Installing GCC

For GCC version 4.2.3

(Sourcery G++ Lite 4.2-129)

Table of Contents

1 Installing GCC

The latest version of this document is always available at http://gcc.gnu.org/install/.

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 39. 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 C90 compiler
: Necessary to bootstrap GCC, although 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 2.95 or later) because source code for language frontends other than C might use GCC extensions.

GNAT
: In order to build the Ada compiler (GNAT) you must already have GNAT installed because portions of the Ada frontend are written in Ada (with GNAT extensions.) Refer to the Ada installation instructions for more specific information.

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.

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 FTP mirror sites.

GNU make version 3.79.1 (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.

GNU Multiple Precision Library (GMP) version 4.1 (or later)
: Necessary to build the Fortran frontend. If you do not have it installed 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`'.

MPFR Library version 2.2.1 (or later)
: Necessary to build the Fortran frontend. It can be downloaded from `http://www.mpfr.org/`. The version of MPFR that is bundled with GMP 4.1.x contains numerous bugs. Although GNU Fortran will appear to function with the buggy versions of MPFR, there are a few GNU Fortran bugs that will not be fixed when using this version. It is strongly recommended to upgrade to the recommended version of MPFR.

  The '`--with-mpfr`' configure option should be used if your MPFR Library is not installed in your default library search path. See also '`--with-mpfr-lib`' and '`--with-mpfr-include`'.

`jar`, or InfoZIP (`zip` and `unzip`)
: Necessary to build libgcj, the GCJ runtime.

Tools/packages necessary for modifying GCC

autoconf versions 2.13 and 2.59
GNU m4 version 1.4 (or later)
: Necessary when modifying '`configure.ac`', '`aclocal.m4`', etc. to regenerate '`configure`' and '`config.in`' files. Most directories require autoconf 2.59 (exactly), but the toplevel still requires autoconf 2.13 (exactly).

automake version 1.9.6
: 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 1.9.x series, which is currently 1.9.6. When regenerating a directory to a newer version, please update all the directories using an older 1.9.x 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`'.

GNU Bison version 1.28 (or later)

Berkeley `yacc` (`byacc`) is also reported to work other than for GCJ.

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

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

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 SVN repository. They are included in releases.

Texinfo version 4.4 (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 SVN 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.

SVN (any version)
SSH (any version)

Necessary to access the SVN repository. Public releases and weekly snapshots of the development sources are also available via FTP.

Perl version 5.6.1 (or later)

Necessary when regenerating '`Makefile`' dependencies in libiberty. Necessary when regenerating '`libiberty/functions.texi`'. Necessary when generating manpages from Texinfo manuals. Necessary when targetting Darwin, building libstdc++, and not using '`--disable-symvers`'. Used by various scripts to generate some files included in SVN (mainly Unicode-related and rarely changing) from source tables.

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 SVN and FTP tarballs compressed with `gzip` or `bzip2`. It is possible to download a full distribution or specific components.

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

The full distribution includes the C, C++, Objective-C, Fortran, Java, and Ada (in the case of GCC 3.1 and later) compilers. The full distribution also includes runtime libraries for C++, Objective-C, Fortran, and Java. In GCC 3.0 and later versions, the GNU compiler testsuites are also included in the full distribution.

If you choose to download specific components, you must download the core GCC distribution plus any language specific distributions you wish to use. The core distribution includes the C language front end as well as the shared components. Each language has a tarball which includes the language front end as well as the language runtime (when appropriate).

Unpack the core distribution as well as any language specific distributions in the same directory.

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.

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 via SVN, *srcdir* must refer to the top '`gcc`' directory, the one where the '`MAINTAINERS`' 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 than 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-versuffix=`*suffix*

Specify a string that identifies your organization. You may also 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; instead, it is appended to the existing version string.

The default value is the empty string.

`--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.

Target specification

- GCC has code to correctly determine the correct value for *target* for nearly all native systems. Therefore, we highly recommend you 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-coff, 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*'.

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

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

`--mandir=`*dirname*
: Specify the installation directory for manual pages. The default is '*prefix*`/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 is '*prefix*`/include/c++/`*version*'.

