Installing GCC

For GCC version 10.2.0

(ncc-v1.0.11)

Copyright © 1988-2020 Free Software Foundation, Inc.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, the Front-Cover texts being (a) (see below), and with the Back-Cover Texts being (b) (see below). A copy of the license is included in the section entitled "GNU Free Documentation License".

(a) The FSF's Front-Cover Text is:

A GNU Manual

(b) The FSF's Back-Cover Text is:

You have freedom to copy and modify this GNU Manual, like GNU software. Copies published by the Free Software Foundation raise funds for GNU development.

Table of Contents

1 Installing GCC 1

2 Prerequisites 3

3 Downloading GCC 9

4 Installing GCC: Configuration 11

5 Building 43
- 5.1 Building a native compiler 43
- 5.2 Building a cross compiler 46
- 5.3 Building in parallel 47
- 5.4 Building the Ada compiler 47
- 5.5 Building with profile feedback 47

6 Installing GCC: Testing 49
- 6.1 How can you run the testsuite on selected tests? 49
- 6.2 Passing options and running multiple testsuites 50
- 6.3 How to interpret test results 51
- 6.4 Submitting test results 51

7 Installing GCC: Final installation 53

8 Installing GCC: Binaries 55

9 Host/target specific installation notes for GCC 57

10 Old installation documentation 77
- 10.1 Configurations Supported by GCC 77

GNU Free Documentation License 79
- ADDENDUM: How to use this License for your documents 86

1 Installing GCC

The latest version of this document is always available at http://gcc.gnu.org/install/. It refers to the current development sources, instructions for specific released versions are included with the sources.

This document describes the generic installation procedure for GCC as well as detailing some target specific installation instructions.

GCC includes several components that previously were separate distributions with their own installation instructions. This document supersedes all package-specific installation instructions.

Before starting the build/install procedure please check the Chapter 9 [Specific], page 57. We recommend you browse the entire generic installation instructions before you proceed.

Lists of successful builds for released versions of GCC are available at `http://gcc.gnu.org/buildstat.html`. These lists are updated as new information becomes available.

The installation procedure itself is broken into five steps.

Please note that GCC does not support '`make uninstall`' and probably won't do so in the near future as this would open a can of worms. Instead, we suggest that you install GCC into a directory of its own and simply remove that directory when you do not need that specific version of GCC any longer, and, if shared libraries are installed there as well, no more binaries exist that use them.

2 Prerequisites

GCC requires that various tools and packages be available for use in the build procedure. Modifying GCC sources requires additional tools described below.

Tools/packages necessary for building GCC

ISO C++98 compiler

Necessary to bootstrap GCC, although versions of GCC prior to 4.8 also allow bootstrapping with a ISO C89 compiler and versions of GCC prior to 3.4 also allow bootstrapping with a traditional (K&R) C compiler.

To build all languages in a cross-compiler or other configuration where 3-stage bootstrap is not performed, you need to start with an existing GCC binary (version 3.4 or later) because source code for language frontends other than C might use GCC extensions.

Note that to bootstrap GCC with versions of GCC earlier than 3.4, you may need to use '`--disable-stage1-checking`', though bootstrapping the compiler with such earlier compilers is strongly discouraged.

C standard library and headers

In order to build GCC, the C standard library and headers must be present for all target variants for which target libraries will be built (and not only the variant of the host C++ compiler).

This affects the popular '`x86_64-pc-linux-gnu`' platform (among other multilib targets), for which 64-bit ('`x86_64`') and 32-bit ('`i386`') libc headers are usually packaged separately. If you do a build of a native compiler on '`x86_64-pc-linux-gnu`', make sure you either have the 32-bit libc developer package properly installed (the exact name of the package depends on your distro) or you must build GCC as a 64-bit only compiler by configuring with the option '`--disable-multilib`'. Otherwise, you may encounter an error such as '`fatal error: gnu/stubs-32.h: No such file`'

GNAT

In order to build GNAT, the Ada compiler, you need a working GNAT compiler (GCC version 4.7 or later).

This includes GNAT tools such as `gnatmake` and `gnatlink`, since the Ada front end is written in Ada and uses some GNAT-specific extensions.

In order to build a cross compiler, it is strongly recommended to install the new compiler as native first, and then use it to build the cross compiler. Other native compiler versions may work but this is not guaranteed and will typically fail with hard to understand compilation errors during the build.

Similarly, it is strongly recommended to use an older version of GNAT to build GNAT. More recent versions of GNAT than the version built are not guaranteed to work and will often fail during the build with compilation errors.

Note that `configure` does not test whether the GNAT installation works and has a sufficiently recent version; if too old a GNAT version is installed and '`--enable-languages=ada`' is used, the build will fail.

`ADA_INCLUDE_PATH` and `ADA_OBJECT_PATH` environment variables must not be set when building the Ada compiler, the Ada tools, or the Ada runtime libraries. You can check that your build environment is clean by verifying that '`gnatls -v`' lists only one explicit path in each section.

A "working" POSIX compatible shell, or GNU bash
: Necessary when running `configure` because some `/bin/sh` shells have bugs and may crash when configuring the target libraries. In other cases, `/bin/sh` or `ksh` have disastrous corner-case performance problems. This can cause target `configure` runs to literally take days to complete in some cases.

  So on some platforms `/bin/ksh` is sufficient, on others it isn't. See the host/target specific instructions for your platform, or use `bash` to be sure. Then set `CONFIG_SHELL` in your environment to your "good" shell prior to running `configure`/`make`.

  `zsh` is not a fully compliant POSIX shell and will not work when configuring GCC.

A POSIX or SVR4 awk
: Necessary for creating some of the generated source files for GCC. If in doubt, use a recent GNU awk version, as some of the older ones are broken. GNU awk version 3.1.5 is known to work.

GNU binutils
: Necessary in some circumstances, optional in others. See the host/target specific instructions for your platform for the exact requirements.

gzip version 1.2.4 (or later) or
bzip2 version 1.0.2 (or later)
: Necessary to uncompress GCC `tar` files when source code is obtained via HTTPS mirror sites.

GNU make version 3.80 (or later)
: You must have GNU make installed to build GCC.

GNU tar version 1.14 (or later)
: Necessary (only on some platforms) to untar the source code. Many systems' `tar` programs will also work, only try GNU `tar` if you have problems.

Perl version between 5.6.1 and 5.6.24
: Necessary when targeting Darwin, building '`libstdc++`', and not using '`--disable-symvers`'. Necessary when targeting Solaris 2 with Solaris `ld` and not using '`--disable-symvers`'.

  Necessary when regenerating '`Makefile`' dependencies in libiberty. Necessary when regenerating '`libiberty/functions.texi`'. Necessary when generating manpages from Texinfo manuals. Used by various scripts to generate some files included in the source repository (mainly Unicode-related and rarely changing) from source tables.

  Used by `automake`.

Several support libraries are necessary to build GCC, some are required, others optional. While any sufficiently new version of required tools usually work, library requirements are

generally stricter. Newer versions may work in some cases, but it's safer to use the exact versions documented. We appreciate bug reports about problems with newer versions, though. If your OS vendor provides packages for the support libraries then using those packages may be the simplest way to install the libraries.

GNU Multiple Precision Library (GMP) version 4.3.2 (or later)
: Necessary to build GCC. If a GMP source distribution is found in a subdirectory of your GCC sources named 'gmp', it will be built together with GCC. Alternatively, if GMP is already installed but it is not in your library search path, you will have to configure with the '--with-gmp' configure option. See also '--with-gmp-lib' and '--with-gmp-include'. The in-tree build is only supported with the GMP version that download_prerequisites installs.

MPFR Library version 3.1.0 (or later)
: Necessary to build GCC. It can be downloaded from https://www.mpfr.org. If an MPFR source distribution is found in a subdirectory of your GCC sources named 'mpfr', it will be built together with GCC. Alternatively, if MPFR is already installed but it is not in your default library search path, the '--with-mpfr' configure option should be used. See also '--with-mpfr-lib' and '--with-mpfr-include'. The in-tree build is only supported with the MPFR version that download_prerequisites installs.

MPC Library version 1.0.1 (or later)
: Necessary to build GCC. It can be downloaded from http://www.multiprecision.org/mpc/. If an MPC source distribution is found in a subdirectory of your GCC sources named 'mpc', it will be built together with GCC. Alternatively, if MPC is already installed but it is not in your default library search path, the '--with-mpc' configure option should be used. See also '--with-mpc-lib' and '--with-mpc-include'. The in-tree build is only supported with the MPC version that download_prerequisites installs.

isl Library version 0.15 or later.
: Necessary to build GCC with the Graphite loop optimizations. It can be downloaded from https://gcc.gnu.org/pub/gcc/infrastructure/. If an isl source distribution is found in a subdirectory of your GCC sources named 'isl', it will be built together with GCC. Alternatively, the '--with-isl' configure option should be used if isl is not installed in your default library search path.

zstd Library.
: Necessary to build GCC with zstd compression used for LTO bytecode. The library is searched in your default library patch search. Alternatively, the '--with-zstd' configure option should be used.

Tools/packages necessary for modifying GCC

autoconf version 2.69
GNU m4 version 1.4.6 (or later)
: Necessary when modifying 'configure.ac', 'aclocal.m4', etc. to regenerate 'configure' and 'config.in' files.

automake version 1.15.1

Necessary when modifying a '`Makefile.am`' file to regenerate its associated '`Makefile.in`'.

Much of GCC does not use automake, so directly edit the '`Makefile.in`' file. Specifically this applies to the '`gcc`', '`intl`', '`libcpp`', '`libiberty`', '`libobjc`' directories as well as any of their subdirectories.

For directories that use automake, GCC requires the latest release in the 1.15 series, which is currently 1.15.1. When regenerating a directory to a newer version, please update all the directories using an older 1.15 to the latest released version.

gettext version 0.14.5 (or later)

Needed to regenerate '`gcc.pot`'.

gperf version 2.7.2 (or later)

Necessary when modifying `gperf` input files, e.g. '`gcc/cp/cfns.gperf`' to regenerate its associated header file, e.g. '`gcc/cp/cfns.h`'.

DejaGnu 1.4.4
Expect
Tcl

Necessary to run the GCC testsuite; see the section on testing for details.

autogen version 5.5.4 (or later) and
guile version 1.4.1 (or later)

Necessary to regenerate '`fixinc/fixincl.x`' from '`fixinc/inclhack.def`' and '`fixinc/*.tpl`'.

Necessary to run '`make check`' for '`fixinc`'.

Necessary to regenerate the top level '`Makefile.in`' file from '`Makefile.tpl`' and '`Makefile.def`'.

Flex version 2.5.4 (or later)

Necessary when modifying '`*.l`' files.

Necessary to build GCC during development because the generated output files are not included in the version-controlled source repository. They are included in releases.

Texinfo version 4.7 (or later)

Necessary for running `makeinfo` when modifying '`*.texi`' files to test your changes.

Necessary for running `make dvi` or `make pdf` to create printable documentation in DVI or PDF format. Texinfo version 4.8 or later is required for `make pdf`.

Necessary to build GCC documentation during development because the generated output files are not included in the repository. They are included in releases.

TeX (any working version)

Necessary for running `texi2dvi` and `texi2pdf`, which are used when running `make dvi` or `make pdf` to create DVI or PDF files, respectively.

Sphinx version 1.0 (or later)
: Necessary to regenerate '`jit/docs/_build/texinfo`' from the '`.rst`' files in the directories below '`jit/docs`'.

git (any version)
SSH (any version)
: Necessary to access the source repository. Public releases and weekly snapshots of the development sources are also available via HTTPS.

GNU diffutils version 2.7 (or later)
: Useful when submitting patches for the GCC source code.

patch version 2.5.4 (or later)
: Necessary when applying patches, created with `diff`, to one's own sources.

3 Downloading GCC

GCC is distributed via git and via HTTPS as tarballs compressed with `gzip` or `bzip2`.

Please refer to the releases web page for information on how to obtain GCC.

The source distribution includes the C, C++, Objective-C, Fortran, and Ada (in the case of GCC 3.1 and later) compilers, as well as runtime libraries for C++, Objective-C, and Fortran. For previous versions these were downloadable as separate components such as the core GCC distribution, which included the C language front end and shared components, and language-specific distributions including the language front end and the language runtime (where appropriate).

If you also intend to build binutils (either to upgrade an existing installation or for use in place of the corresponding tools of your OS), unpack the binutils distribution either in the same directory or a separate one. In the latter case, add symbolic links to any components of the binutils you intend to build alongside the compiler (‘`bfd`’, ‘`binutils`’, ‘`gas`’, ‘`gprof`’, ‘`ld`’, ‘`opcodes`’, . . .) to the directory containing the GCC sources.

Likewise the GMP, MPFR and MPC libraries can be automatically built together with GCC. You may simply run the `contrib/download_prerequisites` script in the GCC source directory to set up everything. Otherwise unpack the GMP, MPFR and/or MPC source distributions in the directory containing the GCC sources and rename their directories to ‘`gmp`’, ‘`mpfr`’ and ‘`mpc`’, respectively (or use symbolic links with the same name).

4 Installing GCC: Configuration

Like most GNU software, GCC must be configured before it can be built. This document describes the recommended configuration procedure for both native and cross targets.

We use *srcdir* to refer to the toplevel source directory for GCC; we use *objdir* to refer to the toplevel build/object directory.

If you obtained the sources by cloning the repository, *srcdir* must refer to the top '`gcc`' directory, the one where the '`MAINTAINERS`' file can be found, and not its '`gcc`' subdirectory, otherwise the build will fail.

If either *srcdir* or *objdir* is located on an automounted NFS file system, the shell's built-in `pwd` command will return temporary pathnames. Using these can lead to various sorts of build problems. To avoid this issue, set the `PWDCMD` environment variable to an automounter-aware `pwd` command, e.g., `pawd` or '`amq -w`', during the configuration and build phases.

First, we **highly** recommend that GCC be built into a separate directory from the sources which does **not** reside within the source tree. This is how we generally build GCC; building where *srcdir* == *objdir* should still work, but doesn't get extensive testing; building where *objdir* is a subdirectory of *srcdir* is unsupported.

If you have previously built GCC in the same directory for a different target machine, do '`make distclean`' to delete all files that might be invalid. One of the files this deletes is '`Makefile`'; if '`make distclean`' complains that '`Makefile`' does not exist or issues a message like "don't know how to make distclean" it probably means that the directory is already suitably clean. However, with the recommended method of building in a separate *objdir*, you should simply use a different *objdir* for each target.

Second, when configuring a native system, either `cc` or `gcc` must be in your path or you must set `CC` in your environment before running configure. Otherwise the configuration scripts may fail.

To configure GCC:

```
% mkdir objdir
% cd objdir
% srcdir/configure [options] [target]
```

Distributor options

If you will be distributing binary versions of GCC, with modifications to the source code, you should use the options described in this section to make clear that your version contains modifications.

`--with-pkgversion=`*version*

> Specify a string that identifies your package. You may wish to include a build number or build date. This version string will be included in the output of `gcc --version`. This suffix does not replace the default version string, only the '`GCC`' part.
>
> The default value is '`GCC`'.

`--with-bugurl=`*url*
: Specify the URL that users should visit if they wish to report a bug. You are of course welcome to forward bugs reported to you to the FSF, if you determine that they are not bugs in your modifications.

  The default value refers to the FSF's GCC bug tracker.

`--with-documentation-root-url=`*url*
: Specify the URL root that contains GCC option documentation. The *url* should end with a `/` character.

  The default value is https://gcc.gnu.org/onlinedocs/.

`--with-changes-root-url=`*url*
: Specify the URL root that contains information about changes in GCC releases like `gcc-`*version*`/changes.html`. The *url* should end with a `/` character.

  The default value is https://gcc.gnu.org/.

Target specification

- GCC has code to correctly determine the correct value for *target* for nearly all native systems. Therefore, we highly recommend you do not provide a configure target when configuring a native compiler.
- *target* must be specified as '`--target=`*target*' when configuring a cross compiler; examples of valid targets would be m68k-elf, sh-elf, etc.
- Specifying just *target* instead of '`--target=`*target*' implies that the host defaults to *target*.

Options specification

Use *options* to override several configure time options for GCC. A list of supported *options* follows; '`configure --help`' may list other options, but those not listed below may not work and should not normally be used.

Note that each '`--enable`' option has a corresponding '`--disable`' option and that each '`--with`' option has a corresponding '`--without`' option.

`--prefix=`*dirname*
: Specify the toplevel installation directory. This is the recommended way to install the tools into a directory other than the default. The toplevel installation directory defaults to '`/usr/local`'.

  We **highly** recommend against *dirname* being the same or a subdirectory of *objdir* or vice versa. If specifying a directory beneath a user's home directory tree, some shells will not expand *dirname* correctly if it contains the '`~`' metacharacter; use `$HOME` instead.

  The following standard `autoconf` options are supported. Normally you should not need to use these options.

  `--exec-prefix=`*dirname*
  : Specify the toplevel installation directory for architecture-dependent files. The default is '*prefix*'.

`--bindir=`*dirname*
: Specify the installation directory for the executables called by users (such as `gcc` and `g++`). The default is '*exec-prefix*`/bin`'.

`--libdir=`*dirname*
: Specify the installation directory for object code libraries and internal data files of GCC. The default is '*exec-prefix*`/lib`'.

`--libexecdir=`*dirname*
: Specify the installation directory for internal executables of GCC. The default is '*exec-prefix*`/libexec`'.

`--with-slibdir=`*dirname*
: Specify the installation directory for the shared libgcc library. The default is '*libdir*'.

`--datarootdir=`*dirname*
: Specify the root of the directory tree for read-only architecture-independent data files referenced by GCC. The default is '*prefix*`/share`'.

`--infodir=`*dirname*
: Specify the installation directory for documentation in info format. The default is '*datarootdir*`/info`'.

`--datadir=`*dirname*
: Specify the installation directory for some architecture-independent data files referenced by GCC. The default is '*datarootdir*'.

`--docdir=`*dirname*
: Specify the installation directory for documentation files (other than Info) for GCC. The default is '*datarootdir*`/doc`'.

`--htmldir=`*dirname*
: Specify the installation directory for HTML documentation files. The default is '*docdir*'.

`--pdfdir=`*dirname*
: Specify the installation directory for PDF documentation files. The default is '*docdir*'.

`--mandir=`*dirname*
: Specify the installation directory for manual pages. The default is '*datarootdir*`/man`'. (Note that the manual pages are only extracts from the full GCC manuals, which are provided in Texinfo format. The manpages are derived by an automatic conversion process from parts of the full manual.)

`--with-gxx-include-dir=`*dirname*
: Specify the installation directory for G++ header files. The default depends on other configuration options, and differs between cross and native configurations.

`--with-specs=`*specs*
: Specify additional command line driver SPECS. This can be useful if you need to turn on a non-standard feature by default without modifying the compiler's source code, for instance '`--with-specs=%{!fcommon:%{!fno-common:-fno-common}}`'.
See Section "Specifying subprocesses and the switches to pass to them" in *Using the GNU Compiler Collection (GCC)*,

`--program-prefix=`*prefix*
: GCC supports some transformations of the names of its programs when installing them. This option prepends *prefix* to the names of programs to install in *bindir* (see above). For example, specifying '`--program-prefix=foo-`' would result in '`gcc`' being installed as '`/usr/local/bin/foo-gcc`'.

`--program-suffix=`*suffix*
: Appends *suffix* to the names of programs to install in *bindir* (see above). For example, specifying '`--program-suffix=-3.1`' would result in '`gcc`' being installed as '`/usr/local/bin/gcc-3.1`'.

