

## Re: gcc/gnat 3.3

**Source:** <http://coding.derkeiler.com/Archive/Ada/comp.lang.ada/2003-11/0077.html>

---

**From:** Adrian Hoe ([adrianhoe\\_at\\_nowhere.com](mailto:adrianhoe_at_nowhere.com))

**Date:** 11/03/03

Date: Mon, 03 Nov 2003 17:36:34 +0800

Ludovic Brenta wrote:

```
> Adrian, you do not need to change your path. You can pass the CC
> environment variable to ./configure so it'll pick up the correct
> compiler driver. Like so in bash or ksh:
>
> $ CC=/opt/sfw/bin/gnat/gnatgcc ./configure --enable-languages=c,ada
>
> That's how I bootstrap gnat 3.15p on Debian GNU/Linux.
>
> In csh I suppose you'd do:
>
> % setenv CC /opt/sfw/bin/gnat/gcc
> % ./configure --enable-languages=c,ada
>
> HTH
```

Thanks for the tips. Now the configure recognized ada and created  
ada/Makefile in obj/gcc but I encountered an error during bootstrap:

Bootstrapping the compiler

```
make: Fatal error in reader: Makefile, line 987: Unexpected end of line seen
Current working directory /home/byhoe/download/gcc-3.3.2/gcc-3.3.2/obj/gcc
*** Error code 1
make: Fatal error: Command failed for target `bootstrap'
```

I touched those files as in the Build document of gcc.

I attach my obj/Makefile:

```
--
Adrian Hoe
m a i l b o x AT a d r i a n h o e . c o m
```

```
# This file was generated automatically by configure. Do not edit.
VPATH = ..
```

Re: gcc/gnat 3.3

```
links =
host_alias = sparc-sun-solaris2.9
host_cpu = sparc
host_vendor = sun
host_os = solaris2.9
host_canonical = sparc-sun-solaris2.9
target_alias = sparc-sun-solaris2.9
target_cpu = sparc
target_vendor = sun
target_os = solaris2.9
target_canonical = sparc-sun-solaris2.9
build_alias = sparc-sun-solaris2.9
build_cpu = sparc
build_vendor = sun
build_os = solaris2.9
build_canonical = sparc-sun-solaris2.9
host_makefile_frag = mh-frag
enable_shared = no
enable_threads = no
enable_version_specific_runtime_libs = no
gcc_version_trigger = /home/byhoe/download/gcc-3.3.2/gcc-3.3.2/gcc/version.c
gcc_version = 3.3.2

# Makefile.in is generated from Makefile.tpl by 'autogen Makefile.def'.
#
# Makefile for directory with subdirs to build.
# Copyright (C) 1990, 1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998,
# 1999, 2000, 2001, 2002 Free Software Foundation
#
# This file is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program; if not, write to the Free Software
# Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.
#

# Tell GNU make 3.79 not to run the top level in parallel. This
# prevents contention for $builddir/$target/config.cache, as well
# as minimizing scatter in file system caches.
NOTPARALLEL = .NOTPARALLEL
$(NOTPARALLEL):

srcdir = ..
```

```
prefix = /usr/local
exec_prefix = $(prefix)

bindir = ${exec_prefix}/bin
sbindir = ${exec_prefix}/sbin
libexecdir = ${exec_prefix}/libexec
datadir = ${prefix}/share
sysconfdir = ${prefix}/etc
sharedstatedir = ${prefix}/com
localstatedir = ${prefix}/var
libdir = ${exec_prefix}/lib
includedir = ${prefix}/include
oldincludedir = /usr/include
infodir = ${prefix}/info
mandir = ${prefix}/man
gxx_include_dir=${prefix}/include/c++/3.3.2

tooldir = $(exec_prefix)/sparc-sun-solaris2.9
build_tooldir = $(exec_prefix)/sparc-sun-solaris2.9

program_transform_name =

man1dir = $(mandir)/man1
man2dir = $(mandir)/man2
man3dir = $(mandir)/man3
man4dir = $(mandir)/man4
man5dir = $(mandir)/man5
man6dir = $(mandir)/man6
man7dir = $(mandir)/man7
man8dir = $(mandir)/man8
man9dir = $(mandir)/man9