`--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.

`--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`', '`libjava`' and '`libobjc`'. 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.

`--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*'
- '`i386-`*any*`-sysv`'
- '`m68k-bull-sysv`'
- '`m68k-hp-hpux`'

- ‘m68000-hp-hpux’
- ‘m68000-att-sysv’
- ‘sparc-sun-solaris2.*any*’
- ‘sparc64-*any*-solaris2.*any*’

On the systems listed above (except for the HP-PA, the SPARC, for ISC on the 386, if you use the GNU assembler, you should also use the GNU linker (and specify ‘--with-gnu-ld’).

--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 Sun 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.

  On MIPS based systems and on Alphas, you must specify whether you want GCC to create the normal ECOFF debugging format, or to use BSD-style stabs passed through the ECOFF symbol table. The normal ECOFF debug format cannot fully handle languages other than C. BSD stabs format can handle other languages, but it only works with the GNU debugger GDB.

  Normally, GCC uses the ECOFF debugging format by default; if you prefer BSD stabs, specify ‘--with-stabs’ when you configure GCC.

No matter which default you choose when you configure GCC, the user can use the '`-gcoff`' and '`-gstabs+`' options to specify explicitly the debug format for a particular compilation.

'`--with-stabs`' is meaningful on the ISC system on the 386, also, if '`--with-gas`' is used. It selects use of stabs debugging information embedded in COFF output. This kind of debugging information supports C++ well; ordinary COFF debugging information does not.

'`--with-stabs`' is also meaningful on 386 systems running SVR4. It selects use of stabs debugging information embedded in ELF output. The C++ compiler currently (2.6.0) does not support the DWARF debugging information normally used on 386 SVR4 platforms; stabs provide a workable alternative. This requires gas and gdb, as the normal SVR4 tools can not generate or interpret stabs.

`--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`'):

  `arc-*-elf*`
  : biendian.

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

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

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

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

`--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++ and Java. 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++ and Java. The possibilities for *lib* are:

`aix` AIX thread support.

`dce` DCE thread support.

`gnat` Ada tasking support. For non-Ada programs, this setting is equivalent to '`single`'. When used in conjunction with the Ada run time, it causes GCC to use the same thread primitives as Ada uses. This option is necessary when using both Ada and the back end exception handling, which is the default for most Ada targets.

`mach` Generic MACH thread support, known to work on NeXTSTEP. (Please note that the file needed to support this configuration, '`gthr-mach.h`', is missing and thus this setting will cause a known bootstrap failure.)

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

`posix` Generic POSIX/Unix98 thread support.

`posix95` Generic POSIX/Unix95 thread support.

`rtems` RTEMS thread support.

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

`solaris` Sun Solaris 2 thread support.

`vxworks` VxWorks thread support.

`win32` Microsoft Win32 API thread support.

`nks` Novell Kernel Services 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`'.

`--with-cpu=`*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 ARM, i386, M68k, PowerPC, and SPARC.

`--with-schedule=`*cpu*
`--with-arch=`*cpu*
`--with-tune=`*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-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.

`--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-exit`' to be passed by default.

`--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.

`--disable-cpp`
: Specify that a user visible `cpp` program should not be installed.

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

`--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-maintainer-mode`
: The build rules that regenerate 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 could happen 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 SVN 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. This is currently supported by '`libgfortran`', '`libjava`', '`libmudflap`', '`libstdc++`', and '`libobjc`'.

`--with-java-home=`*dirname*
: This '`libjava`' option overrides the default value of the '`java.home`' system property. It is also used to set '`sun.boot.class.path`' to '*dirname*`/lib/rt.jar`'. By default '`java.home`' is set to '*prefix*' and '`sun.boot.class.path`' to '*datadir*`/java/libgcj-`*version*`.jar`'.

`--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`, `ada`, `c`, `c++`, `fortran`, `java`, `objc`, `obj-c++`, `treelang`. Building the Ada compiler has special requirements, see below. If you do not pass this flag, or specify the option `all`, then all default languages available in the '`gcc`' sub-tree will be configured. Ada, Objective-C++, and treelang are not default languages; the rest are. Re-defining `LANGUAGES` when calling '`make`' **does not** work anymore, as those language subdirectories might not have been configured!