`--program-transform-name=`*pattern*
: Applies the '`sed`' script *pattern* to be applied to the names of programs to install in *bindir* (see above). *pattern* has to consist of one or more basic '`sed`' editing commands, separated by semicolons. For example, if you want the '`gcc`' program name to be transformed to the installed program '`/usr/local/bin/myowngcc`' and the '`g++`' program name to be transformed to '`/usr/local/bin/gspecial++`' without changing other program names, you could use the pattern '`--program-transform-name='s/^gcc$/myowngcc/; s/^g++$/gspecial++/''` to achieve this effect.

  All three options can be combined and used together, resulting in more complex conversion patterns. As a basic rule, *prefix* (and *suffix*) are prepended (appended) before further transformations can happen with a special transformation script *pattern*.

  As currently implemented, this option only takes effect for native builds; cross compiler binaries' names are not transformed even when a transformation is explicitly asked for by one of these options.

  For native builds, some of the installed programs are also installed with the target alias in front of their name, as in '`i686-pc-linux-gnu-gcc`'. All of the above transformations happen before the target alias is prepended to the name—so, specifying '`--program-prefix=foo-`' and '`program-suffix=-3.1`', the resulting binary would be installed as '`/usr/local/bin/i686-pc-linux-gnu-foo-gcc-3.1`'.

  As a last shortcoming, none of the installed Ada programs are transformed yet, which will be fixed in some time.

`--with-local-prefix=`*dirname*
: Specify the installation directory for local include files. The default is '`/usr/local`'. Specify this option if you want the compiler to search directory '*dirname*`/include`' for locally installed header files *instead* of '`/usr/local/include`'.

You should specify '`--with-local-prefix`' **only** if your site has a different convention (not '`/usr/local`') for where to put site-specific files.

The default value for '`--with-local-prefix`' is '`/usr/local`' regardless of the value of '`--prefix`'. Specifying '`--prefix`' has no effect on which directory GCC searches for local header files. This may seem counterintuitive, but actually it is logical.

The purpose of '`--prefix`' is to specify where to *install GCC*. The local header files in '`/usr/local/include`'—if you put any in that directory—are not part of GCC. They are part of other programs—perhaps many others. (GCC installs its own header files in another directory which is based on the '`--prefix`' value.)

Both the local-prefix include directory and the GCC-prefix include directory are part of GCC's "system include" directories. Although these two directories are not fixed, they need to be searched in the proper order for the correct processing of the include_next directive. The local-prefix include directory is searched before the GCC-prefix include directory. Another characteristic of system include directories is that pedantic warnings are turned off for headers in these directories.

Some autoconf macros add '`-I` *`directory`*' options to the compiler command line, to ensure that directories containing installed packages' headers are searched. When *directory* is one of GCC's system include directories, GCC will ignore the option so that system directories continue to be processed in the correct order. This may result in a search order different from what was specified but the directory will still be searched.

GCC automatically searches for ordinary libraries using `GCC_EXEC_PREFIX`. Thus, when the same installation prefix is used for both GCC and packages, GCC will automatically search for both headers and libraries. This provides a configuration that is easy to use. GCC behaves in a manner similar to that when it is installed as a system compiler in '`/usr`'.

Sites that need to install multiple versions of GCC may not want to use the above simple configuration. It is possible to use the '`--program-prefix`', '`--program-suffix`' and '`--program-transform-name`' options to install multiple versions into a single directory, but it may be simpler to use different prefixes and the '`--with-local-prefix`' option to specify the location of the site-specific files for each version. It will then be necessary for users to specify explicitly the location of local site libraries (e.g., with `LIBRARY_PATH`).

The same value can be used for both '`--with-local-prefix`' and '`--prefix`' provided it is not '`/usr`'. This can be used to avoid the default search of '`/usr/local/include`'.

Do not specify '`/usr`' as the '`--with-local-prefix`'! The directory you use for '`--with-local-prefix`' **must not** contain any of the system's standard header files. If it did contain them, certain programs would be miscompiled (including GNU Emacs, on certain targets), because this would override and nullify the header file corrections made by the `fixincludes` script.

Indications are that people who use this option use it based on mistaken ideas of what it is for. People use it as if it specified where to install part of GCC.

Perhaps they make this assumption because installing GCC creates the directory.

`--with-gcc-major-version-only`
: Specifies that GCC should use only the major number rather than *major.minor.patchlevel* in filesystem paths.

`--with-native-system-header-dir=`*dirname*
: Specifies that *dirname* is the directory that contains native system header files, rather than '`/usr/include`'. This option is most useful if you are creating a compiler that should be isolated from the system as much as possible. It is most commonly used with the '`--with-sysroot`' option and will cause GCC to search *dirname* inside the system root specified by that option.

`--enable-shared[=`*package*`[,...]]`
: Build shared versions of libraries, if shared libraries are supported on the target platform. Unlike GCC 2.95.x and earlier, shared libraries are enabled by default on all platforms that support shared libraries.

  If a list of packages is given as an argument, build shared libraries only for the listed packages. For other packages, only static libraries will be built. Package names currently recognized in the GCC tree are '`libgcc`' (also known as '`gcc`'), '`libstdc++`' (not '`libstdc++-v3`'), '`libffi`', '`zlib`', '`boehm-gc`', '`ada`', '`libada`', '`libgo`', '`libobjc`', and '`libphobos`'. Note '`libiberty`' does not support shared libraries at all.

  Use '`--disable-shared`' to build only static libraries. Note that '`--disable-shared`' does not accept a list of package names as argument, only '`--enable-shared`' does.

  Contrast with '`--enable-host-shared`', which affects *host* code.

`--enable-host-shared`
: Specify that the *host* code should be built into position-independent machine code (with -fPIC), allowing it to be used within shared libraries, but yielding a slightly slower compiler.

  This option is required when building the libgccjit.so library.

  Contrast with '`--enable-shared`', which affects *target* libraries.

`--with-gnu-as`
: Specify that the compiler should assume that the assembler it finds is the GNU assembler. However, this does not modify the rules to find an assembler and will result in confusion if the assembler found is not actually the GNU assembler. (Confusion may also result if the compiler finds the GNU assembler but has not been configured with '`--with-gnu-as`'.) If you have more than one assembler installed on your system, you may want to use this option in connection with '`--with-as=`*pathname*' or '`--with-build-time-tools=`*pathname*'.

  The following systems are the only ones where it makes a difference whether you use the GNU assembler. On any other system, '`--with-gnu-as`' has no effect.

  - '`hppa1.0-`*any*`-`*any*'

- ‘hppa1.1-*any*-*any*’
- ‘sparc-sun-solaris2.*any*’
- ‘sparc64-*any*-solaris2.*any*’

`--with-as=`*pathname*
: Specify that the compiler should use the assembler pointed to by *pathname*, rather than the one found by the standard rules to find an assembler, which are:

  - Unless GCC is being built with a cross compiler, check the ‘*libexec*/gcc/*target*/*version*’ directory. *libexec* defaults to ‘*exec-prefix*/libexec’; *exec-prefix* defaults to *prefix*, which defaults to ‘/usr/local’ unless overridden by the ‘`--prefix=`*pathname*’ switch described above. *target* is the target system triple, such as ‘sparc-sun-solaris2.7’, and *version* denotes the GCC version, such as 3.0.
  - If the target system is the same that you are building on, check operating system specific directories (e.g. ‘/usr/ccs/bin’ on Solaris 2).
  - Check in the `PATH` for a tool whose name is prefixed by the target system triple.
  - Check in the `PATH` for a tool whose name is not prefixed by the target system triple, if the host and target system triple are the same (in other words, we use a host tool if it can be used for the target as well).

  You may want to use ‘`--with-as`’ if no assembler is installed in the directories listed above, or if you have multiple assemblers installed and want to choose one that is not found by the above rules.

`--with-gnu-ld`
: Same as ‘`--with-gnu-as`’ but for the linker.

`--with-ld=`*pathname*
: Same as ‘`--with-as`’ but for the linker.

`--with-stabs`
: Specify that stabs debugging information should be used instead of whatever format the host normally uses. Normally GCC uses the same debug format as the host system.

`--with-tls=`*dialect*
: Specify the default TLS dialect, for systems were there is a choice. For ARM targets, possible values for *dialect* are `gnu` or `gnu2`, which select between the original GNU dialect and the GNU TLS descriptor-based dialect.

`--enable-multiarch`
: Specify whether to enable or disable multiarch support. The default is to check for glibc start files in a multiarch location, and enable it if the files are found. The auto detection is enabled for native builds, and for cross builds configured with ‘`--with-sysroot`’, and without ‘`--with-native-system-header-dir`’. More documentation about multiarch can be found at `https://wiki.debian.org/Multiarch`.

`--enable-sjlj-exceptions`
: Force use of the `setjmp/longjmp`-based scheme for exceptions. '`configure`' ordinarily picks the correct value based on the platform. Only use this option if you are sure you need a different setting.

`--enable-vtable-verify`
: Specify whether to enable or disable the vtable verification feature. Enabling this feature causes libstdc++ to be built with its virtual calls in verifiable mode. This means that, when linked with libvtv, every virtual call in libstdc++ will verify the vtable pointer through which the call will be made before actually making the call. If not linked with libvtv, the verifier will call stub functions (in libstdc++ itself) and do nothing. If vtable verification is disabled, then libstdc++ is not built with its virtual calls in verifiable mode at all. However the libvtv library will still be built (see '`--disable-libvtv`' to turn off building libvtv). '`--disable-vtable-verify`' is the default.

`--disable-gcov`
: Specify that the run-time library used for coverage analysis and associated host tools should not be built.

`--disable-multilib`
: Specify that multiple target libraries to support different target variants, calling conventions, etc. should not be built. The default is to build a predefined set of them.

  Some targets provide finer-grained control over which multilibs are built (e.g., '`--disable-softfloat`'):

  `arm-*-*` fpu, 26bit, underscore, interwork, biendian, nofmult.

  `m68*-*-*` softfloat, m68881, m68000, m68020.

  `mips*-*-*`
  : single-float, biendian, softfloat.

  `msp430-*-*`
  : no-exceptions

  `powerpc*-*-*, rs6000*-*-*`
  : aix64, pthread, softfloat, powercpu, powerpccpu, powerpcos, biendian, sysv, aix.

`--with-multilib-list=`*list*
`--without-multilib-list`
: Specify what multilibs to build. *list* is a comma separated list of values, possibly consisting of a single value. Currently only implemented for aarch64*-*-*, arm*-*-*, riscv*-*-*, sh*-*-* and x86-64-*-linux*. The accepted values and meaning for each target is given below.

  `aarch64*-*-*`
  : *list* is a comma separated list of `ilp32`, and `lp64` to enable ILP32 and LP64 run-time libraries, respectively. If *list* is empty, then there will be no multilibs and only the default run-time library will

be built. If *list* is `default` or –with-multilib-list= is not specified, then the default set of libraries is selected based on the value of '`--target`'.

`arm*-*-*` *list* is a comma separated list of `aprofile` and `rmprofile` to build multilibs for A or R and M architecture profiles respectively. Note that, due to some limitation of the current multilib framework, using the combined `aprofile,rmprofile` multilibs selects in some cases a less optimal multilib than when using the multilib profile for the architecture targetted. The special value `default` is also accepted and is equivalent to omitting the option, i.e., only the default run-time library will be enabled.

list may instead contain `@name`, to use the multilib configuration Makefile fragment '`name`' in '`gcc/config/arm`' in the source tree (it is part of the corresponding sources, after all). It is recommended, but not required, that files used for this purpose to be named starting with '`t-ml-`', to make their intended purpose self-evident, in line with GCC conventions. Such files enable custom, user-chosen multilib lists to be configured. Whether multiple such files can be used together depends on the contents of the supplied files. See '`gcc/config/arm/t-multilib`' and its supplementary '`gcc/config/arm/t-*profile`' files for an example of what such Makefile fragments might look like for this version of GCC. The macros expected to be defined in these fragments are not stable across GCC releases, so make sure they define the `MULTILIB`-related macros expected by the version of GCC you are building. See Section "Target Makefile Fragments" in *GNU Compiler Collection (GCC) Internals*.

The table below gives the combination of ISAs, architectures, FPUs and floating-point ABIs for which multilibs are built for each predefined profile. The union of these options is considered when specifying both `aprofile` and `rmprofile`.

Option	aprofile	rmprofile
ISAs	`-marm` and `-mthumb`	`-mthumb`
Architectures	default architecture	default architecture
	`-march=armv7-a`	`-march=armv6s-m`
	`-march=armv7ve`	`-march=armv7-m`
	`-march=armv8-a`	`-march=armv7e-m`
		`-march=armv8-m.base`
		`-march=armv8-m.main`
		`-march=armv7`
FPUs	none	none
	`-mfpu=vfpv3-d16`	`-mfpu=vfpv3-d16`
	`-mfpu=neon`	`-mfpu=fpv4-sp-d16`
	`-mfpu=vfpv4-d16`	`-mfpu=fpv5-sp-d16`
	`-mfpu=neon-vfpv4`	`-mfpu=fpv5-d16`
	`-mfpu=neon-fp-armv8`	

floating-point ABIs	`-mfloat-abi=soft`	`-mfloat-abi=soft`
	`-mfloat-abi=softfp`	`-mfloat-abi=softfp`
	`-mfloat-abi=hard`	`-mfloat-abi=hard`

`riscv*-*-*`

list is a single ABI name. The target architecture must be either `rv32gc` or `rv64gc`. This will build a single multilib for the specified architecture and ABI pair. If `--with-multilib-list` is not given, then a default set of multilibs is selected based on the value of '`--target`'. This is usually a large set of multilibs.

`sh*-*-*` *list* is a comma separated list of CPU names. These must be of the form `sh*` or `m*` (in which case they match the compiler option for that processor). The list should not contain any endian options - these are handled by '`--with-endian`'.

If *list* is empty, then there will be no multilibs for extra processors. The multilib for the secondary endian remains enabled.

As a special case, if an entry in the list starts with a `!` (exclamation point), then it is added to the list of excluded multilibs. Entries of this sort should be compatible with '`MULTILIB_EXCLUDES`' (once the leading `!` has been stripped).

If '`--with-multilib-list`' is not given, then a default set of multilibs is selected based on the value of '`--target`'. This is usually the complete set of libraries, but some targets imply a more specialized subset.

Example 1: to configure a compiler for SH4A only, but supporting both endians, with little endian being the default:

```
--with-cpu=sh4a --with-endian=little,big --with-multilib-list=
```

Example 2: to configure a compiler for both SH4A and SH4AL-DSP, but with only little endian SH4AL:

```
--with-cpu=sh4a --with-endian=little,big \
--with-multilib-list=sh4al,!mb/m4al
```

`x86-64-*-linux*`

list is a comma separated list of `m32`, `m64` and `mx32` to enable 32-bit, 64-bit and x32 run-time libraries, respectively. If *list* is empty, then there will be no multilibs and only the default run-time library will be enabled.

If '`--with-multilib-list`' is not given, then only 32-bit and 64-bit run-time libraries will be enabled.

`--with-endian=`*endians*

Specify what endians to use. Currently only implemented for sh*-*-*.

endians may be one of the following:

`big` Use big endian exclusively.

`little` Use little endian exclusively.

`big,little`
: Use big endian by default. Provide a multilib for little endian.

`little,big`
: Use little endian by default. Provide a multilib for big endian.

`--enable-threads`
: Specify that the target supports threads. This affects the Objective-C compiler and runtime library, and exception handling for other languages like C++. On some systems, this is the default.

  In general, the best (and, in many cases, the only known) threading model available will be configured for use. Beware that on some systems, GCC has not been taught what threading models are generally available for the system. In this case, '`--enable-threads`' is an alias for '`--enable-threads=single`'.

`--disable-threads`
: Specify that threading support should be disabled for the system. This is an alias for '`--enable-threads=single`'.

`--enable-threads=`*lib*
: Specify that *lib* is the thread support library. This affects the Objective-C compiler and runtime library, and exception handling for other languages like C++. The possibilities for *lib* are:

  `aix` AIX thread support.

  `dce` DCE thread support.

  `lynx` LynxOS thread support.

  `mipssde` MIPS SDE thread support.

  `no` This is an alias for '`single`'.

  `posix` Generic POSIX/Unix98 thread support.

  `rtems` RTEMS thread support.

  `single` Disable thread support, should work for all platforms.

  `tpf` TPF thread support.

  `vxworks` VxWorks thread support.

  `win32` Microsoft Win32 API thread support.

`--enable-tls`
: Specify that the target supports TLS (Thread Local Storage). Usually configure can correctly determine if TLS is supported. In cases where it guesses incorrectly, TLS can be explicitly enabled or disabled with '`--enable-tls`' or '`--disable-tls`'. This can happen if the assembler supports TLS but the C library does not, or if the assumptions made by the configure test are incorrect.

`--disable-tls`
: Specify that the target does not support TLS. This is an alias for '`--enable-tls=no`'.

`--disable-tm-clone-registry`
: Disable TM clone registry in libgcc. It is enabled in libgcc by default. This option helps to reduce code size for embedded targets which do not use transactional memory.

`--with-cpu=`*cpu*
`--with-cpu-32=`*cpu*
`--with-cpu-64=`*cpu*
: Specify which cpu variant the compiler should generate code for by default. *cpu* will be used as the default value of the '`-mcpu=`' switch. This option is only supported on some targets, including ARC, ARM, i386, M68k, PowerPC, and SPARC. It is mandatory for ARC. The '`--with-cpu-32`' and '`--with-cpu-64`' options specify separate default CPUs for 32-bit and 64-bit modes; these options are only supported for i386, x86-64, PowerPC, and SPARC.

`--with-schedule=`*cpu*
`--with-arch=`*cpu*
`--with-arch-32=`*cpu*
`--with-arch-64=`*cpu*
`--with-tune=`*cpu*
`--with-tune-32=`*cpu*
`--with-tune-64=`*cpu*
`--with-abi=`*abi*
`--with-fpu=`*type*
`--with-float=`*type*
: These configure options provide default values for the '`-mschedule=`', '`-march=`', '`-mtune=`', '`-mabi=`', and '`-mfpu=`' options and for '`-mhard-float`' or '`-msoft-float`'. As with '`--with-cpu`', which switches will be accepted and acceptable values of the arguments depend on the target.

`--with-mode=`*mode*
: Specify if the compiler should default to '`-marm`' or '`-mthumb`'. This option is only supported on ARM targets.

`--with-stack-offset=`*num*
: This option sets the default for the -mstack-offset=*num* option, and will thus generally also control the setting of this option for libraries. This option is only supported on Epiphany targets.

`--with-fpmath=`*isa*
: This options sets '`-mfpmath=sse`' by default and specifies the default ISA for floating-point arithmetics. You can select either '`sse`' which enables '`-msse2`' or '`avx`' which enables '`-mavx`' by default. This option is only supported on i386 and x86-64 targets.

`--with-fp-32=`*mode*
: On MIPS targets, set the default value for the '`-mfp`' option when using the o32 ABI. The possibilities for *mode* are:

  `32` Use the o32 FP32 ABI extension, as with the '`-mfp32`' command-line option.

`xx` Use the o32 FPXX ABI extension, as with the '`-mfpxx`' command-line option.

`64` Use the o32 FP64 ABI extension, as with the '`-mfp64`' command-line option.

In the absence of this configuration option the default is to use the o32 FP32 ABI extension.

`--with-odd-spreg-32`
On MIPS targets, set the '`-modd-spreg`' option by default when using the o32 ABI.

`--without-odd-spreg-32`
On MIPS targets, set the '`-mno-odd-spreg`' option by default when using the o32 ABI. This is normally used in conjunction with '`--with-fp-32=64`' in order to target the o32 FP64A ABI extension.