# Directory in which the compiler finds executables, libraries, etc.
libsubdir = $(libdir)/gcc-lib/${target_alias}/${gcc_version}
GDB_NLM_DEPS =

SHELL = /bin/sh

# pwd command to use. Allow user to override default by setting PWDCMD in
# the environment to account for automounters. The make variable must not
# be called PWDCMD, otherwise the value set here is passed to make
# subprocesses and overrides the setting from the user's environment.
# Don't use PWD since it is a common shell environment variable and we
# don't want to corrupt it.
PWD_COMMAND = ${PWDCMD-pwd}

# INSTALL_PROGRAM_ARGS is changed by configure.in to use -x for a
# cygwin host.
INSTALL_PROGRAM_ARGS =

INSTALL = $(SHELL) $$s/install-sh -c
INSTALL_PROGRAM = $(INSTALL) $(INSTALL_PROGRAM_ARGS)
```

```
INSTALL_SCRIPT = $(INSTALL)
INSTALL_DATA = $(INSTALL) -m 644

INSTALL_DOSREL = install-dosrel-fake

AS = as
AR = ar
AR_FLAGS = rc
CC = /opt/sfw/bin/gnat/gcc

# Special variables passed down in EXTRA_GCC_FLAGS. They are defined
# here so that they can be overridden by Makefile fragments.
HOST_CC = $(CC_FOR_BUILD)
BUILD_PREFIX =
BUILD_PREFIX_1 = loser-

# These flag values are normally overridden by the configure script.
CFLAGS = -g -O2
CXXFLAGS = -g -O2

LDLFLAGS =
LIBCFLAGS = $(CFLAGS)
CFLAGS_FOR_BUILD = $(CFLAGS)
# During gcc bootstrap, if we use some random cc for stage1 then
# CFLAGS will be just -g. We want to ensure that TARGET libraries
# (which we know are built with gcc) are built with optimizations so
# prepend -O2 when setting CFLAGS_FOR_TARGET.
CFLAGS_FOR_TARGET = -O2 $(CFLAGS)
LDLFLAGS_FOR_TARGET =
LIBCFLAGS_FOR_TARGET = $(CFLAGS_FOR_TARGET)
PICFLAG =
PICFLAG_FOR_TARGET =

CXX = c++

# Use -O2 to stress test the compiler.
LIBCXXFLAGS = $(CXXFLAGS) -fno-implicit-templates
CXXFLAGS_FOR_TARGET = $(CXXFLAGS)
LIBCXXFLAGS_FOR_TARGET = $(CXXFLAGS_FOR_TARGET) -fno-implicit-templates

DLLTOOL = dlltool
WINDRES = windres

NM = nm

LD = ld

# These values are substituted by configure.
DEFAULT_YACC = bison -y
DEFAULT_LEX = flex
DEFAULT_M4 = gm4
```

```

BISON = `if [ -f $$r/bison/bison ] ; then \
    echo $$r/bison/bison -L $$s/bison/ ; \
else \
    echo bison ; \
fi`

YACC = `if [ -f $$r/bison/bison ] ; then \
    echo $$r/bison/bison -y -L $$s/bison/ ; \
elif [ -f $$r/byacc/byacc ] ; then \
    echo $$r/byacc/byacc ; \
else \
    echo ${DEFAULT_YACC} ; \
fi`

LEX = `if [ -f $$r/flex/flex ] ; \
then echo $$r/flex/flex ; \
else echo ${DEFAULT_LEX} ; fi`

M4 = `if [ -f $$r/m4/m4 ] ; \
then echo $$r/m4/m4 ; \
else echo ${DEFAULT_M4} ; fi`

# For an installed makeinfo, we require it to be from texinfo 4.2 or
# higher, else we use the "missing" dummy.
MAKEINFO = `if [ -f $$r/texinfo/makeinfo/makeinfo ] ; \
then echo $$r/texinfo/makeinfo/makeinfo ; \
else if (makeinfo --version \
| egrep 'texinfo[^0-9]*([1-3][0-9]|4\.[2-9]|[5-9])') >/dev/null 2>&1 ; \
then echo makeinfo; else echo $$s/missing makeinfo; fi; fi`

# This just becomes part of the MAKEINFO definition passed down to
# sub-makes. It lets flags be given on the command line while still
# using the makeinfo from the object tree.
# (Default to avoid splitting info files.)
MAKEINFOFLAGS = --no-split

EXPECT = `if [ -f $$r/expect/expect ] ; \
then echo $$r/expect/expect ; \
else echo expect ; fi`

RUNTEST = `if [ -f $$s/dejagnum/runtest ] ; \
then echo $$s/dejagnum/runtest ; \
else echo runtest ; fi`

# compilers to use to create programs which must be run in the build
# environment.
CC_FOR_BUILD = $(CC)
CXX_FOR_BUILD = $(CXX)

SUBDIRS = libiberty gcc

