```
**********************************************************
    8641 Sat Apr 25 16:51:17 2015
new/usr/src/lib/Makefile.lib
5465 CFLAGS missing from BUILD.SO in lib/Makefile.lib
**********************************************************
```

```
#
# 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 2015 Gary Mills
# Copyright (c) 1989, 2010, Oracle and/or its affiliates. All rights reserved.
#
#
# Definitions common to libraries.
#
# include global definitions; SRC should be defined in the shell.
# SRC is needed until RFE 1026993 is implemented.

include         $(SRC)/Makefile.master

LORDER=         lorder
TSORT=          tsort
AWK=            awk

#
# By default, we define the source directory for libraries to be
# one level up from the ISA-specific directory, where the code is
# actually built.  Many libraries define a 'common' directory to
# contain the source.  These libraries must redefine SRCDIR as:
#       SRCDIR = ../common
# Other variations are possible (../port, ../src, etc).
#
SRCDIR =        ..

#
# We define MAPFILES here for the benefit of most libraries, those that
# follow the convention of having source files and other common files
# in the $(SRCDIR) directory.  Libraries that do not follow this
# convention must define MAPFILES, or MAPFILEDIR for themselves.
# Libraries that do follow this convention but that need supplemental
# ISA-specific mapfiles can augment MAPFILES like this:
#       MAPFILES += mapfile-vers
#
MAPFILEDIR =    $(SRCDIR)
MAPFILES =      $(MAPFILEDIR)/mapfile-vers

#
# If HDRDIR is left unset, then it's possible for the $(ROOTHDRDIR)/%
# install rule in lib/Makefile.targ to generate false matches if there
# are any common directory names between / and /usr/include ('xfn' is
```



```
# one common example).  To prevent this, we set HDRDIR to a directory
# name that will almost surely not exist on the build machine.
#
HDRDIR=         /__nonexistent_directory__

#
# We don't build archive (*.a) libraries by default anymore.
# If a component of the build needs to build an archive library
# for its own internal purposes, it can define LIBS for itself
# after including Makefile.lib, like this:
#       LIBS = $(LIBRARY)
# or:
#       LIBS = $(LIBRARYCCC)
# Archive libraries must not be installed in the proto area.
#
LIBS=
MACHLIBS=       $(LIBS:%=$(MACH)/%)
MACHLIBS64=     $(LIBS:%=$(MACH64)/%)
DYNLIB=         $(LIBRARY:.a=.so$(VERS))
DYNLIBPSR=      $(LIBRARY:.a=_psr.so$(VERS))
DYNLIBCCC=      $(LIBRARYCCC:.a=.so$(VERS))
LIBLINKS=       $(LIBRARY:.a=.so)
LIBLINKSCCC=    $(LIBRARYCCC:.a=.so)
LIBNAME=        $(LIBRARY:lib%.a=%)
LIBLINKPATH=
LIBNULL=        null.a
ROOTHDRDIR=     $(ROOT)/usr/include
ROOTLIBDIR=     $(ROOT)/usr/lib
ROOTLIBDIR64=   $(ROOT)/usr/lib/$(MACH64)
ROOTFS_LIBDIR=  $(ROOT)/lib
ROOTFS_LIBDIR64=        $(ROOT)/lib/$(MACH64)
ROOTLINTDIR=    $(ROOTLIBDIR)
ROOTFS_LINTDIR= $(ROOTFS_LIBDIR)
ROOTFS_LINTDIR64=       $(ROOTFS_LIBDIR64)
ROOTHDRS=       $(HDRS:%=$(ROOTHDRDIR)/%)
HDRSRCS=        $(HDRS:%=$(HDRDIR)/%)
CHECKHDRS=      $(HDRSRCS:%.h=%.check)