`--with-nan=`*encoding*
On MIPS targets, set the default encoding convention to use for the special not-a-number (NaN) IEEE 754 floating-point data. The possibilities for *encoding* are:

`legacy` Use the legacy encoding, as with the '`-mnan=legacy`' command-line option.

`2008` Use the 754-2008 encoding, as with the '`-mnan=2008`' command-line option.

To use this configuration option you must have an assembler version installed that supports the '`-mnan=`' command-line option too. In the absence of this configuration option the default convention is the legacy encoding, as when neither of the '`-mnan=2008`' and '`-mnan=legacy`' command-line options has been used.

`--with-divide=`*type*
Specify how the compiler should generate code for checking for division by zero. This option is only supported on the MIPS target. The possibilities for *type* are:

`traps` Division by zero checks use conditional traps (this is the default on systems that support conditional traps).

`breaks` Division by zero checks use the break instruction.

`--with-llsc`
On MIPS targets, make '`-mllsc`' the default when no '`-mno-llsc`' option is passed. This is the default for Linux-based targets, as the kernel will emulate them if the ISA does not provide them.

`--without-llsc`
On MIPS targets, make '`-mno-llsc`' the default when no '`-mllsc`' option is passed.

`--with-synci`
: On MIPS targets, make '`-msynci`' the default when no '`-mno-synci`' option is passed.

`--without-synci`
: On MIPS targets, make '`-mno-synci`' the default when no '`-msynci`' option is passed. This is the default.

`--with-lxc1-sxc1`
: On MIPS targets, make '`-mlxc1-sxc1`' the default when no '`-mno-lxc1-sxc1`' option is passed. This is the default.

`--without-lxc1-sxc1`
: On MIPS targets, make '`-mno-lxc1-sxc1`' the default when no '`-mlxc1-sxc1`' option is passed. The indexed load/store instructions are not directly a problem but can lead to unexpected behaviour when deployed in an application intended for a 32-bit address space but run on a 64-bit processor. The issue is seen because all known MIPS 64-bit Linux kernels execute o32 and n32 applications with 64-bit addressing enabled which affects the overflow behaviour of the indexed addressing mode. GCC will assume that ordinary 32-bit arithmetic overflow behaviour is the same whether performed as an `addu` instruction or as part of the address calculation in `lwxc1` type instructions. This assumption holds true in a pure 32-bit environment and can hold true in a 64-bit environment if the address space is accurately set to be 32-bit for o32 and n32.

`--with-madd4`
: On MIPS targets, make '`-mmadd4`' the default when no '`-mno-madd4`' option is passed. This is the default.

`--without-madd4`
: On MIPS targets, make '`-mno-madd4`' the default when no '`-mmadd4`' option is passed. The `madd4` instruction family can be problematic when targeting a combination of cores that implement these instructions differently. There are two known cores that implement these as fused operations instead of unfused (where unfused is normally expected). Disabling these instructions is the only way to ensure compatible code is generated; this will incur a performance penalty.

`--with-mips-plt`
: On MIPS targets, make use of copy relocations and PLTs. These features are extensions to the traditional SVR4-based MIPS ABIs and require support from GNU binutils and the runtime C library.

`--with-stack-clash-protection-guard-size=`*size*
: On certain targets this option sets the default stack clash protection guard size as a power of two in bytes. On AArch64 *size* is required to be either 12 (4KB) or 16 (64KB).

`--enable-__cxa_atexit`
: Define if you want to use __cxa_atexit, rather than atexit, to register C++ destructors for local statics and global objects. This is essential for fully standards-compliant handling of destructors, but requires __cxa_atexit in libc. This option

is currently only available on systems with GNU libc. When enabled, this will cause '`-fuse-cxa-atexit`' to be passed by default.

`--enable-gnu-indirect-function`
: Define if you want to enable the `ifunc` attribute. This option is currently only available on systems with GNU libc on certain targets.

`--enable-target-optspace`
: Specify that target libraries should be optimized for code space instead of code speed. This is the default for the m32r platform.

`--with-cpp-install-dir=`*dirname*
: Specify that the user visible `cpp` program should be installed in '*prefix*`/`*dirname*`/cpp`', in addition to *bindir*.

`--enable-comdat`
: Enable COMDAT group support. This is primarily used to override the automatically detected value.

`--enable-initfini-array`
: Force the use of sections `.init_array` and `.fini_array` (instead of `.init` and `.fini`) for constructors and destructors. Option '`--disable-initfini-array`' has the opposite effect. If neither option is specified, the configure script will try to guess whether the `.init_array` and `.fini_array` sections are supported and, if they are, use them.

`--enable-link-mutex`
: When building GCC, use a mutex to avoid linking the compilers for multiple languages at the same time, to avoid thrashing on build systems with limited free memory. The default is not to use such a mutex.

`--enable-maintainer-mode`
: The build rules that regenerate the Autoconf and Automake output files as well as the GCC master message catalog '`gcc.pot`' are normally disabled. This is because it can only be rebuilt if the complete source tree is present. If you have changed the sources and want to rebuild the catalog, configuring with '`--enable-maintainer-mode`' will enable this. Note that you need a recent version of the `gettext` tools to do so.

`--disable-bootstrap`
: For a native build, the default configuration is to perform a 3-stage bootstrap of the compiler when '`make`' is invoked, testing that GCC can compile itself correctly. If you want to disable this process, you can configure with '`--disable-bootstrap`'.

`--enable-bootstrap`
: In special cases, you may want to perform a 3-stage build even if the target and host triplets are different. This is possible when the host can run code compiled for the target (e.g. host is i686-linux, target is i486-linux). Starting from GCC 4.2, to do this you have to configure explicitly with '`--enable-bootstrap`'.

`--enable-generated-files-in-srcdir`

Neither the .c and .h files that are generated from Bison and flex nor the info manuals and man pages that are built from the .texi files are present in the repository development tree. When building GCC from that development tree, or from one of our snapshots, those generated files are placed in your build directory, which allows for the source to be in a readonly directory.

If you configure with '`--enable-generated-files-in-srcdir`' then those generated files will go into the source directory. This is mainly intended for generating release or prerelease tarballs of the GCC sources, since it is not a requirement that the users of source releases to have flex, Bison, or makeinfo.

`--enable-version-specific-runtime-libs`

Specify that runtime libraries should be installed in the compiler specific subdirectory ('*libdir*`/gcc`') rather than the usual places. In addition, '`libstdc++`''s include files will be installed into '*libdir*' unless you overruled it by using '`--with-gxx-include-dir=`*dirname*'. Using this option is particularly useful if you intend to use several versions of GCC in parallel. The default is '`yes`' for '`libada`', and '`no`' for the remaining libraries.

`--with-aix-soname='aix', 'svr4' or 'both'`

Traditional AIX shared library versioning (versioned `Shared Object` files as members of unversioned `Archive Library` files named '`lib.a`') causes numerous headaches for package managers. However, `Import Files` as members of `Archive Library` files allow for **filename-based versioning** of shared libraries as seen on Linux/SVR4, where this is called the "SONAME". But as they prevent static linking, `Import Files` may be used with `Runtime Linking` only, where the linker does search for '`libNAME.so`' before '`libNAME.a`' library filenames with the '`-lNAME`' linker flag.

For detailed information please refer to the AIX ld Command reference.

As long as shared library creation is enabled, upon:

`--with-aix-soname=aix`
`--with-aix-soname=both`

A (traditional AIX) `Shared Archive Library` file is created:

- using the '`libNAME.a`' filename scheme
- with the `Shared Object` file as archive member named '`libNAME.so.V`' (except for '`libgcc_s`', where the `Shared Object` file is named '`shr.o`' for backwards compatibility), which
  - is used for runtime loading from inside the '`libNAME.a`' file
  - is used for dynamic loading via `dlopen("libNAME.a(libNAME.so.V)", RTLD_MEMBER)`
  - is used for shared linking
  - is used for static linking, so no separate `Static Archive Library` file is needed

`--with-aix-soname=both`
`--with-aix-soname=svr4`

A (second) `Shared Archive Library` file is created:

- using the '`libNAME.so.V`' filename scheme
- with the `Shared Object` file as archive member named '`shr.o`', which
  - is created with the `-G linker flag`
  - has the `F_LOADONLY` flag set
  - is used for runtime loading from inside the '`libNAME.so.V`' file
  - is used for dynamic loading via `dlopen("libNAME.so.V(shr.o)", RTLD_MEMBER)`
- with the `Import File` as archive member named '`shr.imp`', which
  - refers to '`libNAME.so.V(shr.o)`' as the "SONAME", to be recorded in the `Loader Section` of subsequent binaries
  - indicates whether '`libNAME.so.V(shr.o)`' is 32 or 64 bit
  - lists all the public symbols exported by '`lib.so.V(shr.o)`', eventually decorated with the `'weak' Keyword`
  - is necessary for shared linking against '`lib.so.V(shr.o)`'

A symbolic link using the '`libNAME.so`' filename scheme is created:

- pointing to the '`libNAME.so.V`' `Shared Archive Library` file
- to permit the `ld Command` to find '`lib.so.V(shr.imp)`' via the '`-lNAME`' argument (requires `Runtime Linking` to be enabled)
- to permit dynamic loading of '`lib.so.V(shr.o)`' without the need to specify the version number via `dlopen("libNAME.so(shr.o)", RTLD_MEMBER)`

As long as static library creation is enabled, upon:

`--with-aix-soname=svr4`

A `Static Archive Library` is created:

- using the '`libNAME.a`' filename scheme
- with all the `Static Object` files as archive members, which
  - are used for static linking

While the aix-soname='`svr4`' option does not create `Shared Object` files as members of unversioned `Archive Library` files any more, package managers still are responsible to transfer `Shared Object` files found as member of a previously installed unversioned `Archive Library` file into the newly installed `Archive Library` file with the same filename.

WARNING: Creating `Shared Object` files with `Runtime Linking` enabled may bloat the TOC, eventually leading to `TOC overflow` errors, requiring the use of

either the '`-Wl,-bbigtoc`' linker flag (seen to break with the `GDB` debugger) or some of the TOC-related compiler flags, See Section "RS/6000 and PowerPC Options" in *Using the GNU Compiler Collection (GCC)*.

'`--with-aix-soname`' is currently supported by '`libgcc_s`' only, so this option is still experimental and not for normal use yet.

Default is the traditional behavior '`--with-aix-soname='aix''`'.

`--enable-languages=`*`lang1`*`,`*`lang2`*`,...`
: Specify that only a particular subset of compilers and their runtime libraries should be built. For a list of valid values for *langN* you can issue the following command in the '`gcc`' directory of your GCC source tree:

```
grep ^language= */config-lang.in
```

Currently, you can use any of the following: `all`, `default`, `ada`, `c`, `c++`, `d`, `fortran`, `go`, `jit`, `lto`, `objc`, `obj-c++`. Building the Ada compiler has special requirements, see below. If you do not pass this flag, or specify the option `default`, then the default languages available in the '`gcc`' sub-tree will be configured. Ada, D, Go, Jit, and Objective-C++ are not default languages. LTO is not a default language, but is built by default because '`--enable-lto`' is enabled by default. The other languages are default languages. If `all` is specified, then all available languages are built. An exception is `jit` language, which requires '`--enable-host-shared`' to be included with `all`.

`--enable-stage1-languages=`*`lang1`*`,`*`lang2`*`,...`
: Specify that a particular subset of compilers and their runtime libraries should be built with the system C compiler during stage 1 of the bootstrap process, rather than only in later stages with the bootstrapped C compiler. The list of valid values is the same as for '`--enable-languages`', and the option `all` will select all of the languages enabled by '`--enable-languages`'. This option is primarily useful for GCC development; for instance, when a development version of the compiler cannot bootstrap due to compiler bugs, or when one is debugging front ends other than the C front end. When this option is used, one can then build the target libraries for the specified languages with the stage-1 compiler by using `make stage1-bubble all-target`, or run the testsuite on the stage-1 compiler for the specified languages using `make stage1-start check-gcc`.

`--disable-libada`
: Specify that the run-time libraries and tools used by GNAT should not be built. This can be useful for debugging, or for compatibility with previous Ada build procedures, when it was required to explicitly do a '`make -C gcc gnatlib_and_tools`'.

`--disable-libsanitizer`
: Specify that the run-time libraries for the various sanitizers should not be built.

`--disable-libssp`
: Specify that the run-time libraries for stack smashing protection should not be built or linked against. On many targets library support is provided by the C library instead.

`--disable-libquadmath`
: Specify that the GCC quad-precision math library should not be built. On some systems, the library is required to be linkable when building the Fortran front end, unless '`--disable-libquadmath-support`' is used.

`--disable-libquadmath-support`
: Specify that the Fortran front end and `libgfortran` do not add support for `libquadmath` on systems supporting it.

`--disable-libgomp`
: Specify that the GNU Offloading and Multi Processing Runtime Library should not be built.

`--disable-libvtv`
: Specify that the run-time libraries used by vtable verification should not be built.

`--with-dwarf2`
: Specify that the compiler should use DWARF 2 debugging information as the default.

`--with-advance-toolchain=`*at*
: On 64-bit PowerPC Linux systems, configure the compiler to use the header files, library files, and the dynamic linker from the Advance Toolchain release *at* instead of the default versions that are provided by the Linux distribution. In general, this option is intended for the developers of GCC, and it is not intended for general use.

`--enable-targets=all`
`--enable-targets=`*target_list*
: Some GCC targets, e.g. powerpc64-linux, build bi-arch compilers. These are compilers that are able to generate either 64-bit or 32-bit code. Typically, the corresponding 32-bit target, e.g. powerpc-linux for powerpc64-linux, only generates 32-bit code. This option enables the 32-bit target to be a bi-arch compiler, which is useful when you want a bi-arch compiler that defaults to 32-bit, and you are building a bi-arch or multi-arch binutils in a combined tree. On mips-linux, this will build a tri-arch compiler (ABI o32/n32/64), defaulted to o32. Currently, this option only affects sparc-linux, powerpc-linux, x86-linux, mips-linux and s390-linux.

`--enable-default-pie`
: Turn on '`-fPIE`' and '`-pie`' by default.

`--enable-secureplt`
: This option enables '`-msecure-plt`' by default for powerpc-linux. See Section "RS/6000 and PowerPC Options" in *Using the GNU Compiler Collection (GCC)*,

`--enable-default-ssp`
: Turn on '`-fstack-protector-strong`' by default.

`--enable-cld`
: This option enables '`-mcld`' by default for 32-bit x86 targets. See Section "i386 and x86-64 Options" in *Using the GNU Compiler Collection (GCC)*,

`--enable-large-address-aware`
: The '`--enable-large-address-aware`' option arranges for MinGW executables to be linked using the '`--large-address-aware`' option, that enables the use of more than 2GB of memory. If GCC is configured with this option, its effects can be reversed by passing the '`-Wl,--disable-large-address-aware`' option to the so-configured compiler driver.

`--enable-win32-registry`
`--enable-win32-registry=`*key*
`--disable-win32-registry`
: The '`--enable-win32-registry`' option enables Microsoft Windows-hosted GCC to look up installations paths in the registry using the following key:

    `HKEY_LOCAL_MACHINE\SOFTWARE\Free Software Foundation\`*key*

  key defaults to GCC version number, and can be overridden by the '`--enable-win32-registry=`*key*' option. Vendors and distributors who use custom installers are encouraged to provide a different key, perhaps one comprised of vendor name and GCC version number, to avoid conflict with existing installations. This feature is enabled by default, and can be disabled by '`--disable-win32-registry`' option. This option has no effect on the other hosts.

`--nfp`
: Specify that the machine does not have a floating point unit. This option only applies to '`m68k-sun-sunos`*n*'. On any other system, '`--nfp`' has no effect.

`--enable-werror`
`--disable-werror`
`--enable-werror=yes`
`--enable-werror=no`
: When you specify this option, it controls whether certain files in the compiler are built with '`-Werror`' in bootstrap stage2 and later. If you don't specify it, '`-Werror`' is turned on for the main development trunk. However it defaults to off for release branches and final releases. The specific files which get '`-Werror`' are controlled by the Makefiles.

`--enable-checking`
`--disable-checking`
`--enable-checking=`*list*
: This option controls performing internal consistency checks in the compiler. It does not change the generated code, but adds error checking of the requested complexity. This slows down the compiler and may only work properly if you are building the compiler with GCC.

  When the option is not specified, the active set of checks depends on context. Namely, bootstrap stage 1 defaults to '`--enable-checking=yes`', builds from release branches or release archives default to '`--enable-checking=release`', and otherwise '`--enable-checking=yes,extra`' is used. When the option is

specified without a *list*, the result is the same as '`--enable-checking=yes`'. Likewise, '`--disable-checking`' is equivalent to '`--enable-checking=no`'.

The categories of checks available in *list* are '`yes`' (most common checks '`assert,misc,gc,gimple,rtlflag,runtime,tree,types`'), '`no`' (no checks at all), '`all`' (all but '`valgrind`'), '`release`' (cheapest checks '`assert,runtime`') or '`none`' (same as '`no`'). '`release`' checks are always on and to disable them '`--disable-checking`' or '`--enable-checking=no[,<other checks>]`' must be explicitly requested. Disabling assertions makes the compiler and runtime slightly faster but increases the risk of undetected internal errors causing wrong code to be generated.

Individual checks can be enabled with these flags: '`assert`', '`df`', '`extra`', '`fold`', '`gc`', '`gcac`', '`gimple`', '`misc`', '`rtl`', '`rtlflag`', '`runtime`', '`tree`', '`types`' and '`valgrind`'. '`extra`' extends '`misc`' checking with extra checks that might affect code generation and should therefore not differ between stage1 and later stages in bootstrap.

The '`valgrind`' check requires the external `valgrind` simulator, available from http://valgrind.org/. The '`rtl`' checks are expensive and the '`df`', '`gcac`' and '`valgrind`' checks are very expensive.

`--disable-stage1-checking`
`--enable-stage1-checking`
`--enable-stage1-checking=`*`list`*

This option affects only bootstrap build. If no '`--enable-checking`' option is specified the stage1 compiler is built with '`yes`' checking enabled, otherwise the stage1 checking flags are the same as specified by '`--enable-checking`'. To build the stage1 compiler with different checking options use '`--enable-stage1-checking`'. The list of checking options is the same as for '`--enable-checking`'. If your system is too slow or too small to bootstrap a released compiler with checking for stage1 enabled, you can use '`--disable-stage1-checking`' to disable checking for the stage1 compiler.

`--enable-coverage`
`--enable-coverage=`*`level`*

With this option, the compiler is built to collect self coverage information, every time it is run. This is for internal development purposes, and only works when the compiler is being built with gcc. The *level* argument controls whether the compiler is built optimized or not, values are '`opt`' and '`noopt`'. For coverage analysis you want to disable optimization, for performance analysis you want to enable optimization. When coverage is enabled, the default level is without optimization.

`--enable-gather-detailed-mem-stats`

When this option is specified more detailed information on memory allocation is gathered. This information is printed when using '`-fmem-report`'.

`--enable-valgrind-annotations`

Mark selected memory related operations in the compiler when run under valgrind to suppress false positives.

`--enable-nls`
`--disable-nls`
: The '`--enable-nls`' option enables Native Language Support (NLS), which lets GCC output diagnostics in languages other than American English. Native Language Support is enabled by default if not doing a canadian cross build. The '`--disable-nls`' option disables NLS.

`--with-included-gettext`
: If NLS is enabled, the '`--with-included-gettext`' option causes the build procedure to prefer its copy of GNU `gettext`.

`--with-catgets`
: If NLS is enabled, and if the host lacks `gettext` but has the inferior `catgets` interface, the GCC build procedure normally ignores `catgets` and instead uses GCC's copy of the GNU `gettext` library. The '`--with-catgets`' option causes the build procedure to use the host's `catgets` in this situation.