```

```
# This is set by the configure script to the list of directories which
# should be built using the target tools.
TARGET_CONFIGDIRS =

# Target libraries are put under this directory:
# Changed by configure to $(target_alias) if cross.
TARGET_SUBDIR = sparc-sun-solaris2.9

BUILD_CONFIGDIRS = libiberty
BUILD_SUBDIR = sparc-sun-solaris2.9

# This is set by the configure script to the arguments to use when configuring
# directories built for the target.
TARGET_CONFIGARGS = --cache-file=../config.cache --host=sparc-sun-solaris2.9
--build=sparc-sun-solaris2.9 --enable-multilib
--with-gcc-version-trigger=/home/byhoe/download/gcc-3.3.2/gcc-3.3.2/gcc/version.c
--enable-languages=c,ada

# This is set by the configure script to the arguments to use when configuring
# directories built for the build system.
BUILD_CONFIGARGS = --cache-file=../config.cache --build=sparc-sun-solaris2.9
--host=sparc-sun-solaris2.9
--with-gcc-version-trigger=/home/byhoe/download/gcc-3.3.2/gcc-3.3.2/gcc/version.c
--enable-languages=c,ada

# This is set by configure to REALLY_SET_LIB_PATH if --enable-shared
# was used.
SET_LIB_PATH =

# This is the name of the environment variable used for the path to
# the libraries. This may be changed by configure.in.
RPATH_ENVVAR = LD_LIBRARY_PATH

# This is the list of directories that may be needed in RPATH_ENVVAR
# so that programs built for the host machine work.
HOST_LIB_PATH = $$r/bfd:$$r/opcodes

# This is the list of directories that may be needed in RPATH_ENVVAR
# so that programs built for the target machine work.
TARGET_LIB_PATH = $$r/$(TARGET_SUBDIR)/libstdc++-v3/src/.libs:

# configure.in sets SET_LIB_PATH to this if --enable-shared was used.
# Some platforms don't like blank entries, so we remove duplicate,
# leading and trailing colons.
REALLY_SET_LIB_PATH = \
  $(RPATH_ENVVAR)=`echo "$(HOST_LIB_PATH):$(TARGET_LIB_PATH):$$$(RPATH_ENVVAR)" |
sed 's,::*,:,g;s,^:*,s,:*$$,^'; export $(RPATH_ENVVAR);

ALL = all.normal
INSTALL_TARGET = installdirs \
  install-gcc \
```

```

$(INSTALL_MODULES) \
$(INSTALL_TARGET_MODULES) \
$(INSTALL_X11_MODULES) \
$(INSTALL_DOSREL)

INSTALL_TARGET_CROSS = installdirs \
install-gcc-cross \
$(INSTALL_MODULES) \
$(INSTALL_TARGET_MODULES) \
$(INSTALL_X11_MODULES) \
$(INSTALL_DOSREL)

# Should be subst'd by configure.in
FLAGS_FOR_TARGET = -B$(build_tooldir)/bin/ -B$(build_tooldir)/lib/ -isystem $(build_tooldir)/include
CC_FOR_TARGET = $(STAGE_CC_WRAPPER) $$r/gcc/xgcc -B$$r/gcc/ $(FLAGS_FOR_TARGET)
CXX_FOR_TARGET = $(STAGE_CC_WRAPPER) $$r/gcc/ case $$dir in libstdc++-v3 | libjava) echo xgcc
-shared-libgcc ;; *) echo g++ ;; esac ` -B$$r/gcc/ -nostdinc++ ` case $$dir in libstdc++-v3 | libjava) ;; *) test
! -f $$r/$(TARGET_SUBDIR)/libstdc++-v3/testsuite_flags || $(SHELL)
$$r/$(TARGET_SUBDIR)/libstdc++-v3/testsuite_flags --build-includes;; esac `
-L$$r/$(TARGET_SUBDIR)/libstdc++-v3/src -L$$r/$(TARGET_SUBDIR)/libstdc++-v3/src/.libs
$(FLAGS_FOR_TARGET)
CXX_FOR_TARGET_FOR_RECURSIVE_MAKE = $(STAGE_CC_WRAPPER) $$$r/gcc/ case $$$dir
in libstdc++-v3 | libjava) echo xgcc -shared-libgcc ;; *) echo g++ ;; esac ` -B$$$r/gcc/ -nostdinc++ ` case
$$$dir in libstdc++-v3 | libjava) ;; *) test ! -f $$$r/$(TARGET_SUBDIR)/libstdc++-v3/testsuite_flags ||
$(SHELL) $$$r/$(TARGET_SUBDIR)/libstdc++-v3/testsuite_flags --build-includes;; esac `
-L$$$r/$(TARGET_SUBDIR)/libstdc++-v3/src -L$$$r/$(TARGET_SUBDIR)/libstdc++-v3/src/.libs
$(FLAGS_FOR_TARGET)
GCJ_FOR_TARGET = $(STAGE_CC_WRAPPER) $$r/gcc/gcj -B$$r/gcc/ $(FLAGS_FOR_TARGET)

# If GCC_FOR_TARGET is not overridden on the command line, then this
# variable is passed down to the gcc Makefile, where it is used to
# build libgcc2.a. We define it here so that it can itself be
# overridden on the command line.
GCC_FOR_TARGET = $(STAGE_CC_WRAPPER) $$r/gcc/xgcc -B$$r/gcc/ $(FLAGS_FOR_TARGET)

AS_FOR_TARGET = ` \
if [ -f $$r/gas/as-new ] ; then \
echo $$r/gas/as-new ; \
elif [ -f $$r/gcc/xgcc ] ; then \
$(CC_FOR_TARGET) -print-prog-name=as ; \
else \
if [ '(host_canonical)' = '(target_canonical)' ] ; then \
echo $(AS); \
else \
t='$(program_transform_name)'; echo as | sed -e 's/x/x/' $$t ; \
fi; \
fi`

LD_FOR_TARGET = ` \
if [ -f $$r/ld/ld-new ] ; then \
echo $$r/ld/ld-new ; \