ROOTLIBS=       $(LIBS:%=$(ROOTLIBDIR)/%)
ROOTLIBS64=     $(LIBS:%=$(ROOTLIBDIR64)/%)
ROOTFS_LIBS=    $(DYNLIB:%=$(ROOTFS_LIBDIR)/%)
ROOTFS_LIBS64=  $(DYNLIB:%=$(ROOTFS_LIBDIR64)/%)
ROOTLINKS=      $(ROOTLIBDIR)/$(LIBLINKS)
ROOTLINKS64=    $(ROOTLIBDIR64)/$(LIBLINKS)
ROOTFS_LINKS=   $(ROOTFS_LIBDIR)/$(LIBLINKS)
ROOTFS_LINKS64= $(ROOTFS_LIBDIR64)/$(LIBLINKS)
ROOTLINKSCCC=   $(ROOTLIBDIR)/$(LIBLINKSCCC)
ROOTLINKSCCC64= $(ROOTLIBDIR64)/$(LIBLINKSCCC)
ROOTFS_LINKSCCC=        $(ROOTFS_LIBDIR)/$(LIBLINKSCCC)
ROOTFS_LINKSCCC64=      $(ROOTFS_LIBDIR64)/$(LIBLINKSCCC)
ROOTLINT=       $(LINTSRC:%=$(ROOTLINTDIR)/%)
ROOTFS_LINT=    $(LINTSRC:%=$(ROOTFS_LINTDIR)/%)
ROOTFS_LINT64=  $(LINTSRC:%=$(ROOTFS_LINTDIR64)/%)
ROOTMAN3=     $(ROOT)/usr/share/man/man3
ROOTMAN3FILES=        $(MAN3FILES:%=$(ROOTMAN3)/%)
$(ROOTMAN3FILES) := FILEMODE= 444

# Demo rules
DEMOFILES=
DEMOFILESRCDIR=         common
ROOTDEMODIRBASE=        __nonexistent_directory__
ROOTDEMODIRS=
ROOTDEMOFILES=  $(DEMOFILES:%=$(ROOTDEMODIRBASE)/%)
$(ROOTDEMODIRS) :=      DIRMODE =       755

LINTLIB=        llib-l$(LIBNAME).ln
LINTFLAGS=      -uaxm
```

```
LINTFLAGS64=    -uaxm -m64
LINTSRC=        $(LINTLIB:%.ln=%)
LINTOUT=        lint.out
ARFLAGS=        r
SONAME=         $(DYNLIB)
# For most libraries, we should be able to resolve all symbols at link time,
# either within the library or as dependencies, all text should be pure, and
# combining relocations into one relocation table reduces startup costs.
# All options are tunable to allow overload/omission from lower makefiles.


HSONAME=        -h$(SONAME)
DYNFLAGS=       $(HSONAME) $(ZTEXT) $(ZDEFS) $(BDIRECT) \
                $(MAPFILES:%=-M%) $(MAPFILE.PGA:%=-M%) $(MAPFILE.NED:%=-M%)

LDLIBS=         $(LDLIBS.lib)

OBJS=           $(OBJECTS:%=objs/%)
PICS=           $(OBJECTS:%=pics/%)

# Declare that all library .o's can all be made in parallel.
# The DUMMY target is for those instances where OBJS and PICS
# are empty (to avoid an unconditional .PARALLEL declaration).
.PARALLEL:      $(OBJS) $(PICS) DUMMY

# default value for "portable" source
SRCS=           $(OBJECTS:%.o=$(SRCDIR)/%.c)

# default build of an archive and a shared object,
# overridden locally when extra processing is needed
BUILD.AR=       $(AR) $(ARFLAGS) $@ $(AROBJS)
BUILD.SO=       $(CC) $(CFLAGS) -o $@ $(GSHARED) $(DYNFLAGS) \
                $(PICS) $(EXTPICS) $(LDLIBS)
BUILDCCC.SO=    $(CCC) $(CCFLAGS) -o $@ $(GSHARED) $(DYNFLAGS) \
                $(PICS) $(EXTPICS) $(LDLIBS)
BUILD.SO=       $(CC) -o $@ $(GSHARED) $(DYNFLAGS) $(PICS) $(EXTPICS) $(LDLIBS)
BUILDCCC.SO=    $(CCC) -o $@ $(GSHARED) $(DYNFLAGS) $(PICS) $(EXTPICS) $(LDLIBS)

# default dynamic library symlink
INS.liblink=    -$(RM) $@; $(SYMLINK) $(LIBLINKPATH)$(LIBLINKS)$(VERS) $@
INS.liblinkccc= -$(RM) $@; $(SYMLINK) $(LIBLINKPATH)$(LIBLINKSCCC)$(VERS) $@

# default 64-bit dynamic library symlink
INS.liblink64=  -$(RM) $@; $(SYMLINK) $(LIBLINKPATH)$(LIBLINKS)$(VERS) $@
INS.liblinkccc64= -$(RM) $@; $(SYMLINK) $(LIBLINKPATH)$(LIBLINKSCCC)$(VERS) $@