`--with-libiconv-prefix=`*`dir`*
: Search for libiconv header files in '*`dir`*`/include`' and libiconv library files in '*`dir`*`/lib`'.

`--enable-obsolete`
: Enable configuration for an obsoleted system. If you attempt to configure GCC for a system (build, host, or target) which has been obsoleted, and you do not specify this flag, configure will halt with an error message.

  All support for systems which have been obsoleted in one release of GCC is removed entirely in the next major release, unless someone steps forward to maintain the port.

`--enable-decimal-float`
`--enable-decimal-float=yes`
`--enable-decimal-float=no`
`--enable-decimal-float=bid`
`--enable-decimal-float=dpd`
`--disable-decimal-float`
: Enable (or disable) support for the C decimal floating point extension that is in the IEEE 754-2008 standard. This is enabled by default only on PowerPC, i386, and x86_64 GNU/Linux systems. Other systems may also support it, but require the user to specifically enable it. You can optionally control which decimal floating point format is used (either '`bid`' or '`dpd`'). The '`bid`' (binary integer decimal) format is default on i386 and x86_64 systems, and the '`dpd`' (densely packed decimal) format is default on PowerPC systems.

`--enable-fixed-point`
`--disable-fixed-point`
: Enable (or disable) support for C fixed-point arithmetic. This option is enabled by default for some targets (such as MIPS) which have hardware-support for fixed-point operations. On other targets, you may enable this option manually.

`--with-long-double-128`
: Specify if `long double` type should be 128-bit by default on selected GNU/Linux architectures. If using `--without-long-double-128`, `long double` will be by default 64-bit, the same as `double` type. When neither of these configure options are used, the default will be 128-bit `long double` when built against GNU C Library 2.4 and later, 64-bit `long double` otherwise.

`--with-long-double-format=ibm`
`--with-long-double-format=ieee`
: Specify whether `long double` uses the IBM extended double format or the IEEE 128-bit floating point format on PowerPC Linux systems. This configuration switch will only work on little endian PowerPC Linux systems and on big endian 64-bit systems where the default cpu is at least power7 (i.e. '`--with-cpu=power7`', '`--with-cpu=power8`', or '`--with-cpu=power9`' is used).

  If you use the '`--with-long-double-64`' configuration option, the '`--with-long-double-format=ibm`' and '`--with-long-double-format=ieee`' options are ignored.

  The default `long double` format is to use IBM extended double. Until all of the libraries are converted to use IEEE 128-bit floating point, it is not recommended to use '`--with-long-double-format=ieee`'.

  On little endian PowerPC Linux systems, if you explicitly set the `long double` type, it will build multilibs to allow you to select either `long double` format, unless you disable multilibs with the `--disable-multilib` option. At present, `long double` multilibs are not built on big endian PowerPC Linux systems. If you are building multilibs, you will need to configure the compiler using the '`--with-system-zlib`' option.

  If you do not set the `long double` type explicitly, no multilibs will be generated.

`--enable-fdpic`
: On SH Linux systems, generate ELF FDPIC code.

`--with-gmp=`*`pathname`*
`--with-gmp-include=`*`pathname`*
`--with-gmp-lib=`*`pathname`*
`--with-mpfr=`*`pathname`*
`--with-mpfr-include=`*`pathname`*
`--with-mpfr-lib=`*`pathname`*
`--with-mpc=`*`pathname`*
`--with-mpc-include=`*`pathname`*
`--with-mpc-lib=`*`pathname`*
: If you want to build GCC but do not have the GMP library, the MPFR library and/or the MPC library installed in a standard location and do not have their sources present in the GCC source tree then you can explicitly specify the directory where they are installed ('`--with-gmp=`*`gmpinstalldir`*', '`--with-mpfr=`*`mpfrinstalldir`*', '`--with-mpc=`*`mpcinstalldir`*'). The '`--with-gmp=`*`gmpinstalldir`*' option is shorthand for '`--with-gmp-lib=`*`gmpinstalldir`*`/lib`' and '`--with-gmp-include=`*`gmpinstalldir`*`/include`'.

Likewise the '`--with-mpfr=`*`mpfrinstalldir`*' option is shorthand for '`--with-mpfr-lib=`*`mpfrinstalldir`*`/lib`' and '`--with-mpfr-include=`*`mpfrinstalldir`*`/include`', also the '`--with-mpc=`*`mpcinstalldir`*' option is shorthand for '`--with-mpc-lib=`*`mpcinstalldir`*`/lib`' and '`--with-mpc-include=`*`mpcinstalldir`*`/include`'. If these shorthand assumptions are not correct, you can use the explicit include and lib options directly. You might also need to ensure the shared libraries can be found by the dynamic linker when building and using GCC, for example by setting the runtime shared library path variable (`LD_LIBRARY_PATH` on GNU/Linux and Solaris systems).

These flags are applicable to the host platform only. When building a cross compiler, they will not be used to configure target libraries.

`--with-isl=`*`pathname`*
`--with-isl-include=`*`pathname`*
`--with-isl-lib=`*`pathname`*

If you do not have the isl library installed in a standard location and you want to build GCC, you can explicitly specify the directory where it is installed ('`--with-isl=`*`islinstalldir`*'). The '`--with-isl=`*`islinstalldir`*' option is shorthand for '`--with-isl-lib=`*`islinstalldir`*`/lib`' and '`--with-isl-include=`*`islinstalldir`*`/include`'. If this shorthand assumption is not correct, you can use the explicit include and lib options directly.

These flags are applicable to the host platform only. When building a cross compiler, they will not be used to configure target libraries.

`--with-stage1-ldflags=`*`flags`*

This option may be used to set linker flags to be used when linking stage 1 of GCC. These are also used when linking GCC if configured with '`--disable-bootstrap`'. If '`--with-stage1-libs`' is not set to a value, then the default is '`-static-libstdc++ -static-libgcc`', if supported.

`--with-stage1-libs=`*`libs`*

This option may be used to set libraries to be used when linking stage 1 of GCC. These are also used when linking GCC if configured with '`--disable-bootstrap`'.

`--with-boot-ldflags=`*`flags`*

This option may be used to set linker flags to be used when linking stage 2 and later when bootstrapping GCC. If –with-boot-libs is not is set to a value, then the default is '`-static-libstdc++ -static-libgcc`'.

`--with-boot-libs=`*`libs`*

This option may be used to set libraries to be used when linking stage 2 and later when bootstrapping GCC.

`--with-debug-prefix-map=`*`map`*

Convert source directory names using '`-fdebug-prefix-map`' when building runtime libraries. '*`map`*' is a space-separated list of maps of the form '*`old=new`*'.

`--enable-linker-build-id`
: Tells GCC to pass '`--build-id`' option to the linker for all final links (links performed without the '`-r`' or '`--relocatable`' option), if the linker supports it. If you specify '`--enable-linker-build-id`', but your linker does not support '`--build-id`' option, a warning is issued and the '`--enable-linker-build-id`' option is ignored. The default is off.

`--with-linker-hash-style=`*choice*
: Tells GCC to pass '`--hash-style=`*choice*' option to the linker for all final links. *choice* can be one of '`sysv`', '`gnu`', and '`both`' where '`sysv`' is the default.

`--enable-gnu-unique-object`
`--disable-gnu-unique-object`
: Tells GCC to use the gnu_unique_object relocation for C++ template static data members and inline function local statics. Enabled by default for a toolchain with an assembler that accepts it and GLIBC 2.11 or above, otherwise disabled.

`--with-diagnostics-color=`*choice*
: Tells GCC to use *choice* as the default for '`-fdiagnostics-color=`' option (if not used explicitly on the command line). *choice* can be one of '`never`', '`auto`', '`always`', and '`auto-if-env`' where '`auto`' is the default. '`auto-if-env`' makes '`-fdiagnostics-color=auto`' the default if `GCC_COLORS` is present and non-empty in the environment of the compiler, and '`-fdiagnostics-color=never`' otherwise.

`--with-diagnostics-urls=`*choice*
: Tells GCC to use *choice* as the default for '`-fdiagnostics-urls=`' option (if not used explicitly on the command line). *choice* can be one of '`never`', '`auto`', '`always`', and '`auto-if-env`' where '`auto`' is the default. '`auto-if-env`' makes '`-fdiagnostics-urls=auto`' the default if `GCC_URLS` or `TERM_URLS` is present and non-empty in the environment of the compiler, and '`-fdiagnostics-urls=never`' otherwise.

`--enable-lto`
`--disable-lto`
: Enable support for link-time optimization (LTO). This is enabled by default, and may be disabled using '`--disable-lto`'.

`--enable-linker-plugin-configure-flags=FLAGS`
`--enable-linker-plugin-flags=FLAGS`
: By default, linker plugins (such as the LTO plugin) are built for the host system architecture. For the case that the linker has a different (but run-time compatible) architecture, these flags can be specified to build plugins that are compatible to the linker. For example, if you are building GCC for a 64-bit x86_64 ('`x86_64-pc-linux-gnu`') host system, but have a 32-bit x86 GNU/Linux ('`i686-pc-linux-gnu`') linker executable (which is executable on the former system), you can configure GCC as follows for getting compatible linker plugins:

```
% srcdir/configure \
    --host=x86_64-pc-linux-gnu \
    --enable-linker-plugin-configure-flags=--host=i686-pc-linux-gnu \
```

```
    --enable-linker-plugin-flags='CC=gcc\ -m32\ -Wl,-rpath,[...]/i686-pc-linux-
gnu/lib'
```

`--with-plugin-ld=`*`pathname`*

Enable an alternate linker to be used at link-time optimization (LTO) link time when '`-fuse-linker-plugin`' is enabled. This linker should have plugin support such as gold starting with version 2.20 or GNU ld starting with version 2.21. See '`-fuse-linker-plugin`' for details.

`--enable-canonical-system-headers`
`--disable-canonical-system-headers`

Enable system header path canonicalization for '`libcpp`'. This can produce shorter header file paths in diagnostics and dependency output files, but these changed header paths may conflict with some compilation environments. Enabled by default, and may be disabled using '`--disable-canonical-system-headers`'.

`--with-glibc-version=`*`major.minor`*

Tell GCC that when the GNU C Library (glibc) is used on the target it will be version *major.minor* or later. Normally this can be detected from the C library's header files, but this option may be needed when bootstrapping a cross toolchain without the header files available for building the initial bootstrap compiler.

If GCC is configured with some multilibs that use glibc and some that do not, this option applies only to the multilibs that use glibc. However, such configurations may not work well as not all the relevant configuration in GCC is on a per-multilib basis.

`--enable-as-accelerator-for=`*`target`*

Build as offload target compiler. Specify offload host triple by *target*.

`--enable-offload-targets=`*`target1`*`[=`*`path1`*`],...,`*`targetN`*`[=`*`pathN`*`]`

Enable offloading to targets *target1*, ..., *targetN*. Offload compilers are expected to be already installed. Default search path for them is '*`exec-prefix`*', but it can be changed by specifying paths *path1*, ..., *pathN*.

```
% srcdir/configure \
    --enable-offload-targets=x86_64-intelmicemul-linux-gnu=/path/to/x86_64/compiler,nvptx-
none,hsa
```

If '`hsa`' is specified as one of the targets, the compiler will be built with support for HSA GPU accelerators. Because the same compiler will emit the accelerator code, no path should be specified.

`--with-hsa-runtime=`*`pathname`*
`--with-hsa-runtime-include=`*`pathname`*
`--with-hsa-runtime-lib=`*`pathname`*

If you configure GCC with HSA offloading but do not have the HSA runtime library installed in a standard location then you can explicitly specify the directory where they are installed. The '`--with-hsa-runtime=`*`hsainstalldir`*' option is a shorthand for '`--with-hsa-runtime-lib=`*`hsainstalldir`*`/lib`' and '`--with-hsa-runtime-include=`*`hsainstalldir`*`/include`'.

`--enable-cet`
`--disable-cet`
: Enable building target run-time libraries with control-flow instrumentation, see '`-fcf-protection`' option. When `--enable-cet` is specified target libraries are configured to add '`-fcf-protection`' and, if needed, other target specific options to a set of building options.

  The option is disabled by default. When `--enable-cet=auto` is used, it is enabled on Linux/x86 if target binutils supports `Intel CET` instructions and disabled otherwise. In this case the target libraries are configured to get additional '`-fcf-protection`' option.

`--with-riscv-attribute='yes', 'no' or 'default'`
: Generate RISC-V attribute by default, in order to record extra build information in object.

  The option is disabled by default. It is enabled on RISC-V/ELF (bare-metal) target if target binutils supported.

Cross-Compiler-Specific Options

The following options only apply to building cross compilers.

`--with-toolexeclibdir=`*dir*
: Specify the installation directory for libraries built with a cross compiler. The default is '`${gcc_tooldir}/lib`'.

`--with-sysroot`
`--with-sysroot=`*dir*
: Tells GCC to consider *dir* as the root of a tree that contains (a subset of) the root filesystem of the target operating system. Target system headers, libraries and run-time object files will be searched for in there. More specifically, this acts as if '`--sysroot=`*dir*' was added to the default options of the built compiler. The specified directory is not copied into the install tree, unlike the options '`--with-headers`' and '`--with-libs`' that this option obsoletes. The default value, in case '`--with-sysroot`' is not given an argument, is '`${gcc_tooldir}/sys-root`'. If the specified directory is a subdirectory of '`${exec_prefix}`', then it will be found relative to the GCC binaries if the installation tree is moved.

  This option affects the system root for the compiler used to build target libraries (which runs on the build system) and the compiler newly installed with `make install`; it does not affect the compiler which is used to build GCC itself.

  If you specify the '`--with-native-system-header-dir=`*dirname*' option then the compiler will search that directory within *dirname* for native system headers rather than the default '`/usr/include`'.

`--with-build-sysroot`
`--with-build-sysroot=`*dir*
: Tells GCC to consider *dir* as the system root (see '`--with-sysroot`') while building target libraries, instead of the directory specified with '`--with-sysroot`'. This option is only useful when you are already using

'`--with-sysroot`'. You can use '`--with-build-sysroot`' when you are configuring with '`--prefix`' set to a directory that is different from the one in which you are installing GCC and your target libraries.

This option affects the system root for the compiler used to build target libraries (which runs on the build system); it does not affect the compiler which is used to build GCC itself.

If you specify the '`--with-native-system-header-dir=`*dirname*' option then the compiler will search that directory within *dirname* for native system headers rather than the default '`/usr/include`'.

`--with-headers`
`--with-headers=`*dir*

Deprecated in favor of '`--with-sysroot`'. Specifies that target headers are available when building a cross compiler. The *dir* argument specifies a directory which has the target include files. These include files will be copied into the '`gcc`' install directory. *This option with the dir argument is required* when building a cross compiler, if '*prefix*`/`*target*`/sys-include`' doesn't pre-exist. If '*prefix*`/`*target*`/sys-include`' does pre-exist, the *dir* argument may be omitted. `fixincludes` will be run on these files to make them compatible with GCC.

`--without-headers`

Tells GCC not use any target headers from a libc when building a cross compiler. When crossing to GNU/Linux, you need the headers so GCC can build the exception handling for libgcc.

`--with-libs`
`--with-libs="`*dir1 dir2 ... dirN*`"`

Deprecated in favor of '`--with-sysroot`'. Specifies a list of directories which contain the target runtime libraries. These libraries will be copied into the '`gcc`' install directory. If the directory list is omitted, this option has no effect.

`--with-newlib`

Specifies that '`newlib`' is being used as the target C library. This causes `__eprintf` to be omitted from '`libgcc.a`' on the assumption that it will be provided by '`newlib`'.

`--with-avrlibc`

Only supported for the AVR target. Specifies that '`AVR-Libc`' is being used as the target C library. This causes float support functions like `__addsf3` to be omitted from '`libgcc.a`' on the assumption that it will be provided by '`libm.a`'. For more technical details, cf. PR54461. It is not supported for RTEMS configurations, which currently use newlib. The option is supported since version 4.7.2 and is the default in 4.8.0 and newer.

`--with-double={32|64|32,64|64,32}`
`--with-long-double={32|64|32,64|64,32|double}`

Only supported for the AVR target since version 10. Specify the default layout available for the C/C++ '`double`' and '`long double`' type, respectively. The following rules apply:

- The first value after the '=' specifies the default layout (in bits) of the type and also the default for the '-mdouble=' resp. '-mlong-double=' compiler option.
- If more than one value is specified, respective multilib variants are available, and '-mdouble=' resp. '-mlong-double=' acts as a multilib option.
- If '--with-long-double=double' is specified, 'double' and 'long double' will have the same layout.
- The defaults are '--with-long-double=64,32' and '--with-double=32,64'. The default 'double' layout imposed by the latter is compatible with older versions of the compiler that implement 'double' as a 32-bit type, which does not comply to the language standard.

Not all combinations of '--with-double=' and '--with-long-double=' are valid. For example, the combination '--with-double=32,64' '--with-long-double=32' will be rejected because the first option specifies the availability of multilibs for 'double', whereas the second option implies that 'long double' — and hence also 'double' — is always 32 bits wide.

--with-double-comparison={tristate|bool|libf7}
: Only supported for the AVR target since version 10. Specify what result format is returned by library functions that compare 64-bit floating point values (DFmode). The GCC default is 'tristate'. If the floating point implementation returns a boolean instead, set it to 'bool'.

--with-libf7={libgcc|math|math-symbols|no}
: Only supported for the AVR target since version 10. Specify to which degree code from LibF7 is included in libgcc. LibF7 is an ad-hoc, AVR-specific, 64-bit floating point emulation written in C and (inline) assembly. 'libgcc' adds support for functions that one would usually expect in libgcc like double addition, double comparisons and double conversions. 'math' also adds routines that one would expect in 'libm.a', but with __ (two underscores) prepended to the symbol names as specified by 'math.h'. 'math-symbols' also defines weak aliases for the functions declared in 'math.h'. However, --with-libf7 won't install no 'math.h' header file whatsoever, this file must come from elsewhere. This option sets '--with-double-comparison' to 'bool'.

--with-nds32-lib=*library*
: Specifies that *library* setting is used for building 'libgcc.a'. Currently, the valid *library* is 'newlib' or 'mculib'. This option is only supported for the NDS32 target.

--with-build-time-tools=*dir*
: Specifies where to find the set of target tools (assembler, linker, etc.) that will be used while building GCC itself. This option can be useful if the directory layouts are different between the system you are building GCC on, and the system where you will deploy it.

  For example, on an 'ia64-hp-hpux' system, you may have the GNU assembler and linker in '/usr/bin', and the native tools in a different path, and build a toolchain that expects to find the native tools in '/usr/bin'.

When you use this option, you should ensure that *dir* includes `ar`, `as`, `ld`, `nm`, `ranlib` and `strip` if necessary, and possibly `objdump`. Otherwise, GCC may use an inconsistent set of tools.

Overriding `configure` test results

Sometimes, it might be necessary to override the result of some `configure` test, for example in order to ease porting to a new system or work around a bug in a test. The toplevel `configure` script provides three variables for this:

`build_configargs`
: The contents of this variable is passed to all build `configure` scripts.

`host_configargs`
: The contents of this variable is passed to all host `configure` scripts.

`target_configargs`
: The contents of this variable is passed to all target `configure` scripts.

In order to avoid shell and `make` quoting issues for complex overrides, you can pass a setting for `CONFIG_SITE` and set variables in the site file.

Objective-C-Specific Options

The following options apply to the build of the Objective-C runtime library.

`--enable-objc-gc`
: Specify that an additional variant of the GNU Objective-C runtime library is built, using an external build of the Boehm-Demers-Weiser garbage collector (https://www.hboehm.info/gc/). This library needs to be available for each multilib variant, unless configured with '`--enable-objc-gc='auto''` in which case the build of the additional runtime library is skipped when not available and the build continues.