`--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-libssp`
: Specify that the run-time libraries for stack smashing protection should not be built.

`--disable-libgomp`
: Specify that the run-time libraries used by GOMP should not be built.

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

`--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. Currently, this option only affects powerpc-linux and x86-linux.

`--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-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`
`--enable-checking=`*list*

When you specify this option, the compiler is built to perform internal consistency checks of the requested complexity. This does not change the generated code, but adds error checking within the compiler. This will slow down the compiler and may only work properly if you are building the compiler with GCC. This is '`yes`' by default when building from SVN or snapshots, but '`release`' for releases. More control over the checks may be had by specifying *list*. The categories of checks available are '`yes`' (most common checks '`assert,misc,tree,gc,rtlflag,runtime`'), '`no`' (no checks at all), '`all`' (all but '`valgrind`'), '`release`' (cheapest checks '`assert,runtime`') or '`none`' (same as '`no`'). Individual checks can be enabled with these flags '`assert`', '`fold`', '`gc`', '`gcac`' '`misc`', '`rtl`', '`rtlflag`', '`runtime`', '`tree`', and '`valgrind`'.

The '`valgrind`' check requires the external `valgrind` simulator, available from http://valgrind.org/. The '`rtl`', '`gcac`' and '`valgrind`' checks are very expensive. To disable all checking, '`--disable-checking`' or '`--enable-checking=none`' must be explicitly requested. Disabling assertions will make the compiler and runtime slightly faster but increase the risk of undetected internal errors causing wrong code to be generated.

`--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`'.

`--with-gc`
`--with-gc=`*choice*

With this option you can specify the garbage collector implementation used during the compilation process. *choice* can be one of '`page`' and '`zone`', where '`page`' is the default.

`--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`
`--disable-decimal-float`
: Enable (or disable) support for the C decimal floating point extension. This is enabled by default only on PowerPC GNU/Linux systems. Other systems may also support it, but require the user to specifically enable it.

`--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-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*'.

Cross-Compiler-Specific Options

The following options only apply to building cross compilers.

`--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 in there. 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.

`--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.

`--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-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 a '`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.

Fortran-Specific Options

The following options apply to the build of the Fortran front end.

`--with-gmp=`*pathname*
`--with-gmp-include=`*pathname*
`--with-gmp-lib=`*pathname*
`--with-mpfr=`*pathname*
`--with-mpfr-include=`*pathname*
`--with-mpfr-lib=`*pathname*

If you do not have GMP (the GNU Multiple Precision library) and the MPFR Libraries installed in a standard location and you want to build the Fortran front-end, you can explicitly specify the directory where they are installed (‘`--with-gmp=`*gmpinstalldir*’, ‘`--with-mpfr=`*mpfrinstalldir*’). 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*`/includ`
If these shorthand assumptions are not correct, you can use the explicit include and lib options directly.

Java-Specific Options

The following option applies to the build of the Java front end.

`--disable-libgcj`

Specify that the run-time libraries used by GCJ should not be built. This is useful in case you intend to use GCJ with some other run-time, or you’re going to install it separately, or it just happens not to build on your particular machine. In general, if the Java front end is enabled, the GCJ libraries will be enabled too, unless they’re known to not work on the target platform. If GCJ is enabled but ‘`libgcj`’ isn’t built, you may need to port it; in this case, before modifying the top-level ‘`configure.in`’ so that ‘`libgcj`’ is enabled by default on this platform, you may use ‘`--enable-libgcj`’ to override the default.

The following options apply to building ‘`libgcj`’.

General Options

`--disable-getenv-properties`

Don’t set system properties from `GCJ_PROPERTIES`.

`--enable-hash-synchronization`

Use a global hash table for monitor locks. Ordinarily, ‘`libgcj`’’s ‘`configure`’ script automatically makes the correct choice for this option for your platform. Only use this if you know you need the library to be configured differently.