#
# If appropriate, augment POST_PROCESS_O and POST_PROCESS_SO to do CTF
# processing.  We'd like to just conditionally append to POST_PROCESS_O and
# POST_PROCESS_SO, but ParallelMake has a bug which causes the same value to
# sometimes get appended more than once, which will cause ctfconvert to fail.
# So, instead we introduce CTFCONVERT_POST and CTFMERGE_POST, which are always
# appended to POST_PROCESS_O and POST_PROCESS_SO but are no-ops unless CTF
# processing should be done.
#
CTFCONVERT_POST = :
CTFMERGE_POST   = :
POST_PROCESS_O += ; $(CTFCONVERT_POST)
POST_PROCESS_SO += ; $(CTFMERGE_POST)

CTFMERGE_LIB    = $(CTFMERGE) $(CTFMRGFLAGS) -t -f -L VERSION -o $@ $(PICS)

# conditional assignments

$(OBJS)  :=     sparc_CFLAGS += -xregs=no%appl
```



```
$(PICS)  :=     sparc_CFLAGS += -xregs=no%appl $(sparc_C_PICFLAGS)
$(PICS)  :=     sparcv9_CFLAGS += -xregs=no%appl $(sparcv9_C_PICFLAGS)
$(PICS)  :=     i386_CFLAGS += $(i386_C_PICFLAGS)
$(PICS)  :=     amd64_CFLAGS += $(amd64_C_PICFLAGS)
$(PICS)  :=     CCFLAGS += $(CC_PICFLAGS)
$(PICS)  :=     CPPFLAGS += -DPIC -D_REENTRANT
$(PICS)  :=     sparcv9_CCFLAGS += -xregs=no%appl $(sparcv9_CC_PICFLAGS)
$(PICS)  :=     amd64_CCFLAGS += $(amd64_CC_PICFLAGS)
$(PICS)  :=     CFLAGS += $(CTF_FLAGS)
$(PICS)  :=     CFLAGS64 += $(CTF_FLAGS)
$(PICS)  :=     CTFCONVERT_POST = $(CTFCONVERT_O)
$(DYNLIB) :=    CTFMERGE_POST = $(CTFMERGE_LIB)

$(LINTLIB):=    LOG = -DLOGGING
$(LIBRARY):=    AROBJS = $(OBJS)
$(LIBRARY):=    DIR = objs
$(DYNLIB):=     DIR = pics
$(DYNLIBCCC):=  DIR = pics

SONAMECCC=      $(DYNLIBCCC)
HSONAMECCC=     -h $(SONAMECCC)
#
# Keep in sync with the standard DYNFLAGS
#
$(DYNLIBCCC):=  DYNFLAGS = $(HSONAMECCC) $(ZTEXT) $(ZDEFS) \
                $(MAPFILES:%=-M%) $(MAPFILE.PGA:%=-M%) $(MAPFILE.NED:%=-M%) \
                $(BDIRECT) $(NORUNPATH)


# build rule for "portable" source
objs/%.o pics/%.o: %.c
        $(COMPILE.c) -o $@ $<
        $(POST_PROCESS_O)

objs/%.o pics/%.o: %.cc
        $(COMPILE.cc) -o $@ $<
        $(POST_PROCESS_O)

.PRECIOUS: $(LIBS)

# Define the majority text domain in this directory.
TEXT_DOMAIN= SUNW_OST_OSLIB

$(ROOTMAN3)/%: %.sunman
        $(INS.rename)

#
# For library source code, we expect that some symbols may not be used or
# may *appear* to be able to rescoped to static; shut lint up.  Never add
# a flag here unless you're *sure* that all libraries need to be linted
# with it.
#
LINTCHECKFLAGS = -m -erroff=E_NAME_DEF_NOT_USED2
LINTCHECKFLAGS += -erroff=E_NAME_DECL_NOT_USED_DEF2

#
# Target Architecture
#
TARGETMACH=     $(MACH)

#
# Allow people to define their own clobber rules.  Normal makefiles
# shouldn't override this - they should override $(CLOBBERFILES) instead.
#
CLOBBERTARGFILES= $(LIBS) $(DYNLIB) $(CLOBBERFILES)
```