`--with-target-bdw-gc=`*list*
`--with-target-bdw-gc-include=`*list*
`--with-target-bdw-gc-lib=`*list*
: Specify search directories for the garbage collector header files and libraries. *list* is a comma separated list of key value pairs of the form '*multilibdir=path*', where the default multilib key is named as '.' (dot), or is omitted (e.g. '`--with-target-bdw-gc=/opt/bdw-gc,32=/opt-bdw-gc32`').

  The options '`--with-target-bdw-gc-include`' and '`--with-target-bdw-gc-lib`' must always be specified together for each multilib variant and they take precedence over '`--with-target-bdw-gc`'. If '`--with-target-bdw-gc-include`' is missing values for a multilib, then the value for the default multilib is used (e.g. '`--with-target-bdw-gc-include=/opt/bdw-gc/include`' '`--with-target-bdw-gc-lib=/opt/bdw-gc/lib64,32=/opt-bdw-gc/lib32`'). If none of these options are specified, the library is assumed in default locations.

D-Specific Options

The following options apply to the build of the D runtime library.

`--enable-libphobos-checking`
`--disable-libphobos-checking`
`--enable-libphobos-checking=`*list*

This option controls whether run-time checks and contracts are compiled into the D runtime library. When the option is not specified, the library is built with '`release`' checking. When the option is specified without a *list*, the result is the same as '`--enable-libphobos-checking=yes`'. Likewise, '`--disable-libphobos-checking`' is equivalent to '`--enable-libphobos-checking=no`'.

The categories of checks available in *list* are '`yes`' (compiles libphobos with '`-fno-release`'), '`no`' (compiles libphobos with '`-frelease`'), '`all`' (same as '`yes`'), '`none`' or '`release`' (same as '`no`').

Individual checks available in *list* are '`assert`' (compiles libphobos with an extra option '`-fassert`').

`--with-libphobos-druntime-only`
`--with-libphobos-druntime-only=`*choice*

Specify whether to build only the core D runtime library (druntime), or both the core and standard library (phobos) into libphobos. This is useful for targets that have full support in druntime, but no or incomplete support in phobos. *choice* can be one of '`auto`', '`yes`', and '`no`' where '`auto`' is the default.

When the option is not specified, the default choice '`auto`' means that it is inferred whether the target has support for the phobos standard library. When the option is specified without a *choice*, the result is the same as '`--with-libphobos-druntime-only=yes`'.

`--with-target-system-zlib`

Use installed '`zlib`' rather than that included with GCC. This needs to be available for each multilib variant, unless configured with '`--with-target-system-zlib=‘auto’`' in which case the GCC included '`zlib`' is only used when the system installed library is not available.

5 Building

Now that GCC is configured, you are ready to build the compiler and runtime libraries.

Some commands executed when making the compiler may fail (return a nonzero status) and be ignored by `make`. These failures, which are often due to files that were not found, are expected, and can safely be ignored.

It is normal to have compiler warnings when compiling certain files. Unless you are a GCC developer, you can generally ignore these warnings unless they cause compilation to fail. Developers should attempt to fix any warnings encountered, however they can temporarily continue past warnings-as-errors by specifying the configure flag '`--disable-werror`'.

On certain old systems, defining certain environment variables such as `CC` can interfere with the functioning of `make`.

If you encounter seemingly strange errors when trying to build the compiler in a directory other than the source directory, it could be because you have previously configured the compiler in the source directory. Make sure you have done all the necessary preparations.

If you build GCC on a BSD system using a directory stored in an old System V file system, problems may occur in running `fixincludes` if the System V file system doesn't support symbolic links. These problems result in a failure to fix the declaration of `size_t` in '`sys/types.h`'. If you find that `size_t` is a signed type and that type mismatches occur, this could be the cause.

The solution is not to use such a directory for building GCC.

Similarly, when building from the source repository or snapshots, or if you modify '`*.l`' files, you need the Flex lexical analyzer generator installed. If you do not modify '`*.l`' files, releases contain the Flex-generated files and you do not need Flex installed to build them. There is still one Flex-based lexical analyzer (part of the build machinery, not of GCC itself) that is used even if you only build the C front end.

When building from the source repository or snapshots, or if you modify Texinfo documentation, you need version 4.7 or later of Texinfo installed if you want Info documentation to be regenerated. Releases contain Info documentation pre-built for the unmodified documentation in the release.

5.1 Building a native compiler

For a native build, the default configuration is to perform a 3-stage bootstrap of the compiler when '`make`' is invoked. This will build the entire GCC system and ensure that it compiles itself correctly. It can be disabled with the '`--disable-bootstrap`' parameter to '`configure`', but bootstrapping is suggested because the compiler will be tested more completely and could also have better performance.

The bootstrapping process will complete the following steps:

- Build tools necessary to build the compiler.
- Perform a 3-stage bootstrap of the compiler. This includes building three times the target tools for use by the compiler such as binutils (bfd, binutils, gas, gprof, ld, and opcodes) if they have been individually linked or moved into the top level GCC source tree before configuring.

- Perform a comparison test of the stage2 and stage3 compilers.
- Build runtime libraries using the stage3 compiler from the previous step.

If you are short on disk space you might consider '`make bootstrap-lean`' instead. The sequence of compilation is the same described above, but object files from the stage1 and stage2 of the 3-stage bootstrap of the compiler are deleted as soon as they are no longer needed.

If you wish to use non-default GCC flags when compiling the stage2 and stage3 compilers, set `BOOT_CFLAGS` on the command line when doing '`make`'. For example, if you want to save additional space during the bootstrap and in the final installation as well, you can build the compiler binaries without debugging information as in the following example. This will save roughly 40% of disk space both for the bootstrap and the final installation. (Libraries will still contain debugging information.)

```
make BOOT_CFLAGS='-O' bootstrap
```

You can place non-default optimization flags into `BOOT_CFLAGS`; they are less well tested here than the default of '`-g -O2`', but should still work. In a few cases, you may find that you need to specify special flags such as '`-msoft-float`' here to complete the bootstrap; or, if the native compiler miscompiles the stage1 compiler, you may need to work around this, by choosing `BOOT_CFLAGS` to avoid the parts of the stage1 compiler that were miscompiled, or by using '`make bootstrap4`' to increase the number of stages of bootstrap.

`BOOT_CFLAGS` does not apply to bootstrapped target libraries. Since these are always compiled with the compiler currently being bootstrapped, you can use `CFLAGS_FOR_TARGET` to modify their compilation flags, as for non-bootstrapped target libraries. Again, if the native compiler miscompiles the stage1 compiler, you may need to work around this by avoiding non-working parts of the stage1 compiler. Use `STAGE1_TFLAGS` to this end.

If you used the flag '`--enable-languages=...`' to restrict the compilers to be built, only those you've actually enabled will be built. This will of course only build those runtime libraries, for which the particular compiler has been built. Please note, that re-defining `LANGUAGES` when calling '`make`' **does not** work anymore!

If the comparison of stage2 and stage3 fails, this normally indicates that the stage2 compiler has compiled GCC incorrectly, and is therefore a potentially serious bug which you should investigate and report. (On a few systems, meaningful comparison of object files is impossible; they always appear "different". If you encounter this problem, you will need to disable comparison in the '`Makefile`'.)

If you do not want to bootstrap your compiler, you can configure with '`--disable-bootstrap`'. In particular cases, you may want to bootstrap your compiler even if the target system is not the same as the one you are building on: for example, you could build a `powerpc-unknown-linux-gnu` toolchain on a `powerpc64-unknown-linux-gnu` host. In this case, pass '`--enable-bootstrap`' to the configure script.

`BUILD_CONFIG` can be used to bring in additional customization to the build. It can be set to a whitespace-separated list of names. For each such `NAME`, top-level '`config/NAME.mk`' will be included by the top-level '`Makefile`', bringing in any settings it contains. The default `BUILD_CONFIG` can be set using the configure option '`--with-build-config=NAME...`'. Some examples of supported build configurations are:

‘bootstrap-O1’
: Removes any ‘-O’-started option from BOOT_CFLAGS, and adds ‘-O1’ to it. ‘BUILD_CONFIG=bootstrap-O1’ is equivalent to ‘BOOT_CFLAGS='-g -O1'’.

‘bootstrap-O3’
‘bootstrap-Og’
: Analogous to bootstrap-O1.

‘bootstrap-lto’
: Enables Link-Time Optimization for host tools during bootstrapping. ‘BUILD_CONFIG=bootstrap-lto’ is equivalent to adding ‘-flto’ to ‘BOOT_CFLAGS’. This option assumes that the host supports the linker plugin (e.g. GNU ld version 2.21 or later or GNU gold version 2.21 or later).

‘bootstrap-lto-noplugin’
: This option is similar to bootstrap-lto, but is intended for hosts that do not support the linker plugin. Without the linker plugin static libraries are not compiled with link-time optimizations. Since the GCC middle end and back end are in ‘libbackend.a’ this means that only the front end is actually LTO optimized.

‘bootstrap-lto-lean’
: This option is similar to bootstrap-lto, but is intended for faster build by only using LTO in the final bootstrap stage. With ‘make profiledbootstrap’ the LTO frontend is trained only on generator files.

‘bootstrap-debug’
: Verifies that the compiler generates the same executable code, whether or not it is asked to emit debug information. To this end, this option builds stage2 host programs without debug information, and uses ‘contrib/compare-debug’ to compare them with the stripped stage3 object files. If BOOT_CFLAGS is overridden so as to not enable debug information, stage2 will have it, and stage3 won't. This option is enabled by default when GCC bootstrapping is enabled, if strip can turn object files compiled with and without debug info into identical object files. In addition to better test coverage, this option makes default bootstraps faster and leaner.

‘bootstrap-debug-big’
: Rather than comparing stripped object files, as in bootstrap-debug, this option saves internal compiler dumps during stage2 and stage3 and compares them as well, which helps catch additional potential problems, but at a great cost in terms of disk space. It can be specified in addition to ‘bootstrap-debug’.

‘bootstrap-debug-lean’
: This option saves disk space compared with bootstrap-debug-big, but at the expense of some recompilation. Instead of saving the dumps of stage2 and stage3 until the final compare, it uses ‘-fcompare-debug’ to generate, compare and remove the dumps during stage3, repeating the compilation that already took place in stage2, whose dumps were not saved.

'`bootstrap-debug-lib`'
: This option tests executable code invariance over debug information generation on target libraries, just like `bootstrap-debug-lean` tests it on host programs. It builds stage3 libraries with '`-fcompare-debug`', and it can be used along with any of the `bootstrap-debug` options above.

  There aren't `-lean` or `-big` counterparts to this option because most libraries are only build in stage3, so bootstrap compares would not get significant coverage. Moreover, the few libraries built in stage2 are used in stage3 host programs, so we wouldn't want to compile stage2 libraries with different options for comparison purposes.

'`bootstrap-debug-ckovw`'
: Arranges for error messages to be issued if the compiler built on any stage is run without the option '`-fcompare-debug`'. This is useful to verify the full '`-fcompare-debug`' testing coverage. It must be used along with `bootstrap-debug-lean` and `bootstrap-debug-lib`.

'`bootstrap-cet`'
: This option enables Intel CET for host tools during bootstrapping. '`BUILD_CONFIG=bootstrap-cet`' is equivalent to adding '`-fcf-protection`' to '`BOOT_CFLAGS`'. This option assumes that the host supports Intel CET (e.g. GNU assembler version 2.30 or later).

'`bootstrap-time`'
: Arranges for the run time of each program started by the GCC driver, built in any stage, to be logged to '`time.log`', in the top level of the build tree.

5.2 Building a cross compiler

When building a cross compiler, it is not generally possible to do a 3-stage bootstrap of the compiler. This makes for an interesting problem as parts of GCC can only be built with GCC.

To build a cross compiler, we recommend first building and installing a native compiler. You can then use the native GCC compiler to build the cross compiler. The installed native compiler needs to be GCC version 2.95 or later.

Assuming you have already installed a native copy of GCC and configured your cross compiler, issue the command `make`, which performs the following steps:

- Build host tools necessary to build the compiler.
- Build target tools for use by the compiler such as binutils (bfd, binutils, gas, gprof, ld, and opcodes) if they have been individually linked or moved into the top level GCC source tree before configuring.
- Build the compiler (single stage only).
- Build runtime libraries using the compiler from the previous step.

Note that if an error occurs in any step the make process will exit.

If you are not building GNU binutils in the same source tree as GCC, you will need a cross-assembler and cross-linker installed before configuring GCC. Put them in the directory '*`prefix/target`*`/bin`'. Here is a table of the tools you should put in this directory:

‘`as`’ This should be the cross-assembler.

‘`ld`’ This should be the cross-linker.

‘`ar`’ This should be the cross-archiver: a program which can manipulate archive files (linker libraries) in the target machine’s format.

‘`ranlib`’ This should be a program to construct a symbol table in an archive file.

The installation of GCC will find these programs in that directory, and copy or link them to the proper place to for the cross-compiler to find them when run later.

The easiest way to provide these files is to build the Binutils package. Configure it with the same ‘`--host`’ and ‘`--target`’ options that you use for configuring GCC, then build and install them. They install their executables automatically into the proper directory. Alas, they do not support all the targets that GCC supports.

If you are not building a C library in the same source tree as GCC, you should also provide the target libraries and headers before configuring GCC, specifying the directories with ‘`--with-sysroot`’ or ‘`--with-headers`’ and ‘`--with-libs`’. Many targets also require “start files” such as ‘`crt0.o`’ and ‘`crtn.o`’ which are linked into each executable. There may be several alternatives for ‘`crt0.o`’, for use with profiling or other compilation options. Check your target’s definition of `STARTFILE_SPEC` to find out what start files it uses.

5.3 Building in parallel

GNU Make 3.80 and above, which is necessary to build GCC, support building in parallel. To activate this, you can use ‘`make -j 2`’ instead of ‘`make`’. You can also specify a bigger number, and in most cases using a value greater than the number of processors in your machine will result in fewer and shorter I/O latency hits, thus improving overall throughput; this is especially true for slow drives and network filesystems.

5.4 Building the Ada compiler

[GNAT-prerequisite], page 3.

5.5 Building with profile feedback

It is possible to use profile feedback to optimize the compiler itself. This should result in a faster compiler binary. Experiments done on x86 using gcc 3.3 showed approximately 7 percent speedup on compiling C programs. To bootstrap the compiler with profile feedback, use `make profiledbootstrap`.

When ‘`make profiledbootstrap`’ is run, it will first build a `stage1` compiler. This compiler is used to build a `stageprofile` compiler instrumented to collect execution counts of instruction and branch probabilities. Training run is done by building `stagetrain` compiler. Finally a `stagefeedback` compiler is built using the information collected.

Unlike standard bootstrap, several additional restrictions apply. The compiler used to build `stage1` needs to support a 64-bit integral type. It is recommended to only use GCC for this.

On Linux/x86_64 hosts with some restrictions (no virtualization) it is also possible to do autofdo build with ‘`make autoprofiledback`’. This uses Linux perf to sample branches

in the binary and then rebuild it with feedback derived from the profile. Linux perf and the `autofdo` toolkit needs to be installed for this.

Only the profile from the current build is used, so when an error occurs it is recommended to clean before restarting. Otherwise the code quality may be much worse.

6 Installing GCC: Testing

Before you install GCC, we encourage you to run the testsuites and to compare your results with results from a similar configuration that have been submitted to the gcc-testresults mailing list. Some of these archived results are linked from the build status lists at `http://gcc.gnu.org/buildstat.html`, although not everyone who reports a successful build runs the testsuites and submits the results. This step is optional and may require you to download additional software, but it can give you confidence in your new GCC installation or point out problems before you install and start using your new GCC.

First, you must have downloaded the testsuites. These are part of the full distribution, but if you downloaded the "core" compiler plus any front ends, you must download the testsuites separately.

Second, you must have the testing tools installed. This includes DejaGnu, Tcl, and Expect; the DejaGnu site has links to these. For running the BRIG frontend tests, a tool to assemble the binary BRIGs from HSAIL text, HSAILasm must be installed.

If the directories where `runtest` and `expect` were installed are not in the `PATH`, you may need to set the following environment variables appropriately, as in the following example (which assumes that DejaGnu has been installed under '`/usr/local`'):

```
TCL_LIBRARY = /usr/local/share/tcl8.0
DEJAGNULIBS = /usr/local/share/dejagnu
```

(On systems such as Cygwin, these paths are required to be actual paths, not mounts or links; presumably this is due to some lack of portability in the DejaGnu code.)

Finally, you can run the testsuite (which may take a long time):

```
cd objdir; make -k check
```

This will test various components of GCC, such as compiler front ends and runtime libraries. While running the testsuite, DejaGnu might emit some harmless messages resembling '`WARNING: Couldn't find the global config file.`' or '`WARNING: Couldn't find tool init file`' that can be ignored.

If you are testing a cross-compiler, you may want to run the testsuite on a simulator as described at `http://gcc.gnu.org/simtest-howto.html`.

6.1 How can you run the testsuite on selected tests?

In order to run sets of tests selectively, there are targets '`make check-gcc`' and language specific '`make check-c`', '`make check-c++`', '`make check-d`' '`make check-fortran`', '`make check-ada`', '`make check-objc`', '`make check-obj-c++`', '`make check-lto`' in the '`gcc`' subdirectory of the object directory. You can also just run '`make check`' in a subdirectory of the object directory.

A more selective way to just run all `gcc` execute tests in the testsuite is to use

```
make check-gcc RUNTESTFLAGS="execute.exp other-options"
```

Likewise, in order to run only the `g++` "old-deja" tests in the testsuite with filenames matching '`9805*`', you would use

```
make check-g++ RUNTESTFLAGS="old-deja.exp=9805* other-options"
```

The file-matching expression following *filename*`.exp=` is treated as a series of whitespace-delimited glob expressions so that multiple patterns may be passed, although any whitespace

must either be escaped or surrounded by single quotes if multiple expressions are desired. For example,

```
make check-g++ RUNTESTFLAGS="old-deja.exp=9805*\ virtual2.c other-options"
make check-g++ RUNTESTFLAGS="'old-deja.exp=9805* virtual2.c' other-options"
```

The '`*.exp`' files are located in the testsuite directories of the GCC source, the most important ones being '`compile.exp`', '`execute.exp`', '`dg.exp`' and '`old-deja.exp`'. To get a list of the possible '`*.exp`' files, pipe the output of '`make check`' into a file and look at the '`Running ... .exp`' lines.

6.2 Passing options and running multiple testsuites

You can pass multiple options to the testsuite using the '`--target_board`' option of DejaGNU, either passed as part of '`RUNTESTFLAGS`', or directly to `runtest` if you prefer to work outside the makefiles. For example,

```
make check-g++ RUNTESTFLAGS="--target_board=unix/-O3/-fmerge-constants"
```

will run the standard `g++` testsuites ("unix" is the target name for a standard native testsuite situation), passing '`-O3 -fmerge-constants`' to the compiler on every test, i.e., slashes separate options.

You can run the testsuites multiple times using combinations of options with a syntax similar to the brace expansion of popular shells:

```
..."--target_board=arm-sim\{-mhard-float,-msoft-float\}\{-O1,-O2,-O3,\}"
```

(Note the empty option caused by the trailing comma in the final group.) The following will run each testsuite eight times using the '`arm-sim`' target, as if you had specified all possible combinations yourself:

```
--target_board='arm-sim/-mhard-float/-O1 \
                arm-sim/-mhard-float/-O2 \
                arm-sim/-mhard-float/-O3 \
                arm-sim/-mhard-float \
                arm-sim/-msoft-float/-O1 \
                arm-sim/-msoft-float/-O2 \
                arm-sim/-msoft-float/-O3 \
                arm-sim/-msoft-float'
```

They can be combined as many times as you wish, in arbitrary ways. This list:

```
..."--target_board=unix/-Wextra\{-O3,-fno-strength\}\{-fomit-frame,\}"
```

will generate four combinations, all involving '`-Wextra`'.