`--enable-interpreter`
: Enable the Java interpreter. The interpreter is automatically enabled by default on all platforms that support it. This option is really only useful if you want to disable the interpreter (using '`--disable-interpreter`').

`--disable-java-net`
: Disable java.net. This disables the native part of java.net only, using non-functional stubs for native method implementations.

`--disable-jvmpi`
: Disable JVMPI support.

`--with-ecos`
: Enable runtime eCos target support.

`--without-libffi`
: Don't use '`libffi`'. This will disable the interpreter and JNI support as well, as these require '`libffi`' to work.

`--enable-libgcj-debug`
: Enable runtime debugging code.

`--enable-libgcj-multifile`
: If specified, causes all '`.java`' source files to be compiled into '`.class`' files in one invocation of '`gcj`'. This can speed up build time, but is more resource-intensive. If this option is unspecified or disabled, '`gcj`' is invoked once for each '`.java`' file to compile into a '`.class`' file.

`--with-libiconv-prefix=DIR`
: Search for libiconv in '`DIR/include`' and '`DIR/lib`'.

`--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.

`--with-system-zlib`
: Use installed '`zlib`' rather than that included with GCC.

`--with-win32-nlsapi=ansi, unicows or unicode`
: Indicates how MinGW '`libgcj`' translates between UNICODE characters and the Win32 API.

  `ansi` — Use the single-byte `char` and the Win32 A functions natively, translating to and from UNICODE when using these functions. If unspecified, this is the default.

  `unicows` — Use the `WCHAR` and Win32 W functions natively. Adds `-lunicows` to '`libgcj.spec`' to link with '`libunicows`'. '`unicows.dll`' needs to be deployed on Microsoft Windows 9X machines running built executables. '`libunicows.a`', an open-source import library around Microsoft's `unicows.dll`, is obtained from http://libunicows.sourceforge.net/, which also gives details on getting '`unicows.dll`' from Microsoft.

`unicode` Use the `WCHAR` and Win32 W functions natively. Does *not* add `-lunicows` to '`libgcj.spec`'. The built executables will only run on Microsoft Windows NT and above.

AWT-Specific Options

`--with-x` Use the X Window System.

`--enable-java-awt=PEER(S)`
Specifies the AWT peer library or libraries to build alongside '`libgcj`'. If this option is unspecified or disabled, AWT will be non-functional. Current valid values are '`gtk`' and '`xlib`'. Multiple libraries should be separated by a comma (i.e. '`--enable-java-awt=gtk,xlib`').

`--enable-gtk-cairo`
Build the cairo Graphics2D implementation on GTK.

`--enable-java-gc=TYPE`
Choose garbage collector. Defaults to '`boehm`' if unspecified.

`--disable-gtktest`
Do not try to compile and run a test GTK+ program.

`--disable-glibtest`
Do not try to compile and run a test GLIB program.

`--with-libart-prefix=PFX`
Prefix where libart is installed (optional).

`--with-libart-exec-prefix=PFX`
Exec prefix where libart is installed (optional).

`--disable-libarttest`
Do not try to compile and run a test libart program.

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.

When building from SVN or snapshots, or if you modify parser sources, you need the Bison parser generator installed. If you do not modify parser sources, releases contain the Bison-generated files and you do not need Bison installed to build them.

When building from SVN or snapshots, or if you modify Texinfo documentation, you need version 4.4 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 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 CFLAGS='-O' LIBCFLAGS='-g -O2' \
  LIBCXXFLAGS='-g -O2 -fno-implicit-templates' bootstrap
```

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`'. Non-default optimization flags 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.

Note that using non-standard `CFLAGS` can cause bootstrap to fail if these trigger a warning with the new compiler. For example using '`-O2 -g -mcpu=i686`' on `i686-pc-linux-gnu` will cause bootstrap failure as '`-mcpu=`' is deprecated in 3.4.0 and above.

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.

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 first recommend 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 3.79 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

In order to build GNAT, the Ada compiler, you need a working GNAT compiler (GNAT version 3.14 or later, or GCC version 3.1 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 suggested to install the new compiler as native first, and then use it to build the cross compiler.

`configure` does not test whether the GNAT installation works and has a sufficiently recent version; if too old a GNAT version is installed, the build will fail unless '`--enable-languages`' is used to disable building the Ada front end.

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. Then runtime libraries are compiled with profile collected. 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. Also parallel make is currently not supported since collisions in profile collecting may occur.

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.

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.

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 '`make check-g++`' 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 '`*.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/-fno-strength-reduce"
```