```

```

elif [ -f $$r/gcc/xgcc ]; then \
  $(CC_FOR_TARGET) -print-prog-name=ld ; \
else \
  if [ '$(host_canonical)' = '$(target_canonical)' ]; then \
    echo $(LD); \
  else \
    t='$(program_transform_name)'; echo ld | sed -e 's/x/x/' $$t ; \
  fi; \
fi`

```

```

DLLTOOL_FOR_TARGET = ` \
if [ -f $$r/binutils/dlltool ]; then \
  echo $$r/binutils/dlltool ; \
else \
  if [ '$(host_canonical)' = '$(target_canonical)' ]; then \
    echo $(DLLTOOL); \
  else \
    t='$(program_transform_name)'; echo dlltool | sed -e 's/x/x/' $$t ; \
  fi; \
fi`

```

```

WINDRES_FOR_TARGET = ` \
if [ -f $$r/binutils/windres ]; then \
  echo $$r/binutils/windres ; \
else \
  if [ '$(host_canonical)' = '$(target_canonical)' ]; then \
    echo $(WINDRES); \
  else \
    t='$(program_transform_name)'; echo windres | sed -e 's/x/x/' $$t ; \
  fi; \
fi`

```

```

AR_FOR_TARGET = ` \
if [ -f $$r/binutils/ar ]; then \
  echo $$r/binutils/ar ; \
else \
  if [ '$(host_canonical)' = '$(target_canonical)' ]; then \
    echo $(AR); \
  else \
    t='$(program_transform_name)'; echo ar | sed -e 's/x/x/' $$t ; \
  fi; \
fi`

```

```

RANLIB_FOR_TARGET = ` \
if [ -f $$r/binutils/ranlib ]; then \
  echo $$r/binutils/ranlib ; \
else \
  if [ '$(host_canonical)' = '$(target_canonical)' ]; then \
    if [ x'$(RANLIB)' != x ]; then \
      echo $(RANLIB); \
    else \

```

```

    echo ranlib; \
    fi; \
else \
    t="$(program_transform_name)"; echo ranlib | sed -e 's/x/x/' $$t ; \
    fi; \
fi`

NM_FOR_TARGET = ` \
if [ -f $$r/binutils/nm-new ] ; then \
    echo $$r/binutils/nm-new ; \
elif [ -f $$r/gcc/xgcc ] ; then \
    $(CC_FOR_TARGET) -print-prog-name=nm ; \
else \
    if [ "$(host_canonical)" = "$(target_canonical)" ] ; then \
        echo $(NM); \
    else \
        t="$(program_transform_name)"; echo nm | sed -e 's/x/x/' $$t ; \
    fi; \
fi`

# The first rule in the file had better be this one. Don't put any above it.
# This lives here to allow makefile fragments to contain dependencies.
all: all.normal
.PHONY: all

# These can be overridden by config/mt-*.
# The _TARGET_ is because they're specified in mt-foo.
# The _HOST_ is because they're programs that run on the host.
EXTRA_TARGET_HOST_ALL_MODULES =
EXTRA_TARGET_HOST_INSTALL_MODULES =
EXTRA_TARGET_HOST_CHECK_MODULES =

#### host and target specific makefile fragments come in here.
# Makefile changes for Suns running Solaris 2

RANLIB = true

X11_EXTRA_LIBS = -lnsl -lsocket
###

# Flags to pass down to all sub-makes.
# Please keep these in alphabetical order.
BASE_FLAGS_TO_PASS = \
    "AR_FLAGS=$(AR_FLAGS)" \
    "AR_FOR_TARGET=$(AR_FOR_TARGET)" \
    "AS_FOR_TARGET=$(AS_FOR_TARGET)" \
    "BISON=$(BISON)" \
    "CC_FOR_BUILD=$(CC_FOR_BUILD)" \
    "CC_FOR_TARGET=$(CC_FOR_TARGET)" \
    "CFLAGS=$(CFLAGS)" \
    "CFLAGS_FOR_TARGET=$(CFLAGS_FOR_TARGET)" \

```

```

"GCJ_FOR_TARGET=$(GCJ_FOR_TARGET)" \
"CXX_FOR_BUILD=$(CXX_FOR_BUILD)" \
"CXXFLAGS=$(CXXFLAGS)" \
"CXXFLAGS_FOR_TARGET=$(CXXFLAGS_FOR_TARGET)" \
"CXX_FOR_TARGET=$(CXX_FOR_TARGET)" \
"DESTDIR=$(DESTDIR)" \
"DLLTOOL_FOR_TARGET=$(DLLTOOL_FOR_TARGET)" \
"INSTALL=$(INSTALL)" \
"INSTALL_DATA=$(INSTALL_DATA)" \
"INSTALL_PROGRAM=$(INSTALL_PROGRAM)" \
"INSTALL_SCRIPT=$(INSTALL_SCRIPT)" \
"LDFLAGS=$(LDFLAGS)" \
"LEX=$(LEX)" \
"LD_FOR_TARGET=$(LD_FOR_TARGET)" \
"LIBCFLAGS=$(LIBCFLAGS)" \
"LIBCFLAGS_FOR_TARGET=$(LIBCFLAGS_FOR_TARGET)" \
"LIBCXXFLAGS=$(LIBCXXFLAGS)" \
"LIBCXXFLAGS_FOR_TARGET=$(LIBCXXFLAGS_FOR_TARGET)" \
"M4=$(M4)" \
"MAKE=$(MAKE)" \
"MAKEINFO=$(MAKEINFO) $(MAKEINFOFLAGS)" \
"NM_FOR_TARGET=$(NM_FOR_TARGET)" \
"RANLIB_FOR_TARGET=$(RANLIB_FOR_TARGET)" \
"RPATH_ENVVAR=$(RPATH_ENVVAR)" \
"SHELL=$(SHELL)" \
"EXPECT=$(EXPECT)" \
"RUNTEST=$(RUNTEST)" \
"RUNTESTFLAGS=$(RUNTESTFLAGS)" \
"TARGET_SUBDIR=$(TARGET_SUBDIR)" \
"WINDRES_FOR_TARGET=$(WINDRES_FOR_TARGET)" \
"YACC=$(YACC)" \
"bindir=$(bindir)" \
"datadir=$(datadir)" \
"exec_prefix=$(exec_prefix)" \
"includedir=$(includedir)" \
"infodir=$(infodir)" \
"libdir=$(libdir)" \
"libexecdir=$(libexecdir)" \
"lispdir=$(lispdir)" \
"libstdcxx_incdir=$(libstdcxx_incdir)" \
"libsubdir=$(libsubdir)" \
"localstatedir=$(localstatedir)" \
"mandir=$(mandir)" \
"oldincludedir=$(oldincludedir)" \
"prefix=$(prefix)" \
"sbindir=$(sbindir)" \
"sharedstatedir=$(sharedstatedir)" \
"sysconfdir=$(sysconfdir)" \
"tooldir=$(tooldir)" \
"build_tooldir=$(build_tooldir)" \
"gxx_include_dir=$(gxx_include_dir)" \