The disadvantage to this method is that the testsuites are run in serial, which is a waste on multiprocessor systems. For users with GNU Make and a shell which performs brace expansion, you can run the testsuites in parallel by having the shell perform the combinations and `make` do the parallel runs. Instead of using '`--target_board`', use a special makefile target:

```
make -jN check-testsuite//test-target/option1/option2/...
```

For example,

```
make -j3 check-gcc//sh-hms-sim/{-m1,-m2,-m3,-m3e,-m4}/{,-nofpu}
```

will run three concurrent "make-gcc" testsuites, eventually testing all ten combinations as described above. Note that this is currently only supported in the '`gcc`' subdirectory. (To see how this works, try typing `echo` before the example given here.)

6.3 How to interpret test results

The result of running the testsuite are various '`*.sum`' and '`*.log`' files in the testsuite subdirectories. The '`*.log`' files contain a detailed log of the compiler invocations and the corresponding results, the '`*.sum`' files summarize the results. These summaries contain status codes for all tests:

- PASS: the test passed as expected
- XPASS: the test unexpectedly passed
- FAIL: the test unexpectedly failed
- XFAIL: the test failed as expected
- UNSUPPORTED: the test is not supported on this platform
- ERROR: the testsuite detected an error
- WARNING: the testsuite detected a possible problem

It is normal for some tests to report unexpected failures. At the current time the testing harness does not allow fine grained control over whether or not a test is expected to fail. This problem should be fixed in future releases.

6.4 Submitting test results

If you want to report the results to the GCC project, use the '`contrib/test_summary`' shell script. Start it in the *objdir* with

```
srcdir/contrib/test_summary -p your_commentary.txt \
    -m gcc-testresults@gcc.gnu.org |sh
```

This script uses the `Mail` program to send the results, so make sure it is in your `PATH`. The file '`your_commentary.txt`' is prepended to the testsuite summary and should contain any special remarks you have on your results or your build environment. Please do not edit the testsuite result block or the subject line, as these messages may be automatically processed.

7 Installing GCC: Final installation

Now that GCC has been built (and optionally tested), you can install it with

```
cd objdir && make install
```

We strongly recommend to install into a target directory where there is no previous version of GCC present. Also, the GNAT runtime should not be stripped, as this would break certain features of the debugger that depend on this debugging information (catching Ada exceptions for instance).

That step completes the installation of GCC; user level binaries can be found in '*prefix*/bin' where *prefix* is the value you specified with the '--prefix' to configure (or '/usr/local' by default). (If you specified '--bindir', that directory will be used instead; otherwise, if you specified '--exec-prefix', '*exec-prefix*/bin' will be used.) Headers for the C++ library are installed in '*prefix*/include'; libraries in '*libdir*' (normally '*prefix*/lib'); internal parts of the compiler in '*libdir*/gcc' and '*libexecdir*/gcc'; documentation in info format in '*infodir*' (normally '*prefix*/info').

When installing cross-compilers, GCC's executables are not only installed into '*bindir*', that is, '*exec-prefix*/bin', but additionally into '*exec-prefix*/*target-alias*/bin', if that directory exists. Typically, such *tooldirs* hold target-specific binutils, including assembler and linker.

Installation into a temporary staging area or into a `chroot` jail can be achieved with the command

```
make DESTDIR=path-to-rootdir install
```

where *path-to-rootdir* is the absolute path of a directory relative to which all installation paths will be interpreted. Note that the directory specified by `DESTDIR` need not exist yet; it will be created if necessary.

There is a subtle point with tooldirs and `DESTDIR`: If you relocate a cross-compiler installation with e.g. '`DESTDIR=`*rootdir*', then the directory '*rootdir*/*exec-prefix*/*target-alias*/bin' will be filled with duplicated GCC executables only if it already exists, it will not be created otherwise. This is regarded as a feature, not as a bug, because it gives slightly more control to the packagers using the `DESTDIR` feature.

You can install stripped programs and libraries with

```
make install-strip
```

If you are bootstrapping a released version of GCC then please quickly review the build status page for your release, available from http://gcc.gnu.org/buildstat.html. If your system is not listed for the version of GCC that you built, send a note to gcc@gcc.gnu.org indicating that you successfully built and installed GCC. Include the following information:

- Output from running '*srcdir*/config.guess'. Do not send that file itself, just the one-line output from running it.
- The output of 'gcc -v' for your newly installed gcc. This tells us which version of GCC you built and the options you passed to configure.
- Whether you enabled all languages or a subset of them. If you used a full distribution then this information is part of the configure options in the output of 'gcc -v', but if you downloaded the "core" compiler plus additional front ends then it isn't apparent which ones you built unless you tell us about it.

- If the build was for GNU/Linux, also include:
  - The distribution name and version (e.g., Red Hat 7.1 or Debian 2.2.3); this information should be available from '`/etc/issue`'.
  - The version of the Linux kernel, available from '`uname --version`' or '`uname -a`'.
  - The version of glibc you used; for RPM-based systems like Red Hat, Mandrake, and SuSE type '`rpm -q glibc`' to get the glibc version, and on systems like Debian and Progeny use '`dpkg -l libc6`'.

  For other systems, you can include similar information if you think it is relevant.
- Any other information that you think would be useful to people building GCC on the same configuration. The new entry in the build status list will include a link to the archived copy of your message.

We'd also like to know if the Chapter 9 [Specific], page 57 didn't include your host/target information or if that information is incomplete or out of date. Send a note to `gcc@gcc.gnu.org` detailing how the information should be changed.

If you find a bug, please report it following the bug reporting guidelines.

If you want to print the GCC manuals, do '`cd` *`objdir`*`; make dvi`'. You will need to have `texi2dvi` (version at least 4.7) and TEX installed. This creates a number of '`.dvi`' files in subdirectories of '*`objdir`*'; these may be converted for printing with programs such as `dvips`. Alternately, by using '`make pdf`' in place of '`make dvi`', you can create documentation in the form of '`.pdf`' files; this requires `texi2pdf`, which is included with Texinfo version 4.8 and later. You can also buy printed manuals from the Free Software Foundation, though such manuals may not be for the most recent version of GCC.

If you would like to generate online HTML documentation, do '`cd` *`objdir`*`; make html`' and HTML will be generated for the gcc manuals in '*`objdir`*`/gcc/HTML`'.

8 Installing GCC: Binaries

We are often asked about pre-compiled versions of GCC. While we cannot provide these for all platforms, below you'll find links to binaries for various platforms where creating them by yourself is not easy due to various reasons.

Please note that we did not create these binaries, nor do we support them. If you have any problems installing them, please contact their makers.

- AIX:
  - Bull's Open Source Software Archive for for AIX 6 and AIX 7;
  - AIX Open Source Packages (AIX5L AIX 6.1 AIX 7.1).
- DOS—DJGPP.
- HP-UX:
  - HP-UX Porting Center;
- Solaris 2 (SPARC, Intel):
  - OpenCSW
- macOS:
  - The Homebrew package manager;
  - MacPorts.
- Microsoft Windows:
  - The Cygwin project;
  - The MinGW and mingw-w64 projects.
- OpenPKG offers binaries for quite a number of platforms.
- The GFortran Wiki has links to GNU Fortran binaries for several platforms.

9 Host/target specific installation notes for GCC

Please read this document carefully *before* installing the GNU Compiler Collection on your machine.

Note that this list of install notes is *not* a list of supported hosts or targets. Not all supported hosts and targets are listed here, only the ones that require host-specific or target-specific information have to.

aarch64*-*-*

Binutils pre 2.24 does not have support for selecting '`-mabi`' and does not support ILP32. If it is used to build GCC 4.9 or later, GCC will not support option '`-mabi=ilp32`'.

To enable a workaround for the Cortex-A53 erratum number 835769 by default (for all CPUs regardless of -mcpu option given) at configure time use the '`--enable-fix-cortex-a53-835769`' option. This will enable the fix by default and can be explicitly disabled during compilation by passing the '`-mno-fix-cortex-a53-835769`' option. Conversely, '`--disable-fix-cortex-a53-835769`' will disable the workaround by default. The workaround is disabled by default if neither of '`--enable-fix-cortex-a53-835769`' or '`--disable-fix-cortex-a53-835769`' is given at configure time.

To enable a workaround for the Cortex-A53 erratum number 843419 by default (for all CPUs regardless of -mcpu option given) at configure time use the '`--enable-fix-cortex-a53-843419`' option. This workaround is applied at link time. Enabling the workaround will cause GCC to pass the relevant option to the linker. It can be explicitly disabled during compilation by passing the '`-mno-fix-cortex-a53-843419`' option. Conversely, '`--disable-fix-cortex-a53-843419`' will disable the workaround by default. The workaround is disabled by default if neither of '`--enable-fix-cortex-a53-843419`' or '`--disable-fix-cortex-a53-843419`' is given at configure time.

To enable Branch Target Identification Mechanism and Return Address Signing by default at configure time use the '`--enable-standard-branch-protection`' option. This is equivalent to having '`-mbranch-protection=standard`' during compilation. This can be explicitly disabled during compilation by passing the '`-mbranch-protection=none`' option which turns off all types of branch protections. Conversely, '`--disable-standard-branch-protection`' will disable both the protections by default. This mechanism is turned off by default if neither of the options are given at configure time.

alpha*-*-*

This section contains general configuration information for all Alpha-based platforms using ELF. In addition to reading this section, please read all other sections that match your target.

amd64-*-solaris2*

This is a synonym for '`x86_64-*-solaris2*`'.

amdgcn-*-amdhsa

AMD GCN GPU target.

Instead of GNU Binutils, you will need to install LLVM 6, or later, and copy '`bin/llvm-mc`' to '`amdgcn-amdhsa/bin/as`', '`bin/lld`' to '`amdgcn-amdhsa/bin/ld`', '`bin/llvm-nm`' to '`amdgcn-amdhsa/bin/nm`', and '`bin/llvm-ar`' to both '`bin/amdgcn-amdhsa-ar`' and '`bin/amdgcn-amdhsa-ranlib`'.

Use Newlib (2019-01-16, or newer).

To run the binaries, install the HSA Runtime from the ROCm Platform, and use '`libexec/gcc/amdhsa-amdhsa/`*`version`*`/gcn-run`' to launch them on the GPU.

arc-*-elf32

Use '`configure --target=arc-elf32 --with-cpu=`*`cpu`* `--enable-languages="c,c++"`' to configure GCC, with *cpu* being one of '`arc600`', '`arc601`', or '`arc700`'.

arc-linux-uclibc

Use '`configure --target=arc-linux-uclibc --with-cpu=arc700 --enable-languages="c,c++"`' to configure GCC.

arm-*-eabi

ARM-family processors.

Building the Ada frontend commonly fails (an infinite loop executing `xsinfo`) if the host compiler is GNAT 4.8. Host compilers built from the GNAT 4.6, 4.9 or 5 release branches are known to succeed.

avr

ATMEL AVR-family micro controllers. These are used in embedded applications. There are no standard Unix configurations. See Section "AVR Options" in *Using the GNU Compiler Collection (GCC)*, for the list of supported MCU types.

Use '`configure --target=avr --enable-languages="c"`' to configure GCC.

Further installation notes and other useful information about AVR tools can also be obtained from:

- http://www.nongnu.org/avr/
- http://www.amelek.gda.pl/avr/

The following error:

```
Error: register required
```

indicates that you should upgrade to a newer version of the binutils.

Blackfin

The Blackfin processor, an Analog Devices DSP. See Section "Blackfin Options" in *Using the GNU Compiler Collection (GCC)*,

More information, and a version of binutils with support for this processor, are available at `https://sourceforge.net/projects/adi-toolchain/`.

CR16

The CR16 CompactRISC architecture is a 16-bit architecture. This architecture is used in embedded applications.

See Section "CR16 Options" in *Using and Porting the GNU Compiler Collection (GCC)*,

Use '`configure --target=cr16-elf --enable-languages=c,c++`' to configure GCC for building a CR16 elf cross-compiler.

Use '`configure --target=cr16-uclinux --enable-languages=c,c++`' to configure GCC for building a CR16 uclinux cross-compiler.

CRIS

CRIS is the CPU architecture in Axis Communications ETRAX system-on-a-chip series. These are used in embedded applications.

See Section "CRIS Options" in *Using the GNU Compiler Collection (GCC)*, for a list of CRIS-specific options.

There are a few different CRIS targets:

`cris-axis-elf`
: Mainly for monolithic embedded systems. Includes a multilib for the '`v10`' core used in '`ETRAX 100 LX`'.

`cris-axis-linux-gnu`
: A GNU/Linux port for the CRIS architecture, currently targeting '`ETRAX 100 LX`' by default.

Pre-packaged tools can be obtained from `ftp://ftp.axis.com/pub/axis/tools/cris/compiler-kit/`. More information about this platform is available at `http://developer.axis.com/`.

DOS

Please have a look at the binaries page.

You cannot install GCC by itself on MSDOS; it will not compile under any MSDOS compiler except itself. You need to get the complete compilation package DJGPP, which includes binaries as well as sources, and includes all the necessary compilation tools and libraries.

epiphany-*-elf

Adapteva Epiphany. This configuration is intended for embedded systems.

--freebsd*

Support for FreeBSD 1 was discontinued in GCC 3.2. Support for FreeBSD 2 (and any mutant a.out variants of FreeBSD 3) was discontinued in GCC 4.0.

In order to better utilize FreeBSD base system functionality and match the configuration of the system compiler, GCC 4.5 and above as well as GCC 4.4 past 2010-06-20 leverage SSP support in libc (which is present on FreeBSD 7 or later) and the use of `__cxa_atexit`

by default (on FreeBSD 6 or later). The use of `dl_iterate_phdr` inside '`libgcc_s.so.1`' and boehm-gc (on FreeBSD 7 or later) is enabled by GCC 4.5 and above.

We support FreeBSD using the ELF file format with DWARF 2 debugging for all CPU architectures. You may use '`-gstabs`' instead of '`-g`', if you really want the old debugging format. There are no known issues with mixing object files and libraries with different debugging formats. Otherwise, this release of GCC should now match more of the configuration used in the stock FreeBSD configuration of GCC. In particular, '`--enable-threads`' is now configured by default. However, as a general user, do not attempt to replace the system compiler with this release. Known to bootstrap and check with good results on FreeBSD 7.2-STABLE. In the past, known to bootstrap and check with good results on FreeBSD 3.0, 3.4, 4.0, 4.2, 4.3, 4.4, 4.5, 4.8, 4.9 and 5-CURRENT.

The version of binutils installed in '`/usr/bin`' probably works with this release of GCC. Bootstrapping against the latest GNU binutils and/or the version found in '`/usr/ports/devel/binutils`' has been known to enable additional features and improve overall testsuite results. However, it is currently known that boehm-gc may not configure properly on FreeBSD prior to the FreeBSD 7.0 release with GNU binutils after 2.16.1.

ft32-*-elf

The FT32 processor. This configuration is intended for embedded systems.

h8300-hms

Renesas H8/300 series of processors.

Please have a look at the binaries page.

The calling convention and structure layout has changed in release 2.6. All code must be recompiled. The calling convention now passes the first three arguments in function calls in registers. Structures are no longer a multiple of 2 bytes.

hppa*-hp-hpux*

Support for HP-UX version 9 and older was discontinued in GCC 3.4.

We require using gas/binutils on all hppa platforms. Version 2.19 or later is recommended.

It may be helpful to configure GCC with the '`--with-gnu-as`' and '`--with-as=...`' options to ensure that GCC can find GAS.

The HP assembler should not be used with GCC. It is rarely tested and may not work. It shouldn't be used with any languages other than C due to its many limitations.

Specifically, '`-g`' does not work (HP-UX uses a peculiar debugging format which GCC does not know about). It also inserts timestamps into each object file it creates, causing the 3-stage comparison test to fail during a bootstrap. You should be able to continue by saying '`make all-host all-target`' after getting the failure from '`make`'.

Various GCC features are not supported. For example, it does not support weak symbols or alias definitions. As a result, explicit template instantiations are required when using C++. This makes it difficult if not impossible to build many C++ applications.

There are two default scheduling models for instructions. These are PROCESSOR_7100LC and PROCESSOR_8000. They are selected from the pa-risc architecture specified for the target machine when configuring. PROCESSOR_8000 is the default. PROCESSOR_7100LC is selected when the target is a '`hppa1*`' machine.

The PROCESSOR_8000 model is not well suited to older processors. Thus, it is important to completely specify the machine architecture when configuring if you want a model other than PROCESSOR_8000. The macro TARGET_SCHED_DEFAULT can be defined in BOOT_CFLAGS if a different default scheduling model is desired.

As of GCC 4.0, GCC uses the UNIX 95 namespace for HP-UX 10.10 through 11.00, and the UNIX 98 namespace for HP-UX 11.11 and later. This namespace change might cause problems when bootstrapping with an earlier version of GCC or the HP compiler as essentially the same namespace is required for an entire build. This problem can be avoided in a number of ways. With HP cc, `UNIX_STD` can be set to '`95`' or '`98`'. Another way is to add an appropriate set of predefines to `CC`. The description for the '`munix=`' option contains a list of the predefines used with each standard.

More specific information to '`hppa*-hp-hpux*`' targets follows.

hppa*-hp-hpux10

For hpux10.20, we *highly* recommend you pick up the latest sed patch `PHCO_19798` from HP.

The C++ ABI has changed incompatibly in GCC 4.0. COMDAT subspaces are used for one-only code and data. This resolves many of the previous problems in using C++ on this target. However, the ABI is not compatible with the one implemented under HP-UX 11 using secondary definitions.

hppa*-hp-hpux11

GCC 3.0 and up support HP-UX 11. GCC 2.95.x is not supported and cannot be used to compile GCC 3.0 and up.

The libffi library haven't been ported to 64-bit HP-UX and doesn't build.

Refer to binaries for information about obtaining precompiled GCC binaries for HP-UX. Precompiled binaries must be obtained to build the Ada language as it cannot be bootstrapped using C. Ada is only available for the 32-bit PA-RISC runtime.

Starting with GCC 3.4 an ISO C compiler is required to bootstrap. The bundled compiler supports only traditional C; you will need either HP's unbundled compiler, or a binary distribution of GCC.

It is possible to build GCC 3.3 starting with the bundled HP compiler, but the process requires several steps. GCC 3.3 can then be used to build later versions.

There are several possible approaches to building the distribution. Binutils can be built first using the HP tools. Then, the GCC distribution can be built. The second approach is to build GCC first using the HP tools, then build binutils, then rebuild GCC. There have been problems with various binary distributions, so it is best not to start from a binary distribution.

On 64-bit capable systems, there are two distinct targets. Different installation prefixes must be used if both are to be installed on the same system. The '`hppa[1-2]*-hp-hpux11*`'

target generates code for the 32-bit PA-RISC runtime architecture and uses the HP linker. The '`hppa64-hp-hpux11*`' target generates 64-bit code for the PA-RISC 2.0 architecture.

The script config.guess now selects the target type based on the compiler detected during configuration. You must define `PATH` or `CC` so that configure finds an appropriate compiler for the initial bootstrap. When `CC` is used, the definition should contain the options that are needed whenever `CC` is used.