will run the standard `g++` testsuites ("unix" is the target name for a standard native testsuite situation), passing '`-O3 -fno-strength-reduce`' 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
--target_board=arm-sim/-mhard-float/-O2
--target_board=arm-sim/-mhard-float/-O3
--target_board=arm-sim/-mhard-float
--target_board=arm-sim/-msoft-float/-O1
--target_board=arm-sim/-msoft-float/-O2
--target_board=arm-sim/-msoft-float/-O3
--target_board=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-reduce}{-fomit-frame-pointer,}"
```

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 Additional testing for Java Class Libraries

The Java runtime tests can be executed via '`make check`' in the '*target*`/libjava/testsuite`' directory in the build tree.

The Mauve Project provides a suite of tests for the Java Class Libraries. This suite can be run as part of libgcj testing by placing the Mauve tree within the libjava testsuite at '`libjava/testsuite/libjava.mauve/mauve`', or by specifying the location of that tree when invoking '`make`', as in '`make MAUVEDIR=~/mauve check`'.

Jacks is a free testsuite that tests Java compiler front ends. This suite can be run as part of libgcj testing by placing the Jacks tree within the libjava testsuite at '`libjava/testsuite/libjava.jacks/jacks`'.

6.4 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.

6.5 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.

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++ and Java libraries 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.

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 39 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 4.4) 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 Freeware and Shareware Archive for AIX;
  - UCLA Software Library for AIX.
- DOS—DJGPP.
- Renesas H8/300[HS]—GNU Development Tools for the Renesas H8/300[HS] Series.
- HP-UX:
  - HP-UX Porting Center;
  - Binaries for HP-UX 11.00 at Aachen University of Technology.
- Motorola 68HC11/68HC12—GNU Development Tools for the Motorola 68HC11/68HC12.
- SCO OpenServer/Unixware.
- Solaris 2 (SPARC, Intel)—Sunfreeware.
- SGI—SGI Freeware.
- Microsoft Windows:
  - The Cygwin project;
  - The MinGW project.
- The Written Word offers binaries for AIX 4.3.2. IRIX 6.5, Digital UNIX 4.0D and 5.1, GNU/Linux (i386), HP-UX 10.20, 11.00, and 11.11, and Solaris/SPARC 2.5.1, 2.6, 7, 8, and 9.
- OpenPKG offers binaries for quite a number of platforms.
- The GFortran Wiki has links to GNU Fortran binaries for several platforms.

In addition to those specific offerings, you can get a binary distribution CD-ROM from the Free Software Foundation. It contains binaries for a number of platforms, and includes not only GCC, but other stuff as well. The current CD does not contain the latest version of GCC, but it should allow bootstrapping the compiler. An updated version of that disk is in the works.

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 are.

alpha*-*-*

This section contains general configuration information for all alpha-based platforms using ELF (in particular, ignore this section for DEC OSF/1, Digital UNIX and Tru64 UNIX). In addition to reading this section, please read all other sections that match your target.

We require binutils 2.11.2 or newer. Previous binutils releases had a number of problems with DWARF 2 debugging information, not the least of which is incorrect linking of shared libraries.

alpha*-dec-osf*

Systems using processors that implement the DEC Alpha architecture and are running the DEC/Compaq Unix (DEC OSF/1, Digital UNIX, or Compaq Tru64 UNIX) operating system, for example the DEC Alpha AXP systems.

As of GCC 3.2, versions before `alpha*-dec-osf4` are no longer supported. (These are the versions which identify themselves as DEC OSF/1.)

In Digital Unix V4.0, virtual memory exhausted bootstrap failures may be fixed by configuring with '`--with-gc=simple`', reconfiguring Kernel Virtual Memory and Swap parameters per the `/usr/sbin/sys_check` Tuning Suggestions, or applying the patch in http://gcc.gnu.org/ml/gcc/2002-08/msg00822.html.

In Tru64 UNIX V5.1, Compaq introduced a new assembler that does not currently (2001-06-13) work with `mips-tfile`. As a workaround, we need to use the old assembler, invoked via the barely documented '`-oldas`' option. To bootstrap GCC, you either need to use the Compaq C Compiler:

```
% CC=cc srcdir/configure [options] [target]
```

or you can use a copy of GCC 2.95.3 or higher built on Tru64 UNIX V4.0:

```
% CC=gcc -Wa,-oldas srcdir/configure [options] [target]
```

As of GNU binutils 2.11.2, neither GNU `as` nor GNU `ld` are supported on Tru64 UNIX, so you must not configure GCC with '`--with-gnu-as`' or '`--with-gnu-ld`'.

GCC writes a '`.verstamp`' directive to the assembler output file unless it is built as a cross-compiler. It gets the version to use from the system header file '`/usr/include/stamp.h`'. If you install a new version of DEC Unix, you should rebuild GCC to pick up the new version stamp.

Note that since the Alpha is a 64-bit architecture, cross-compilers from 32-bit machines will not generate code as efficient as that generated when the compiler is running on a 64-bit machine because many optimizations that depend on being able to represent a word on the target in an integral value on the host cannot be performed. Building cross-compilers on the Alpha for 32-bit machines has only been tested in a few cases and may not work properly.

'`make compare`' may fail on old versions of DEC Unix unless you add '`-save-temps`' to `CFLAGS`. On these systems, the name of the assembler input file is stored in the object file, and that makes comparison fail if it differs between the `stage1` and `stage2` compilations. The option '`-save-temps`' forces a fixed name to be used for the assembler input file, instead of a randomly chosen name in '`/tmp`'. Do not add '`-save-temps`' unless the comparisons fail without that option. If you add '`-save-temps`', you will have to manually delete the '`.i`' and '`.s`' files after each series of compilations.

GCC now supports both the native (ECOFF) debugging format used by DBX and GDB and an encapsulated STABS format for use only with GDB. See the discussion of the '`--with-stabs`' option of '`configure`' above for more information on these formats and how to select them.

There is a bug in DEC's assembler that produces incorrect line numbers for ECOFF format when the '`.align`' directive is used. To work around this problem, GCC will not emit such alignment directives while writing ECOFF format debugging information even if optimization is being performed. Unfortunately, this has the very undesirable side-effect that code addresses when '`-O`' is specified are different depending on whether or not '`-g`' is also specified.

To avoid this behavior, specify '`-gstabs+`' and use GDB instead of DBX. DEC is now aware of this problem with the assembler and hopes to provide a fix shortly.

alphaev5-cray-unicosmk*

Cray T3E systems running Unicos/Mk.

This port is incomplete and has many known bugs. We hope to improve the support for this target soon. Currently, only the C front end is supported, and it is not possible to build parallel applications. Cray modules are not supported; in particular, Craylibs are assumed to be in '`/opt/ctl/craylibs/craylibs`'.

On this platform, you need to tell GCC where to find the assembler and the linker. The simplest way to do so is by providing '`--with-as`' and '`--with-ld`' to '`configure`', e.g.