```

```

"gcc_version=$(gcc_version)" \
"gcc_version_trigger=$(gcc_version_trigger)" \
"target_alias=$(target_alias)"

# For any flags above that may contain shell code that varies from one
# target library to another. When doing recursive invocations of the
# top-level Makefile, we don't want the outer make to evaluate them,
# so we pass these variables down unchanged. They must not contain
# single nor double quotes.
RECURSE_FLAGS = \
    CXX_FOR_TARGET='$(CXX_FOR_TARGET_FOR_RECURSIVE_MAKE)'

# Flags to pass down to most sub-makes, in which we're building with
# the host environment.
# If any variables are added here, they must be added to do-*, below.
EXTRA_HOST_FLAGS = \
    'AR=$(AR)' \
    'AS=$(AS)' \
    'CC=$(CC)' \
    'CXX=$(CXX)' \
    'DLLTOOL=$(DLLTOOL)' \
    'LD=$(LD)' \
    'NM=$(NM)' \
    "`echo 'RANLIB=$(RANLIB)' | sed -e s/.*=$$/XFOO=/" \
    'WINDRES=$(WINDRES)'

FLAGS_TO_PASS = $(BASE_FLAGS_TO_PASS) $(EXTRA_HOST_FLAGS)

# Flags that are concerned with the location of the X11 include files
# and library files
#
# NOTE: until the top-level is getting the values via autoconf, it only
# causes problems to have this top-level Makefile overriding the autoconf-set
# values in child directories. Only variables that don't conflict with
# autoconf'ed ones should be passed by X11_FLAGS_TO_PASS for now.
#
X11_FLAGS_TO_PASS = \
    'X11_EXTRA_CFLAGS=$(X11_EXTRA_CFLAGS)' \
    'X11_EXTRA_LIBS=$(X11_EXTRA_LIBS)'

# Flags to pass down to makes which are built with the target environment.
# The double $ decreases the length of the command line; the variables
# are set in BASE_FLAGS_TO_PASS, and the sub-make will expand them.
# If any variables are added here, they must be added to do-*, below.
EXTRA_TARGET_FLAGS = \
    'AR=$$(AR_FOR_TARGET)' \
    'AS=$$(AS_FOR_TARGET)' \
    'CC=$$(CC_FOR_TARGET)' \
    'CFLAGS=$$(CFLAGS_FOR_TARGET)' \
    'CXX=$$(CXX_FOR_TARGET)' \
    'CXXFLAGS=$$(CXXFLAGS_FOR_TARGET)' \

```

```
'DLLTOOL=${DLLTOOL_FOR_TARGET}' \
'LD=${LD_FOR_TARGET}' \
'LIBCFLAGS=${LIBCFLAGS_FOR_TARGET}' \
'LIBCXXFLAGS=${LIBCXXFLAGS_FOR_TARGET}' \
'NM=${NM_FOR_TARGET}' \
'RANLIB=${RANLIB_FOR_TARGET}' \
'WINDRES=${WINDRES_FOR_TARGET}'
```

```
TARGET_FLAGS_TO_PASS = $(BASE_FLAGS_TO_PASS) $(EXTRA_TARGET_FLAGS)
```

```
# Flags to pass down to gcc. gcc builds a library, libgcc.a, so it
# unfortunately needs the native compiler and the target ar and
# ranlib.
# If any variables are added here, they must be added to do-*, below.
# The HOST_* variables are a special case, which are used for the gcc
# cross-building scheme.
```

```
EXTRA_GCC_FLAGS = \
  'AR=$(AR)' \
  'AS=$(AS)' \
  'CC=$(CC)' \
  'CXX=$(CXX)' \
  'DLLTOOL=${DLLTOOL_FOR_TARGET}' \
  'HOST_CC=$(CC_FOR_BUILD)' \
  'BUILD_PREFIX=$(BUILD_PREFIX)' \
  'BUILD_PREFIX_1=$(BUILD_PREFIX_1)' \
  'NM=$(NM)' \
  ""echo 'RANLIB=$(RANLIB)' | sed -e s/.*=$$/XFOO=/" \
  'WINDRES=${WINDRES_FOR_TARGET}' \
  "GCC_FOR_TARGET=$(GCC_FOR_TARGET)" \
  "CFLAGS_FOR_BUILD=$(CFLAGS_FOR_BUILD)" \
  ""echo 'LANGUAGES=$(LANGUAGES)' | sed -e s/.*=$$/XFOO=/" \
  ""echo 'STMP_FIXPROTO=$(STMP_FIXPROTO)' | sed -e s/.*=$$/XFOO=/" \
  ""echo 'LIMITS_H_TEST=$(LIMITS_H_TEST)' | sed -e s/.*=$$/XFOO=/" \
  ""echo 'LIBGCC2_CFLAGS=$(LIBGCC2_CFLAGS)' | sed -e s/.*=$$/XFOO=/" \
  ""echo 'LIBGCC2_DEBUG_CFLAGS=$(LIBGCC2_DEBUG_CFLAGS)' | sed -e s/.*=$$/XFOO=/" \
  ""echo 'LIBGCC2_INCLUDES=$(LIBGCC2_INCLUDES)' | sed -e s/.*=$$/XFOO=/" \
  ""echo 'ENQUIRE=$(ENQUIRE)' | sed -e s/.*=$$/XFOO=/" \
  ""echo 'STAGE1_CFLAGS=$(STAGE1_CFLAGS)' | sed -e s/.*=$$/XFOO=/" \
  ""echo 'BOOT_CFLAGS=$(BOOT_CFLAGS)' | sed -e s/.*=$$/XFOO=/"
```

```
GCC_FLAGS_TO_PASS = $(BASE_FLAGS_TO_PASS) $(EXTRA_GCC_FLAGS)
```

```
# This is a list of the targets for all of the modules which are compiled
# using the build machine's native compiler. Configure edits the second
# macro for build!=host builds.
```

```
ALL_BUILD_MODULES_LIST = \
  all-build-libiberty
ALL_BUILD_MODULES =
```

```
# This is a list of the configure targets for all of the modules which
# are compiled using the native tools.
```

```
CONFIGURE_BUILD_MODULES = \  
  configure-build-libiberty  
  
# This is a list of the targets for all of the modules which are compiled  
# using $(FLAGS_TO_PASS).  
ALL_MODULES = \  
  all-ash \  
  all-autoconf \  
  all-automake \  
  all-bash \  
  all-bfd \  
  all-opcodes \  
  all-binutils \  
  all-bison \  
  all-byacc \  
  all-bzip2 \  
  all-db \  
  all-dejagnu \  
  all-diff \  
  all-dosutils \  
  all-etc \  
  all-fastjar \  
  all-fileutils \  
  all-findutils \  
  all-find \  
  all-flex \  
  all-gas \  
  all-gawk \  
  all-gettext \  
  all-gnuserv \  
  all-gprof \  
  all-grep \  
  all-gzip \  
  all-hello \  
  all-indent \  
  all-intl \  
  all-tcl \  
  all-itcl \  
  all-ld \  
  all-libgui \  
  all-libiberty \  
  all-libtool \  
  all-m4 \  
  all-make \  
  all-mmalloc \  
  all-patch \  
  all-perl \  
  all-prms \  
  all-rcs \  
  all-readline \  
  all-release \  