Specifically, options that determine the runtime architecture must be in `CC` to correctly select the target for the build. It is also convenient to place many other compiler options in `CC`. For example, `CC="cc -Ac +DA2.0W -Wp,-H16376 -D_CLASSIC_TYPES -D_HPUX_SOURCE"` can be used to bootstrap the GCC 3.3 branch with the HP compiler in 64-bit K&R/bundled mode. The '`+DA2.0W`' option will result in the automatic selection of the '`hppa64-hp-hpux11*`' target. The macro definition table of cpp needs to be increased for a successful build with the HP compiler. _CLASSIC_TYPES and _HPUX_SOURCE need to be defined when building with the bundled compiler, or when using the '`-Ac`' option. These defines aren't necessary with '`-Ae`'.

It is best to explicitly configure the '`hppa64-hp-hpux11*`' target with the '`--with-ld=...`' option. This overrides the standard search for ld. The two linkers supported on this target require different commands. The default linker is determined during configuration. As a result, it's not possible to switch linkers in the middle of a GCC build. This has been reported to sometimes occur in unified builds of binutils and GCC.

A recent linker patch must be installed for the correct operation of GCC 3.3 and later. `PHSS_26559` and `PHSS_24304` are the oldest linker patches that are known to work. They are for HP-UX 11.00 and 11.11, respectively. `PHSS_24303`, the companion to `PHSS_24304`, might be usable but it hasn't been tested. These patches have been superseded. Consult the HP patch database to obtain the currently recommended linker patch for your system.

The patches are necessary for the support of weak symbols on the 32-bit port, and for the running of initializers and finalizers. Weak symbols are implemented using SOM secondary definition symbols. Prior to HP-UX 11, there are bugs in the linker support for secondary symbols. The patches correct a problem of linker core dumps creating shared libraries containing secondary symbols, as well as various other linking issues involving secondary symbols.

GCC 3.3 uses the ELF DT_INIT_ARRAY and DT_FINI_ARRAY capabilities to run initializers and finalizers on the 64-bit port. The 32-bit port uses the linker '`+init`' and '`+fini`' options for the same purpose. The patches correct various problems with the +init/+fini options, including program core dumps. Binutils 2.14 corrects a problem on the 64-bit port resulting from HP's non-standard use of the .init and .fini sections for array initializers and finalizers.

Although the HP and GNU linkers are both supported for the '`hppa64-hp-hpux11*`' target, it is strongly recommended that the HP linker be used for link editing on this target.

At this time, the GNU linker does not support the creation of long branch stubs. As a result, it cannot successfully link binaries containing branch offsets larger than 8 megabytes. In addition, there are problems linking shared libraries, linking executables with '`-static`', and with dwarf2 unwind and exception support. It also doesn't provide stubs for internal calls to global functions in shared libraries, so these calls cannot be overloaded.

The HP dynamic loader does not support GNU symbol versioning, so symbol versioning is not supported. It may be necessary to disable symbol versioning with '`--disable-symvers`' when using GNU ld.

POSIX threads are the default. The optional DCE thread library is not supported, so '`--enable-threads=dce`' does not work.

--linux-gnu

Versions of libstdc++-v3 starting with 3.2.1 require bug fixes present in glibc 2.2.5 and later. More information is available in the libstdc++-v3 documentation.

i?86-*-linux*

As of GCC 3.3, binutils 2.13.1 or later is required for this platform. See bug 10877 for more information.

If you receive Signal 11 errors when building on GNU/Linux, then it is possible you have a hardware problem. Further information on this can be found on www.bitwizard.nl.

i?86-*-solaris2*

Use this for Solaris 11.3 or later on x86 and x86-64 systems. Starting with GCC 4.7, there is also a 64-bit '`amd64-*-solaris2*`' or '`x86_64-*-solaris2*`' configuration that corresponds to '`sparcv9-sun-solaris2*`'.

It is recommended that you configure GCC to use the GNU assembler. The versions included in Solaris 11.3, from GNU binutils 2.23.1 or newer (available as '`/usr/bin/gas`' and '`/usr/gnu/bin/as`'), work fine. The current version, from GNU binutils 2.34, is known to work. Recent versions of the Solaris assembler in '`/usr/bin/as`' work almost as well, though.

For linking, the Solaris linker is preferred. If you want to use the GNU linker instead, the version in Solaris 11.3, from GNU binutils 2.23.1 or newer (in '`/usr/gnu/bin/ld`' and '`/usr/bin/gld`'), works, as does the latest version, from GNU binutils 2.34.

To use GNU `as`, configure with the options '`--with-gnu-as --with-as=/usr/gnu/bin/as`'. It may be necessary to configure with '`--without-gnu-ld --with-ld=/usr/ccs/bin/ld`' to guarantee use of Solaris `ld`.

ia64-*-linux

IA-64 processor (also known as IPF, or Itanium Processor Family) running GNU/Linux.

If you are using the installed system libunwind library with '`--with-system-libunwind`', then you must use libunwind 0.98 or later.

None of the following versions of GCC has an ABI that is compatible with any of the other versions in this list, with the exception that Red Hat 2.96 and Trillian 000171 are compatible with each other: 3.1, 3.0.2, 3.0.1, 3.0, Red Hat 2.96, and Trillian 000717. This primarily affects C++ programs and programs that create shared libraries. GCC 3.1 or later is recommended for compiling linux, the kernel. As of version 3.1 GCC is believed to be fully ABI compliant, and hence no more major ABI changes are expected.

ia64-*-hpux*

Building GCC on this target requires the GNU Assembler. The bundled HP assembler will not work. To prevent GCC from using the wrong assembler, the option '`--with-gnu-as`' may be necessary.

The GCC libunwind library has not been ported to HPUX. This means that for GCC versions 3.2.3 and earlier, '`--enable-libunwind-exceptions`' is required to build GCC. For GCC 3.3 and later, this is the default. For gcc 3.4.3 and later, '`--enable-libunwind-exceptions`' is removed and the system libunwind library will always be used.

-ibm-aix

Support for AIX version 3 and older was discontinued in GCC 3.4. Support for AIX version 4.2 and older was discontinued in GCC 4.5.

"out of memory" bootstrap failures may indicate a problem with process resource limits (ulimit). Hard limits are configured in the '`/etc/security/limits`' system configuration file.

GCC 4.9 and above require a C++ compiler for bootstrap. IBM VAC++ / xlC cannot bootstrap GCC. xlc can bootstrap an older version of GCC and G++ can bootstrap recent releases of GCC.

GCC can bootstrap with recent versions of IBM XLC, but bootstrapping with an earlier release of GCC is recommended. Bootstrapping with XLC requires a larger data segment, which can be enabled through the *LDR_CNTRL* environment variable, e.g.,

```
% LDR_CNTRL=MAXDATA=0x50000000
% export LDR_CNTRL
```

One can start with a pre-compiled version of GCC to build from sources. One may delete GCC's "fixed" header files when starting with a version of GCC built for an earlier release of AIX.

To speed up the configuration phases of bootstrapping and installing GCC, one may use GNU Bash instead of AIX `/bin/sh`, e.g.,

```
% CONFIG_SHELL=/opt/freeware/bin/bash
% export CONFIG_SHELL
```

and then proceed as described in the build instructions, where we strongly recommend specifying an absolute path to invoke *srcdir*/configure.

Because GCC on AIX is built as a 32-bit executable by default, (although it can generate 64-bit programs) the GMP and MPFR libraries required by gfortran must be 32-bit libraries. Building GMP and MPFR as static archive libraries works better than shared libraries.

Errors involving `alloca` when building GCC generally are due to an incorrect definition of `CC` in the Makefile or mixing files compiled with the native C compiler and GCC. During the stage1 phase of the build, the native AIX compiler **must** be invoked as `cc` (not `xlc`). Once `configure` has been informed of `xlc`, one needs to use '`make distclean`' to remove the configure cache files and ensure that `CC` environment variable does not provide a definition that will confuse `configure`. If this error occurs during stage2 or later, then the problem most likely is the version of Make (see above).

The native `as` and `ld` are recommended for bootstrapping on AIX. The GNU Assembler, GNU Linker, and GNU Binutils version 2.20 is the minimum level that supports bootstrap on AIX 5. The GNU Assembler has not been updated to support AIX 6 or AIX 7. The native AIX tools do interoperate with GCC.

AIX 7.1 added partial support for DWARF debugging, but full support requires AIX 7.1 TL03 SP7 that supports additional DWARF sections and fixes a bug in the assembler. AIX 7.1 TL03 SP5 distributed a version of libm.a missing important symbols; a fix for IV77796 will be included in SP6.

AIX 5.3 TL10, AIX 6.1 TL05 and AIX 7.1 TL00 introduced an AIX assembler change that sometimes produces corrupt assembly files causing AIX linker errors. The bug breaks GCC bootstrap on AIX and can cause compilation failures with existing GCC installations. An AIX iFix for AIX 5.3 is available (APAR IZ98385 for AIX 5.3 TL10, APAR IZ98477 for AIX 5.3 TL11 and IZ98134 for AIX 5.3 TL12). AIX 5.3 TL11 SP8, AIX 5.3 TL12 SP5, AIX 6.1 TL04 SP11, AIX 6.1 TL05 SP7, AIX 6.1 TL06 SP6, AIX 6.1 TL07 and AIX 7.1 TL01 should include the fix.

Building '`libstdc++.a`' requires a fix for an AIX Assembler bug APAR IY26685 (AIX 4.3) or APAR IY25528 (AIX 5.1). It also requires a fix for another AIX Assembler bug and a co-dependent AIX Archiver fix referenced as APAR IY53606 (AIX 5.2) or as APAR IY54774 (AIX 5.1)

'`libstdc++`' in GCC 3.4 increments the major version number of the shared object and GCC installation places the '`libstdc++.a`' shared library in a common location which will overwrite the and GCC 3.3 version of the shared library. Applications either need to be re-linked against the new shared library or the GCC 3.1 and GCC 3.3 versions of the '`libstdc++`' shared object needs to be available to the AIX runtime loader. The GCC 3.1 '`libstdc++.so.4`', if present, and GCC 3.3 '`libstdc++.so.5`' shared objects can be installed for runtime dynamic loading using the following steps to set the '`F_LOADONLY`' flag in the shared object for *each* multilib '`libstdc++.a`' installed:

Extract the shared objects from the currently installed '`libstdc++.a`' archive:

```
% ar -x libstdc++.a libstdc++.so.4 libstdc++.so.5
```

Enable the '`F_LOADONLY`' flag so that the shared object will be available for runtime dynamic loading, but not linking:

```
% strip -e libstdc++.so.4 libstdc++.so.5
```

Archive the runtime-only shared object in the GCC 3.4 '`libstdc++.a`' archive:

```
% ar -q libstdc++.a libstdc++.so.4 libstdc++.so.5
```

Eventually, the '`--with-aix-soname=svr4`' configure option may drop the need for this procedure for libraries that support it.

Linking executables and shared libraries may produce warnings of duplicate symbols. The assembly files generated by GCC for AIX always have included multiple symbol definitions for certain global variable and function declarations in the original program. The warnings should not prevent the linker from producing a correct library or runnable executable.

AIX 4.3 utilizes a "large format" archive to support both 32-bit and 64-bit object modules. The routines provided in AIX 4.3.0 and AIX 4.3.1 to parse archive libraries did not handle the new format correctly. These routines are used by GCC and result in error messages during linking such as "not a COFF file". The version of the routines shipped with

AIX 4.3.1 should work for a 32-bit environment. The '`-g`' option of the archive command may be used to create archives of 32-bit objects using the original "small format". A correct version of the routines is shipped with AIX 4.3.2 and above.

Some versions of the AIX binder (linker) can fail with a relocation overflow severe error when the '`-bbigtoc`' option is used to link GCC-produced object files into an executable that overflows the TOC. A fix for APAR IX75823 (OVERFLOW DURING LINK WHEN USING GCC AND -BBIGTOC) is available from IBM Customer Support and from its techsupport.services.ibm.com website as PTF U455193.

The AIX 4.3.2.1 linker (bos.rte.bind_cmds Level 4.3.2.1) will dump core with a segmentation fault when invoked by any version of GCC. A fix for APAR IX87327 is available from IBM Customer Support and from its techsupport.services.ibm.com website as PTF U461879. This fix is incorporated in AIX 4.3.3 and above.

The initial assembler shipped with AIX 4.3.0 generates incorrect object files. A fix for APAR IX74254 (64BIT DISASSEMBLED OUTPUT FROM COMPILER FAILS TO ASSEMBLE/BIND) is available from IBM Customer Support and from its techsupport.services.ibm.com website as PTF U453956. This fix is incorporated in AIX 4.3.1 and above.

AIX provides National Language Support (NLS). Compilers and assemblers use NLS to support locale-specific representations of various data formats including floating-point numbers (e.g., '`.`' vs '`,`' for separating decimal fractions). There have been problems reported where GCC does not produce the same floating-point formats that the assembler expects. If one encounters this problem, set the `LANG` environment variable to '`C`' or '`En_US`'.

A default can be specified with the '`-mcpu=`*`cpu_type`*' switch and using the configure option '`--with-cpu-`*`cpu_type`*'.

iq2000-*-elf

Vitesse IQ2000 processors. These are used in embedded applications. There are no standard Unix configurations.

lm32-*-elf

Lattice Mico32 processor. This configuration is intended for embedded systems.

lm32-*-uclinux

Lattice Mico32 processor. This configuration is intended for embedded systems running uClinux.

m32c-*-elf

Renesas M32C processor. This configuration is intended for embedded systems.

m32r-*-elf

Renesas M32R processor. This configuration is intended for embedded systems.

m68k-*-*

By default, '`m68k-*-elf*`', '`m68k-*-rtems`', '`m68k-*-uclinux`' and '`m68k-*-linux`' build libraries for both M680x0 and ColdFire processors. If you only need the M680x0 libraries, you can omit the ColdFire ones by passing '`--with-arch=m68k`' to `configure`. Alternatively, you can omit the M680x0 libraries by passing '`--with-arch=cf`' to `configure`. These targets default to 5206 or 5475 code as appropriate for the target system when configured with '`--with-arch=cf`' and 68020 code otherwise.

The '`m68k-*-netbsd`' and '`m68k-*-openbsd`' targets also support the '`--with-arch`' option. They will generate ColdFire CFV4e code when configured with '`--with-arch=cf`' and 68020 code otherwise.

You can override the default processors listed above by configuring with '`--with-cpu=`*target*'. This *target* can either be a '`-mcpu`' argument or one of the following values: '`m68000`', '`m68010`', '`m68020`', '`m68030`', '`m68040`', '`m68060`', '`m68020-40`' and '`m68020-60`'.

GCC requires at least binutils version 2.17 on these targets.

m68k-*-uclinux

GCC 4.3 changed the uClinux configuration so that it uses the '`m68k-linux-gnu`' ABI rather than the '`m68k-elf`' ABI. It also added improved support for C++ and flat shared libraries, both of which were ABI changes.

microblaze-*-elf

Xilinx MicroBlaze processor. This configuration is intended for embedded systems.

mips-*-*

If on a MIPS system you get an error message saying "does not have gp sections for all it's [sic] sectons [sic]", don't worry about it. This happens whenever you use GAS with the MIPS linker, but there is not really anything wrong, and it is okay to use the output file. You can stop such warnings by installing the GNU linker.

It would be nice to extend GAS to produce the gp tables, but they are optional, and there should not be a warning about their absence.

The libstdc++ atomic locking routines for MIPS targets requires MIPS II and later. A patch went in just after the GCC 3.3 release to make '`mips*-*-*`' use the generic implementation instead. You can also configure for '`mipsel-elf`' as a workaround. The '`mips*-*-linux*`' target continues to use the MIPS II routines. More work on this is expected in future releases.

The built-in `__sync_*` functions are available on MIPS II and later systems and others that support the '`ll`', '`sc`' and '`sync`' instructions. This can be overridden by passing '`--with-llsc`' or '`--without-llsc`' when configuring GCC. Since the Linux kernel emulates these instructions if they are missing, the default for '`mips*-*-linux*`' targets is '`--with-llsc`'. The '`--with-llsc`' and '`--without-llsc`' configure options may be overridden at compile time by passing the '`-mllsc`' or '`-mno-llsc`' options to the compiler.

MIPS systems check for division by zero (unless '`-mno-check-zero-division`' is passed to the compiler) by generating either a conditional trap or a break instruction. Using

trap results in smaller code, but is only supported on MIPS II and later. Also, some versions of the Linux kernel have a bug that prevents trap from generating the proper signal (`SIGFPE`). To enable the use of break, use the '`--with-divide=breaks`' `configure` option when configuring GCC. The default is to use traps on systems that support them.

moxie-*-elf

The moxie processor.

msp430-*-elf*

TI MSP430 processor. This configuration is intended for embedded systems.

'`msp430-*-elf`' is the standard configuration with most GCC features enabled by default.

'`msp430-*-elfbare`' is tuned for a bare-metal environment, and disables features related to shared libraries and other functionality not used for this device. This reduces code and data usage of the GCC libraries, resulting in a minimal run-time environment by default.

Features disabled by default include:

- transactional memory
- __cxa_atexit

nds32le-*-elf

Andes NDS32 target in little endian mode.

nds32be-*-elf

Andes NDS32 target in big endian mode.

nvptx-*-none

Nvidia PTX target.

Instead of GNU binutils, you will need to install nvptx-tools. Tell GCC where to find it: '`--with-build-time-tools=[install-nvptx-tools]/nvptx-none/bin`'.

You will need newlib 3.0 git revision cd31fbb2aea25f94d7ecedc9db16dfc87ab0c316 or later. It can be automatically built together with GCC. For this, add a symbolic link to nvptx-newlib's '`newlib`' directory to the directory containing the GCC sources.

Use the '`--disable-sjlj-exceptions`' and '`--enable-newlib-io-long-long`' options when configuring.

or1k-*-elf

The OpenRISC 1000 32-bit processor with delay slots. This configuration is intended for embedded systems.

or1k-*-linux

The OpenRISC 1000 32-bit processor with delay slots.

powerpc-*-*

You can specify a default version for the '`-mcpu=`*`cpu_type`*' switch by using the configure option '`--with-cpu-`*`cpu_type`*'.

You will need GNU binutils 2.20 or newer.

powerpc-*-darwin*

PowerPC running Darwin (Mac OS X kernel).

Pre-installed versions of Mac OS X may not include any developer tools, meaning that you will not be able to build GCC from source. Tool binaries are available at `https://opensource.apple.com`.

This version of GCC requires at least cctools-590.36. The cctools-590.36 package referenced from `http://gcc.gnu.org/ml/gcc/2006-03/msg00507.html` will not work on systems older than 10.3.9 (aka darwin7.9.0).

powerpc-*-elf

PowerPC system in big endian mode, running System V.4.

powerpc*-*-linux-gnu*

PowerPC system in big endian mode running Linux.

powerpc-*-netbsd*

PowerPC system in big endian mode running NetBSD.

powerpc-*-eabisim

Embedded PowerPC system in big endian mode for use in running under the PSIM simulator.

powerpc-*-eabi

Embedded PowerPC system in big endian mode.

powerpcle-*-elf

PowerPC system in little endian mode, running System V.4.

powerpcle-*-eabisim

Embedded PowerPC system in little endian mode for use in running under the PSIM simulator.

powerpcle-*-eabi

Embedded PowerPC system in little endian mode.

rl78-*-elf

The Renesas RL78 processor. This configuration is intended for embedded systems.

riscv32-*-elf

The RISC-V RV32 instruction set. This configuration is intended for embedded systems. This (and all other RISC-V) targets require the binutils 2.30 release.

riscv32-*-linux

The RISC-V RV32 instruction set running GNU/Linux. This (and all other RISC-V) targets require the binutils 2.30 release.

riscv64-*-elf

The RISC-V RV64 instruction set. This configuration is intended for embedded systems. This (and all other RISC-V) targets require the binutils 2.30 release.

riscv64-*-linux

The RISC-V RV64 instruction set running GNU/Linux. This (and all other RISC-V) targets require the binutils 2.30 release.

rx-*-elf

The Renesas RX processor.

s390-*-linux*

S/390 system running GNU/Linux for S/390.

s390x-*-linux*

zSeries system (64-bit) running GNU/Linux for zSeries.

s390x-ibm-tpf*

zSeries system (64-bit) running TPF. This platform is supported as cross-compilation target only.