```
configure --with-as=/opt/ctl/bin/cam --with-ld=/opt/ctl/bin/cld \
  --enable-languages=c
```

The comparison test at the end of the bootstrapping process fails on Unicos/Mk because the assembler inserts timestamps into object files. You should be able to work around this by doing '`make all`' after getting this failure.

arc-*-elf

Argonaut ARC processor. This configuration is intended for embedded systems.

arm-*-elf

xscale-*-*

ARM-family processors. Subtargets that use the ELF object format require GNU binutils 2.13 or newer. Such subtargets include: `arm-*-freebsd`, `arm-*-netbsdelf`, `arm-*-*linux`, `arm-*-rtems` and `arm-*-kaos`.

arm-*-coff

ARM-family processors. Note that there are two different varieties of PE format subtarget supported: `arm-wince-pe` and `arm-pe` as well as a standard COFF target `arm-*-coff`.

arm-*-aout

ARM-family processors. These targets support the AOUT file format: `arm-*-aout`, `arm-*-netbsd`.

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://home.overta.ru/users/denisc/
- http://www.amelek.gda.pl/avr/

We *strongly* recommend using binutils 2.13 or newer.

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, is available at `http://blackfin.uclinux.org`

c4x

Texas Instruments TMS320C3x and TMS320C4x Floating Point Digital Signal Processors. These are used in embedded applications. There are no standard Unix configurations. See Section "TMS320C3x/C4x Options" in *Using the GNU Compiler Collection (GCC)*, for the list of supported MCU types.

GCC can be configured as a cross compiler for both the C3x and C4x architectures on the same system. Use '`configure --target=c4x --enable-languages="c,c++"`' to configure.

Further installation notes and other useful information about C4x tools can also be obtained from:

- http://www.elec.canterbury.ac.nz/c4x/

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-aout`
: Old target. Includes a multilib for the '`elinux`' a.out-based target. No multilibs for newer architecture variants.

`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.

For `cris-axis-aout` and `cris-axis-elf` you need binutils 2.11 or newer. For `cris-axis-linux-gnu` you need binutils 2.12 or newer.

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/`.

CRX

The CRX CompactRISC architecture is a low-power 32-bit architecture with fast context switching and architectural extensibility features.

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

Use '`configure --target=crx-elf --enable-languages=c,c++`' to configure GCC for building a CRX cross-compiler. The option '`--target=crx-elf`' is also used to build the '`newlib`' C library for CRX.

It is also possible to build libstdc++-v3 for the CRX architecture. This needs to be done in a separate step with the following configure settings: '`gcc/libstdc++-v3/configure --host=crx-elf --with-newlib --enable-sjlj-exceptions --enable-cxx-flags='-fexceptions -frtti''`

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.

--freebsd*

The version of binutils installed in '`/usr/bin`' probably works with this release of GCC. However, on FreeBSD 4, bootstrapping against the latest FSF binutils is known to improve overall testsuite results; and, on FreeBSD/alpha, using binutils 2.14 or later is required to build libjava.

Support for FreeBSD 1 was discontinued in GCC 3.2.

Support for FreeBSD 2 will be discontinued after GCC 3.4. The following was true for GCC 3.1 but the current status is unknown. For FreeBSD 2 or any mutant a.out versions of FreeBSD 3: All configuration support and files as shipped with GCC 2.95 are still in place. FreeBSD 2.2.7 has been known to bootstrap completely; however, it is unknown which version of binutils was used (it is assumed that it was the system copy in '`/usr/bin`') and C++ EH failures were noted.

For FreeBSD using the ELF file format: DWARF 2 debugging is now the default for all CPU architectures. It had been the default on FreeBSD/alpha since its inception. 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 4.9-STABLE and 5-CURRENT. 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-STABLE.

In principle, '`--enable-threads`' is now compatible with '`--enable-libgcj`' on FreeBSD. However, it has only been built and tested on '`i386-*-freebsd[45]`' and '`alpha-*-freebsd[45]`'. The static library may be incorrectly built (symbols are missing at link time). There is a rare timing-based startup hang (probably involves an assumption about the thread library). Multi-threaded boehm-gc (required for libjava) exposes severe threaded signal-handling bugs on FreeBSD before 4.5-RELEASE. Other CPU architectures supported by FreeBSD will require additional configuration tuning in, at the very least, both boehm-gc and libffi.

Shared '`libgcc_s.so`' is now built and installed by default.

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; you may encounter a variety of problems if you try to use the HP assembler.

Specifically, '`-g`' does not work on HP-UX (since that system uses a peculiar debugging format which GCC does not know about), unless you use GAS and GDB. It may be helpful to configure GCC with the '`--with-gnu-as`' and '`--with-as=...`' options to ensure that GCC can find GAS.