```

```
all-recode \  
all-sed \  
all-send-pr \  
all-shellutils \  
all-sid \  
all-sim \  
all-navigator \  
all-tar \  
all-texinfo \  
all-textutils \  
all-time \  
all-uudecode \  
all-wdiff \  
all-zip \  
all-zlib \  
$(EXTRA_TARGET_HOST_ALL_MODULES)
```

```
# This is a list of the check targets for all of the modules which are  
# compiled using $(FLAGS_TO_PASS).  
#
```

```
# The list is in two parts. The first lists those tools which  
# are tested as part of the host's native tool-chain, and not  
# tested in a cross configuration.
```

```
NATIVE_CHECK_MODULES = \  
  check-bison \  
  check-byacc \  
  check-fastjar \  
  check-flex \  
  check-zip
```

```
CROSS_CHECK_MODULES = \  
  check-ash \  
  check-autoconf \  
  check-automake \  
  check-bash \  
  check-bfd \  
  check-opcodes \  
  check-binutils \  
  check-bzip2 \  
  check-db \  
  check-dejagnu \  
  check-diff \  
  check-etc \  
  check-fileutils \  
  check-findutils \  
  check-find \  
  check-gas \  
  check-gawk \  
  check-gettext \  
  check-gnuserv \  
  check-gprof \  
  check-gzip
```

```
check-grep \  
check-gzip \  
check-hello \  
check-indent \  
check-intl \  
check-tcl \  
check-itcl \  
check-ld \  
check-libgui \  
check-libiberty \  
check-libtool \  
check-m4 \  
check-make \  
check-patch \  
check-perl \  
check-prms \  
check-rs \  
check-readline \  
check-recode \  
check-sed \  
check-send-pr \  
check-shellutils \  
check-sid \  
check-sim \  
check-snavigator \  
check-tar \  
check-texinfo \  
check-textutils \  
check-time \  
check-uudecode \  
check-wdiff \  
$(EXTRA_TARGET_HOST_CHECK_MODULES)
```

```
CHECK_MODULES=$(NATIVE_CHECK_MODULES) $(CROSS_CHECK_MODULES)
```

```
# This is a list of the install targets for all of the modules which are  
# compiled using $(FLAGS_TO_PASS).
```

```
INSTALL_MODULES = \  
  install-ash \  
  install-autoconf \  
  install-automake \  
  install-bash \  
  install-bfd \  
  install-opcodes \  
  install-binutils \  
  install-bison \  
  install-byacc \  
  install-bzip2 \  
  install-db \  
  install-dejagnu \  
  install-diff \  
  $(EXTRA_TARGET_HOST_INSTALL_MODULES)
```



```
all-guile \  
all-tclX \  
all-tk \  
all-tix
```

```
# This is a list of the check targets for all of the modules which are  
# compiled using $(X11_FLAGS_TO_PASS).
```

```
CHECK_X11_MODULES = \  
  check-gdb \  
  check-guile \  
  check-expect \  
  check-tclX \  
  check-tk \  
  check-tix
```

```
# This is a list of the install targets for all the modules which are  
# compiled using $(X11_FLAGS_TO_PASS).
```

```
INSTALL_X11_MODULES = \  
  install-gdb \  
  install-guile \  
  install-expect \  
  install-tclX \  
  install-tk \  
  install-tix
```

```
# This is a list of the targets for all of the modules which are compiled  
# using $(TARGET_FLAGS_TO_PASS).
```

```
ALL_TARGET_MODULES = \  
  all-target-libstdc++-v3 \  
  all-target-newlib \  
  all-target-libf2c \  
  all-target-libobjc \  
  all-target-libtermcap \  
  all-target-winsup \  
  all-target-libgloss \  
  all-target-libiberty \  
  all-target-gperf \  
  all-target-examples \  
  all-target-libffi \  
  all-target-libjava \  
  all-target-zlib \  
  all-target-boehm-gc \  
  all-target-qthreads \  
  all-target-rda
```

```
# This is a list of the configure targets for all of the modules which  
# are compiled using the target tools.
```

```
CONFIGURE_TARGET_MODULES = \  
  configure-target-libstdc++-v3 \  
  configure-target-newlib \  
  configure-target-libf2c \  
  configure-target-libffi
```

```
configure-target-libobjc \  
configure-target-libtermcap \  
configure-target-winsup \  
configure-target-libgloss \  
configure-target-libiberty \  
configure-target-gperf \  
configure-target-examples \  
configure-target-libffi \  
configure-target-libjava \  
configure-target-zlib \  
configure-target-boehm-gc \  
configure-target-qthreads \  
configure-target-rda
```

```
# This is a list of the check targets for all of the modules which are  
# compiled using $(TARGET_FLAGS_TO_PASS).
```

```
CHECK_TARGET_MODULES = \  
  check-target-libstdc++-v3 \  
  check-target-newlib \  
  check-target-libf2c \  
  check-target-libobjc \  
  check-target-winsup \  
  check-target-libiberty \  
  check-target-gperf \  
  check-target-libffi \  
  check-target-libjava \  
  check-target-zlib \  
  check-target-boehm-gc \  
  check-target-qthreads \  
  check-target-rda
```

```
# This is a list of the install targets for all of the modules which are  
# compiled using $(TARGET_FLAGS_TO_PASS).
```

```
INSTALL_TARGET_MODULES = \  
  install-target-libstdc++-v3 \  
  install-target-newlib \  
  install-target-libf2c \  
  install-target-libobjc \  
  install-target-libtermcap \  
  install-target-winsup \  
  install-target-libgloss \  
  install-target-libiberty \  
  install-target-gperf \  
  install-target-libffi \  
  install-target-libjava \  
  install-target-zlib \  
  install-target-boehm-gc \  
  install-target-qthreads \  
  install-target-rda
```