--solaris2*

Support for Solaris 10 has been removed in GCC 10. Support for Solaris 9 has been removed in GCC 5. Support for Solaris 8 has been removed in GCC 4.8. Support for Solaris 7 has been removed in GCC 4.6.

Solaris 11.3 provides GCC 4.5.2, 4.7.3, and 4.8.2 as `/usr/gcc/4.5/bin/gcc` or similar. Newer Solaris versions provide one or more of GCC 5, 7, and 9. Alternatively, you can install a pre-built GCC to bootstrap and install GCC. See the binaries page for details.

The Solaris 2 `/bin/sh` will often fail to configure '`libstdc++-v3`'. We therefore recommend using the following initial sequence of commands

```
% CONFIG_SHELL=/bin/ksh
% export CONFIG_SHELL
```

and proceed as described in the configure instructions. In addition we strongly recommend specifying an absolute path to invoke *srcdir*`/configure`.

In Solaris 11, you need to check for `system/header`, `system/linker`, and `developer/assembler` packages.

Trying to use the linker and other tools in '`/usr/ucb`' to install GCC has been observed to cause trouble. For example, the linker may hang indefinitely. The fix is to remove '`/usr/ucb`' from your `PATH`.

The build process works more smoothly with the legacy Solaris tools so, if you have '`/usr/xpg4/bin`' in your `PATH`, we recommend that you place '`/usr/bin`' before '`/usr/xpg4/bin`' for the duration of the build.

We recommend the use of the Solaris assembler or the GNU assembler, in conjunction with the Solaris linker. The GNU `as` versions included in Solaris 11.3, from GNU binutils 2.23.1 or newer (in '`/usr/bin/gas`' and '`/usr/gnu/bin/as`'), are known to work. The current version, from GNU binutils 2.34, is known to work as well. Note that your mileage may vary if you use a combination of the GNU tools and the Solaris tools: while the combination GNU `as` + Solaris `ld` should reasonably work, the reverse combination Solaris `as` + GNU `ld` may fail to build or cause memory corruption at runtime in some cases for C++ programs. GNU `ld` usually works as well. Again, the current version (2.34) is known to work, but generally lacks platform specific features, so better stay with Solaris `ld`. To use the LTO linker plugin ('`-fuse-linker-plugin`') with GNU `ld`, GNU binutils *must* be configured with '`--enable-largefile`'.

To enable symbol versioning in '`libstdc++`' with the Solaris linker, you need to have any version of GNU `c++filt`, which is part of GNU binutils. '`libstdc++`' symbol versioning will be disabled if no appropriate version is found. Solaris `c++filt` from the Solaris Studio compilers does *not* work.

The versions of the GNU Multiple Precision Library (GMP), the MPFR library and the MPC library bundled with Solaris 11.3 and later are usually recent enough to match GCC's requirements. There are two caveats:

- While the version of the GMP library in Solaris 11.3 works with GCC, you need to configure with '`--with-gmp-include=/usr/include/gmp`'.
- The version of the MPFR libary included in Solaris 11.3 is too old; you need to provide a more recent one.

sparc*-*-*

This section contains general configuration information for all SPARC-based platforms. In addition to reading this section, please read all other sections that match your target.

Newer versions of the GNU Multiple Precision Library (GMP), the MPFR library and the MPC library are known to be miscompiled by earlier versions of GCC on these platforms. We therefore recommend the use of the exact versions of these libraries listed as minimal versions in the prerequisites.

sparc-sun-solaris2*

When GCC is configured to use GNU binutils 2.14 or later, the binaries produced are smaller than the ones produced using Solaris native tools; this difference is quite significant for binaries containing debugging information.

Starting with Solaris 7, the operating system is capable of executing 64-bit SPARC V9 binaries. GCC 3.1 and later properly supports this; the '`-m64`' option enables 64-bit code generation. However, if all you want is code tuned for the UltraSPARC CPU, you should try the '`-mtune=ultrasparc`' option instead, which produces code that, unlike full 64-bit code, can still run on non-UltraSPARC machines.

When configuring the GNU Multiple Precision Library (GMP), the MPFR library or the MPC library on a Solaris 7 or later system, the canonical target triplet must be specified as the `build` parameter on the configure line. This target triplet can be obtained by invoking `./config.guess` in the toplevel source directory of GCC (and not that of GMP or MPFR or MPC). For example on a Solaris 11 system:

```
% ./configure --build=sparc-sun-solaris2.11 --prefix=xxx
```

sparc-*-linux*

sparc64-*-solaris2*

When configuring a 64-bit-default GCC on Solaris/SPARC, you must use a build compiler that generates 64-bit code, either by default or by specifying '`CC='gcc -m64' CXX='gcc-m64''` to `configure`. Additionally, you *must* pass '`--build=sparc64-sun-solaris2.11`' or '`--build=sparcv9-sun-solaris2.11`' because '`config.guess`' misdetects this situation, which can cause build failures.

When configuring the GNU Multiple Precision Library (GMP), the MPFR library or the MPC library, the canonical target triplet must be specified as the `build` parameter on the configure line. For example on a Solaris 11 system:

```
% ./configure --build=sparc64-sun-solaris2.11 --prefix=xxx
```

sparcv9-*-solaris2*

This is a synonym for '`sparc64-*-solaris2*`'.

c6x-*-*

The C6X family of processors. This port requires binutils-2.22 or newer.

tilegx-*-linux*

The TILE-Gx processor in little endian mode, running GNU/Linux. This port requires binutils-2.22 or newer.

tilegxbe-*-linux*

The TILE-Gx processor in big endian mode, running GNU/Linux. This port requires binutils-2.23 or newer.

tilepro-*-linux*

The TILEPro processor running GNU/Linux. This port requires binutils-2.22 or newer.

visium-*-elf

CDS VISIUMcore processor. This configuration is intended for embedded systems.

--vxworks*

Support for VxWorks is in flux. At present GCC supports *only* the very recent VxWorks 5.5 (aka Tornado 2.2) release, and only on PowerPC. We welcome patches for other architectures supported by VxWorks 5.5. Support for VxWorks AE would also be welcome; we believe this is merely a matter of writing an appropriate "configlette" (see below). We are not interested in supporting older, a.out or COFF-based, versions of VxWorks in GCC 3.

VxWorks comes with an older version of GCC installed in '*$WIND_BASE*/host'; we recommend you do not overwrite it. Choose an installation *prefix* entirely outside *$WIND_BASE*. Before running `configure`, create the directories '*prefix*' and '*prefix*/bin'. Link or copy the appropriate assembler, linker, etc. into '*prefix*/bin', and set your *PATH* to include that directory while running both `configure` and `make`.

You must give `configure` the '`--with-headers=`*$WIND_BASE*`/target/h`' switch so that it can find the VxWorks system headers. Since VxWorks is a cross compilation target only, you must also specify '`--target=`*target*'. `configure` will attempt to create the directory '*prefix*/*target*/sys-include' and copy files into it; make sure the user running `configure` has sufficient privilege to do so.

GCC's exception handling runtime requires a special "configlette" module, '`contrib/gthr_supp_vxw_5x.c`'. Follow the instructions in that file to add the module to your kernel build. (Future versions of VxWorks will incorporate this module.)

x86_64-*-*, amd64-*-*

GCC supports the x86-64 architecture implemented by the AMD64 processor (amd64-*-* is an alias for x86_64-*-*) on GNU/Linux, FreeBSD and NetBSD. On GNU/Linux the default is a bi-arch compiler which is able to generate both 64-bit x86-64 and 32-bit x86 code (via the '`-m32`' switch).

x86_64-*-solaris2*

GCC also supports the x86-64 architecture implemented by the AMD64 processor ('`amd64-*-*`' is an alias for '`x86_64-*-*`') on Solaris 10 or later. Unlike other systems, without special options a bi-arch compiler is built which generates 32-bit code by default, but can generate 64-bit x86-64 code with the '`-m64`' switch. Since GCC 4.7, there is also a configuration that defaults to 64-bit code, but can generate 32-bit code with '`-m32`'. To configure and build this way, you have to provide all support libraries like '`libgmp`' as 64-bit code, configure with '`--target=x86_64-pc-solaris2.11`' and '`CC=gcc -m64`'.

xtensa*-*-elf

This target is intended for embedded Xtensa systems using the '`newlib`' C library. It uses ELF but does not support shared objects. Designed-defined instructions specified via the Tensilica Instruction Extension (TIE) language are only supported through inline assembly.

The Xtensa configuration information must be specified prior to building GCC. The '`include/xtensa-config.h`' header file contains the configuration information. If you created your own Xtensa configuration with the Xtensa Processor Generator, the downloaded files include a customized copy of this header file, which you can use to replace the default header file.

xtensa*-*-linux*

This target is for Xtensa systems running GNU/Linux. It supports ELF shared objects and the GNU C library (glibc). It also generates position-independent code (PIC) regardless of whether the '`-fpic`' or '`-fPIC`' options are used. In other respects, this target is the same as the '`xtensa*-*-elf`' target.

Microsoft Windows

Intel 16-bit versions

The 16-bit versions of Microsoft Windows, such as Windows 3.1, are not supported.

However, the 32-bit port has limited support for Microsoft Windows 3.11 in the Win32s environment, as a target only. See below.

Intel 32-bit versions

The 32-bit versions of Windows, including Windows 95, Windows NT, Windows XP, and Windows Vista, are supported by several different target platforms. These targets differ in which Windows subsystem they target and which C libraries are used.

- Cygwin *-*-cygwin: Cygwin provides a user-space Linux API emulation layer in the Win32 subsystem.
- MinGW *-*-mingw32: MinGW is a native GCC port for the Win32 subsystem that provides a subset of POSIX.
- MKS i386-pc-mks: NuTCracker from MKS. See `https://www.mkssoftware.com` for more information.

Intel 64-bit versions

GCC contains support for x86-64 using the mingw-w64 runtime library, available from `http://mingw-w64.org/doku.php`. This library should be used with the target triple x86_64-pc-mingw32.

Presently Windows for Itanium is not supported.

Windows CE

Windows CE is supported as a target only on Hitachi SuperH (sh-wince-pe), and MIPS (mips-wince-pe).

Other Windows Platforms

GCC no longer supports Windows NT on the Alpha or PowerPC.

GCC no longer supports the Windows POSIX subsystem. However, it does support the Interix subsystem. See above.

Old target names including *-*-winnt and *-*-windowsnt are no longer used.

PW32 (i386-pc-pw32) support was never completed, and the project seems to be inactive. See `http://pw32.sourceforge.net/` for more information.

UWIN support has been removed due to a lack of maintenance.

--cygwin

Ports of GCC are included with the Cygwin environment.

GCC will build under Cygwin without modification; it does not build with Microsoft's C++ compiler and there are no plans to make it do so.

The Cygwin native compiler can be configured to target any 32-bit x86 cpu architecture desired; the default is i686-pc-cygwin. It should be used with as up-to-date a version of binutils as possible; use either the latest official GNU binutils release in the Cygwin distribution, or version 2.20 or above if building your own.

--mingw32

GCC will build with and support only MinGW runtime 3.12 and later. Earlier versions of headers are incompatible with the new default semantics of `extern inline` in `-std=c99` and `-std=gnu99` modes.

Older systems

GCC contains support files for many older (1980s and early 1990s) Unix variants. For the most part, support for these systems has not been deliberately removed, but it has not been maintained for several years and may suffer from bitrot.

Starting with GCC 3.1, each release has a list of "obsoleted" systems. Support for these systems is still present in that release, but `configure` will fail unless the '`--enable-obsolete`' option is given. Unless a maintainer steps forward, support for these systems will be removed from the next release of GCC.

Support for old systems as hosts for GCC can cause problems if the workarounds for compiler, library and operating system bugs affect the cleanliness or maintainability of the rest of GCC. In some cases, to bring GCC up on such a system, if still possible with current GCC, may require first installing an old version of GCC which did work on that system, and using it to compile a more recent GCC, to avoid bugs in the vendor compiler. Old releases of GCC 1 and GCC 2 are available in the '`old-releases`' directory on the GCC mirror sites. Header bugs may generally be avoided using `fixincludes`, but bugs or deficiencies in libraries and the operating system may still cause problems.

Support for older systems as targets for cross-compilation is less problematic than support for them as hosts for GCC; if an enthusiast wishes to make such a target work again (including resurrecting any of the targets that never worked with GCC 2, starting from the

last version before they were removed), patches following the usual requirements would be likely to be accepted, since they should not affect the support for more modern targets.

For some systems, old versions of GNU binutils may also be useful, and are available from '`pub/binutils/old-releases`' on sourceware.org mirror sites.

Some of the information on specific systems above relates to such older systems, but much of the information about GCC on such systems (which may no longer be applicable to current GCC) is to be found in the GCC texinfo manual.

all ELF targets (SVR4, Solaris 2, etc.)

C++ support is significantly better on ELF targets if you use the GNU linker; duplicate copies of inlines, vtables and template instantiations will be discarded automatically.

10 Old installation documentation

Note most of this information is out of date and superseded by the previous chapters of this manual. It is provided for historical reference only, because of a lack of volunteers to merge it into the main manual.

Here is the procedure for installing GCC on a GNU or Unix system.

1. If you have chosen a configuration for GCC which requires other GNU tools (such as GAS or the GNU linker) instead of the standard system tools, install the required tools in the build directory under the names '`as`', '`ld`' or whatever is appropriate.

   Alternatively, you can do subsequent compilation using a value of the `PATH` environment variable such that the necessary GNU tools come before the standard system tools.

2. Specify the host, build and target machine configurations. You do this when you run the '`configure`' script.

   The *build* machine is the system which you are using, the *host* machine is the system where you want to run the resulting compiler (normally the build machine), and the *target* machine is the system for which you want the compiler to generate code.

   If you are building a compiler to produce code for the machine it runs on (a native compiler), you normally do not need to specify any operands to '`configure`'; it will try to guess the type of machine you are on and use that as the build, host and target machines. So you don't need to specify a configuration when building a native compiler unless '`configure`' cannot figure out what your configuration is or guesses wrong.

   In those cases, specify the build machine's *configuration name* with the '`--host`' option; the host and target will default to be the same as the host machine.

   Here is an example:

   ```
   ./configure --host=sparc-sun-sunos4.1
   ```

   A configuration name may be canonical or it may be more or less abbreviated.

   A canonical configuration name has three parts, separated by dashes. It looks like this: '*`cpu-company-system`*'. (The three parts may themselves contain dashes; '`configure`' can figure out which dashes serve which purpose.) For example, '`m68k-sun-sunos4.1`' specifies a Sun 3.

   You can also replace parts of the configuration by nicknames or aliases. For example, '`sun3`' stands for '`m68k-sun`', so '`sun3-sunos4.1`' is another way to specify a Sun 3.

   You can specify a version number after any of the system types, and some of the CPU types. In most cases, the version is irrelevant, and will be ignored. So you might as well specify the version if you know it.

   See Section 10.1 [Configurations], page 77, for a list of supported configuration names and notes on many of the configurations. You should check the notes in that section before proceeding any further with the installation of GCC.

10.1 Configurations Supported by GCC

Here are the possible CPU types:

1750a, a29k, alpha, arm, avr, c*n*, clipper, dsp16xx, elxsi, fr30, h8300, hppa1.0, hppa1.1, i370, i386, i486, i586, i686, i786, i860, i960, ip2k, m32r, m68000,

m68k, m88k, mcore, mips, mipsel, mips64, mips64el, mn10200, mn10300, ns32k, pdp11, powerpc, powerpcle, romp, rs6000, sh, sparc, sparclite, sparc64, v850, vax, we32k.

Here are the recognized company names. As you can see, customary abbreviations are used rather than the longer official names.

acorn, alliant, altos, apollo, apple, att, bull, cbm, convergent, convex, crds, dec, dg, dolphin, elxsi, encore, harris, hitachi, hp, ibm, intergraph, isi, mips, motorola, ncr, next, ns, omron, plexus, sequent, sgi, sony, sun, tti, unicom, wrs.

The company name is meaningful only to disambiguate when the rest of the information supplied is insufficient. You can omit it, writing just '*cpu-system*', if it is not needed. For example, '`vax-ultrix4.2`' is equivalent to '`vax-dec-ultrix4.2`'.

Here is a list of system types:

386bsd, aix, acis, amigaos, aos, aout, aux, bosx, bsd, clix, coff, ctix, cxux, dgux, dynix, ebmon, ecoff, elf, esix, freebsd, hms, genix, gnu, linux, linux-gnu, hiux, hpux, iris, irix, isc, luna, lynxos, mach, minix, msdos, mvs, netbsd, newsos, nindy, ns, osf, osfrose, ptx, riscix, riscos, rtu, sco, sim, solaris, sunos, sym, sysv, udi, ultrix, unicos, uniplus, unos, vms, vsta, vxworks, winnt, xenix.

You can omit the system type; then '`configure`' guesses the operating system from the CPU and company.

You can add a version number to the system type; this may or may not make a difference. For example, you can write '`bsd4.3`' or '`bsd4.4`' to distinguish versions of BSD. In practice, the version number is most needed for '`sysv3`' and '`sysv4`', which are often treated differently.

'`linux-gnu`' is the canonical name for the GNU/Linux target; however GCC will also accept '`linux`'. The version of the kernel in use is not relevant on these systems. A suffix such as '`libc1`' or '`aout`' distinguishes major versions of the C library; all of the suffixed versions are obsolete.

If you specify an impossible combination such as '`i860-dg-vms`', then you may get an error message from '`configure`', or it may ignore part of the information and do the best it can with the rest. '`configure`' always prints the canonical name for the alternative that it used. GCC does not support all possible alternatives.

Often a particular model of machine has a name. Many machine names are recognized as aliases for CPU/company combinations. Thus, the machine name '`sun3`', mentioned above, is an alias for '`m68k-sun`'. Sometimes we accept a company name as a machine name, when the name is popularly used for a particular machine. Here is a table of the known machine names:

3300, 3b1, 3b*n*, 7300, altos3068, altos, apollo68, att-7300, balance, convex-c*n*, crds, decstation-3100, decstation, delta, encore, fx2800, gmicro, hp7*nn*, hp8*nn*, hp9k2*nn*, hp9k3*nn*, hp9k7*nn*, hp9k8*nn*, iris4d, iris, isi68, m3230, magnum, merlin, miniframe, mmax, news-3600, news800, news, next, pbd, pc532, pmax, powerpc, powerpcle, ps2, risc-news, rtpc, sun2, sun386i, sun386, sun3, sun4, symmetry, tower-32, tower.

Remember that a machine name specifies both the cpu type and the company name.

GNU Free Documentation License

Version 1.3, 3 November 2008

Copyright © 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc.
http://fsf.org/

Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document *free* in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or non-commercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released

under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

   If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

   If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

   If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

   It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

   You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

   A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any,

be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their

titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

   You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

   The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

   In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

   You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

   You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

   A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

   If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

   Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

   If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

   You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

   However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

   Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

   Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

    The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

    Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

    "Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

    "CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

    "Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

    An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

    The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C)  year  your name.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.3
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover
Texts.  A copy of the license is included in the section entitled ``GNU
Free Documentation License''.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

```
with the Invariant Sections being list their titles, with
the Front-Cover Texts being list, and with the Back-Cover Texts
being list.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.