=        $(C99_DISABLE)
C99LMODE=       $(C99MODE:-xc99%=-Xc99%)

# In most places, assignments to these macros should be appended with +=
# (CPPFLAGS.master 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 $(C99MODE) $(CNOGLOBAL) $(CDWARFSTR)
CTF_FLAGS_i386  = -g $(C99MODE) $(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 = -_noecho \
        $(CALLSYMS) \
        $(CDWARFSTR)

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

OFFSETS_CREATE64 = $(GENOFFSETS) -s $(CTFSTABS) -r $(CTFCONVERT) \
        $(CC) $(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) $(C99MODE) $(CCUNBOUND) $(IROPTFLAG) \
                $(CGLOBALSTATIC) $(CCNOAUTOINLINE) $(CSOURCEDEBUGFLAGS) \
                $(CUSERFLAGS)
CFLAGS64=       $(COPTFLAG64) $($(MACH64)_CFLAGS) $(SPACEFLAG64) $(CCMODE64) \
                $(ILDOFF) $(CERRWARN) $(C99MODE) $(CCUNBOUND) $(IROPTFLAG64) \
                $(CGLOBALSTATIC) $(CCNOAUTOINLINE) $(CSOURCEDEBUGFLAGS) \
                $(CUSERFLAGS64)
#
# Flags that are used to build parts of the code that are subsequently
# run on the build machine (also known as the NATIVE_BUILD).
#
NATIVE_CFLAGS=  $(COPTFLAG) $($(NATIVE_MACH)_CFLAGS) $(CCMODE) \
                $(ILDOFF) $(CERRWARN) $(C99MODE) $($(NATIVE_MACH)_CCUNBOUND) \
                $(IROPTFLAG) $(CGLOBALSTATIC) $(CCNOAUTOINLINE) \
                $(CSOURCEDEBUGFLAGS) $(CUSERFLAGS)

DTEXTDOM=-DTEXT_DOMAIN=\"$(TEXT_DOMAIN)\"       # For messaging.
DTS_ERRNO=-D_TS_ERRNO
CPPFLAGS.master=$(DTEXTDOM) $(DTS_ERRNO) \
        $(ENVCPPFLAGS1) $(ENVCPPFLAGS2) $(ENVCPPFLAGS3) $(ENVCPPFLAGS4) \
        $(ADJUNCT_PROTO:%=-I%/usr/include)
CPPFLAGS.native=$(ENVCPPFLAGS1) $(ENVCPPFLAGS2) $(ENVCPPFLAGS3) \
                $(ENVCPPFLAGS4) -I$(NATIVE_ADJUNCT)/include
CPPFLAGS=       $(CPPFLAGS.master)
AS_CPPFLAGS=    $(CPPFLAGS.master)
JAVAFLAGS=      -deprecation

#
# 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=               $(SFW_ROOT)

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

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

sparcv9_CC=             $(ONBLD_TOOLS)/bin/$(MACH)/cw -_cc
$(__GNUC64)sparcv9_CC=  $(ONBLD_TOOLS)/bin/$(MACH)/cw -_gcc
sparcv9_CCC=            $(ONBLD_TOOLS)/bin/$(MACH)/cw -_CC
$(__GNUC64)sparcv9_CCC= $(ONBLD_TOOLS)/bin/$(MACH)/cw -_g++
sparcv9_CPP=            /usr/ccs/lib/cpp
```

```
sparcv9_AS=             /usr/ccs/bin/as -xregsym=no
sparcv9_LD=             /usr/ccs/bin/ld
sparcv9_LINT=           $(SPRO_ROOT)/sunstudio12.1/bin/lint

i386_CC=                $(ONBLD_TOOLS)/bin/$(MACH)/cw -_cc
$(__GNUC)i386_CC=       $(ONBLD_TOOLS)/bin/$(MACH)/cw -_gcc
i386_CCC=               $(ONBLD_TOOLS)/bin/$(MACH)/cw -_CC
$(__GNUC)i386_CCC=      $(ONBLD_TOOLS)/bin/$(MACH)/cw -_g++
i386_CPP=               /usr/ccs/lib/cpp
i386_AS=                /usr/ccs/bin/as
$(__GNUC)i386_AS=       $(ONBLD_TOOLS)/bin/$(MACH)/aw
i386_LD=                /usr/ccs/bin/ld
i386_LINT=              $(SPRO_ROOT)/sunstudio12.1/bin/lint

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

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)

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

# 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 =      -K pic
sparcv9_C_PICFLAGS =    -K pic
i386_C_PICFLAGS =       -K pic
amd64_C_PICFLAGS =      -K pic
C_PICFLAGS =            $($(MACH)_C_PICFLAGS)
C_PICFLAGS64 =          $($(MACH64)_C_PICFLAGS)

sparc_C_BIGPICFLAGS =   -K PIC
sparcv9_C_BIGPICFLAGS = -K PIC
i386_C_BIGPICFLAGS =    -K PIC
amd64_C_BIGPICFLAGS =   -K PIC
C_BIGPICFLAGS =         $($(MACH)_C_BIGPICFLAGS)
C_BIGPICFLAGS64 =       $($(MACH64)_C_BIGPICFLAGS)

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

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

#
# Default label for CTF sections
#
CTFCVTFLAGS=            -i -L VERSION
$(SRCDBGBLD)CTFCVTFLAGS         += -g

#
# 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=
$(SRCDBGBLD)CTFMRGFLAGS         += -g


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

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

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

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

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

POST_PROCESS_O=
POST_PROCESS_O=         $(PROCESS_COMMENT) $@
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) $@; $(CAT) $< > $@; $(CHMOD) +x $@

.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)

.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
```