# This is a list of the targets for which we can do a clean-{target}.

```
CLEAN_MODULES = \  
  clean-ash \  
  clean-autoconf \  
  clean-automake \  
  clean-bash \  
  clean-bfd \  
  clean-opcodes \  
  clean-binutils \  
  clean-bison \  
  clean-byacc \  
  clean-bzip2 \  
  clean-db \  
  clean-dejagnu \  
  clean-diff \  
  clean-dosutils \  
  clean-etc \  
  clean-fastjar \  
  clean-fileutils \  
  clean-findutils \  
  clean-find \  
  clean-flex \  
  clean-gas \  
  clean-gawk \  
  clean-gettext \  
  clean-gnuserv \  
  clean-gprof \  
  clean-grep \  
  clean-gzip \  
  clean-hello \  
  clean-indent \  
  clean-intl \  
  clean-tcl \  
  clean-itcl \  
  clean-ld \  
  clean-libgui \  
  clean-libiberty \  
  clean-libtool \  
  clean-m4 \  
  clean-make \  
  clean-mmalloc \  
  clean-patch \  
  clean-perl \  
  clean-prms \  
  clean-rcs \  
  clean-readline \  
  clean-release \  
  clean-recode \  
  clean-sed \  
  clean-send-pr \  
  clean-shellutils \  

```

```
clean-sid \  
clean-sim \  
clean-snavigator \  
clean-tar \  
clean-texinfo \  
clean-textutils \  
clean-time \  
clean-uudecode \  
clean-wdiff \  
clean-zip \  
clean-zlib
```

# All of the target modules that can be cleaned

```
CLEAN_TARGET_MODULES = \  
clean-target-libstdc++-v3 \  
clean-target-newlib \  
clean-target-libf2c \  
clean-target-libobjc \  
clean-target-winsup \  
clean-target-libgloss \  
clean-target-libiberty \  
clean-target-gperf \  
clean-target-examples \  
clean-target-libffi \  
clean-target-libjava \  
clean-target-zlib \  
clean-target-boehm-gc \  
clean-target-qthreads \  
clean-target-rda
```

# All of the x11 modules that can be cleaned

```
CLEAN_X11_MODULES = \  
clean-gdb \  
clean-expect \  
clean-guile \  
clean-tclX \  
clean-tk \  
clean-tix
```

# The target built for a native build.

```
.PHONY: all.normal  
all.normal: \  
$(ALL_BUILD_MODULES) \  
$(ALL_MODULES) \  
$(ALL_X11_MODULES) \  
$(ALL_TARGET_MODULES) \  
all-gcc
```

# Do a target for all the subdirectories. A ``make do-X" will do a  
# ``make X" in all subdirectories (because, in general, there is a  
# dependency (below) of X upon do-X, a ``make X" will also do this,

```

# but it may do additional work as well).
# This target ensures that $(BASE_FLAGS_TO_PASS) appears only once,
# because it is so large that it can easily overflow the command line
# length limit on some systems.
DO_X = \
do-clean \
do-distclean \
do-dvi \
do-info \
do-install-info \
do-installcheck \
do-mostlyclean \
do-maintainer-clean \
do-TAGS
.PHONY: $(DO_X)
$(DO_X):
@target=`echo $@ | sed -e 's/^do-//'; \
r=`${PWD_COMMAND}`; export r; \
s=`cd $(srcdir); ${PWD_COMMAND}`; export s; \
$(SET_LIB_PATH) \
for i in $(SUBDIRS) -dummy-; do \
if [ -f ./$$i/Makefile ]; then \
case $$i in \
gcc) \
for flag in $(EXTRA_GCC_FLAGS); do \
eval `echo "$$flag" | sed -e "s|^([^\=]*)=(.*)|1='\2'; export \1|"; \
done; \
;; \
*) \
for flag in $(EXTRA_HOST_FLAGS); do \
eval `echo "$$flag" | sed -e "s|^([^\=]*)=(.*)|1='\2'; export \1|"; \
done; \
;; \
esac; \
if (cd ./$$i; \
$(MAKE) $(BASE_FLAGS_TO_PASS) "AR=${AR}" "AS=${AS}" \
"CC=${CC}" "CXX=${CXX}" "LD=${LD}" "NM=${NM}" \
`echo `RANLIB=${RANLIB}` | sed -e 's/.*=${XFOO=}/' \
"DLLTOOL=${DLLTOOL}" "WINDRES=${WINDRES}" \
$$${target}); \
then true; else exit 1; fi; \
else true; fi; \
done
@target=`echo $@ | sed -e 's/^do-//'; \
r=`${PWD_COMMAND}`; export r; \
s=`cd $(srcdir); ${PWD_COMMAND}`; export s; \
$(SET_LIB_PATH) \
for i in $(TARGET_CONFIGDIRS) -dummy-; do \
if [ -f $(TARGET_SUBDIR)/$$i/Makefile ]; then \
for flag in $(EXTRA_TARGET_FLAGS); do \
eval `echo "$$flag" | sed -e "s|^([^\=]*)=(.*)|1='\2'; export \1|"; \

```

```
done; \
if (cd $(TARGET_SUBDIR)/$$i; \
    $(MAKE) $(BASE_FLAGS_TO_PASS) "AR=${AR}" "AS=${AS}" \
        "CC=${CC}" "CXX=${CXX}" "LD=${LD}" "NM=${NM}" \
        "`echo \"RANLIB=${RANLIB}\" | sed -e 's/.*=${XFOO}/^'\" \
        "DLLTOOL=${DLLTOOL}" "WINDRES=${WINDRES}" \
        $$target); \
    then true; else exit 1; fi; \
else true; fi; \
done
```

# Here are the targets which correspond to the do-X targets.

```
.PHONY: info installcheck dvi install-info
.PHONY: clean distclean mostlyclean maintainer-clean realclean
.PHONY: local-clean local-distclean local-maintainer-clean
info: do-info
installcheck: do-installcheck
dvi: do-dvi
```

# Make sure makeinfo is built before we do a `make info'.  
do-info: all-texinfo

```
install-info: do-install-info dir.info
s=`cd $(srcdir); ${PWD_COMMAND}`; export s; \
if [ -f dir.info ] ; then \
    $(INSTALL_DATA) dir.info $(DESTDIR)$(infodir)/dir.info ; \
else true ; fi
```

```
local-clean:
-rm -f *.a TEMP errs core *.o *~ \#* TAGS *.E *.log
```

```
local-distclean:
-rm -f Makefile config.status config.cache mh-frag mt-frag
-if [ "$(TARGET_SUBDIR)" != "." ]; then \
    rm -rf $(TARGET_SUBDIR); \
else true; fi
-rm -f texinfo/po/Makefile texinfo/po/Makefile.in texinfo/info/Makefile
-rm -f texinfo/doc/Makefile texinfo/po/POTFILES
-rmdir texinfo/doc texinfo/info texinfo/intl texinfo/lib 2>/dev/null
-rmdir texinfo/makeinfo texinfo/po texinfo/util 2>/dev/null
-rmdir fastjar gcc libiberty texinfo zlib 2>/dev/null
```

```
local-maintainer-clean:
@echo "This command is intended for maintainers to use;"
@echo "it deletes files that may require special tools to rebuild."
```

```
clean: do-clean local-clean
mostlyclean: do-mostlyclean local-clean
distclean: do-distclean local-clean local-distclean
maintainer-clean: local-maintainer-clean do-maintainer-clean local-clean
```

maintainer-clean: local-distclean  
realclean: maintainer-clean

# This rule is used to clean specific modules.

.PHONY: \$(CLEAN\_MODULES) \$(CLEAN\_X11\_MODULES) clean-gcc

\$(CLEAN\_MODULES) \$(CLEAN\_X11\_MODULES) clean-gcc:

```
@dir=`echo $@ | sed -e 's/clean-//'^`; \  
if [ -f ./${dir}/Makefile ] ; then \  
  r=`${PWD_COMMAND}`; export r; \  
  s=`cd $(srcdir); ${PWD_COMMAND}`; export s; \  
  $(SET_LIB_PATH) \  
  (cd ${dir}; $(MAKE) $(FLAGS_TO_PASS) clean); \  
else \  
  true; \  
fi
```

.PHONY: \$(CLEAN\_TARGET\_MODULES)

\$(CLEAN\_TARGET\_MODULES):

```
@dir=`echo $@ | sed -e 's/clean-target-//'^`; \  
rm -f $(TARGET_SUBDIR)/${dir}/multilib.out $(TARGET_SUBDIR)/${dir}/tmpmult
```