If you wish to use the pa-risc 2.0 architecture support with a 32-bit runtime, you must use gas/binutils 2.11 or newer.

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.

As of GCC 4.1, `DWARF2` exception handling is available on HP-UX. It is now the default. This exposed a bug in the handling of data relocations in the GAS assembler. The handling of 64-bit data relocations was seriously broken, affecting debugging and exception support on all '`hppa64-*-*`' targets. Under some circumstances, 32-bit data relocations could also be handled incorrectly. This problem is fixed in GAS version 2.16.91 20051125.

GCC versions prior to 4.1 incorrectly passed and returned complex values. They are now passed in the same manner as aggregates.

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. HP has two sites which provide patches free of charge:

- `http://us.itrc.hp.com/service/home/home.do` US, Canada, Asia-Pacific, and Latin-America.
- `http://europe.itrc.hp.com/service/home/home.do` Europe.

The HP assembler on these systems has some problems. Most notably the assembler 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`'.

GCC 4.0 requires CVS binutils as of April 28, 2004 or later. Earlier versions require binutils 2.8 or later.

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.

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 can't be bootstrapped using C. Ada is only available for the 32-bit PA-RISC runtime. The libffi and libjava haven't been ported to HP-UX and don't build.

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. The fastjar program contains ISO C code and can't be built with the HP bundled compiler. This problem can be avoided by not building the Java language. For example, use the '`--enable-languages="c,c++,f77,objc"`' option in your configure command.

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 HP and GNU linkers are both supported for this target.

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 been reported to sometimes occur in unified builds of binutils and GCC.

GCC 3.0 through 3.2 require binutils 2.11 or above. GCC 3.3 through GCC 4.0 require binutils 2.14 or later.

Although the HP assembler can be used for an initial build, it shouldn't be used with any languages other than C and perhaps Fortran due to its many limitations. 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. You can't generate debugging information when using the HP assembler. Finally, bootstrapping fails in the final comparison of object modules due to the time stamps that it inserts into the modules. The bootstrap can be continued from this point with '`make all-host all-target`'.

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.

There are a number of issues to consider in selecting which linker to use with the 64-bit port. The GNU 64-bit linker can only create dynamic binaries. The '`-static`' option causes linking with archive libraries but doesn't produce a truly static binary. Dynamic binaries still require final binding by the dynamic loader to resolve a set of dynamic-loader-defined symbols. The default behavior of the HP linker is the same as the GNU linker. However, it can generate true 64-bit static binaries using the '`+compat`' option.

The HP 64-bit linker doesn't support linkonce semantics. As a result, C++ programs have many more sections than they should.

The GNU 64-bit linker has some issues with shared library support and exceptions. As a result, we only support libgcc in archive format. For similar reasons, dwarf2 unwind and exception support are disabled. The GNU linker also has problems creating binaries with '`-static`'. It doesn't provide stubs for internal calls to global functions in shared libraries, so these calls can't be overloaded.

Thread support is not implemented in GCC 3.0 through 3.2, so the '`--enable-threads`' configure option does not work. In 3.3 and later, POSIX threads are supported. The optional DCE thread library is not supported.

This port still is undergoing significant development.

--linux-gnu

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

i?86-*-linux*aout

Use this configuration to generate '`a.out`' binaries on Linux-based GNU systems. This configuration is being superseded.

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-*-sco3.2v5*

Use this for the SCO OpenServer Release 5 family of operating systems.

Unlike earlier versions of GCC, the ability to generate COFF with this target is no longer provided.

Earlier versions of GCC emitted DWARF 1 when generating ELF to allow the system debugger to be used. That support was too burdensome to maintain. GCC now emits only DWARF 2 for this target. This means you may use either the UDK debugger or GDB to debug programs built by this version of GCC.

GCC is now only supported on releases 5.0.4 and later, and requires that you install Support Level Supplement OSS646B or later, and Support Level Supplement OSS631C or later. If you are using release 5.0.7 of OpenServer, you must have at least the first maintenance pack installed (this includes the relevant portions of OSS646). OSS646, also known as the "Execution Environment Update", provides updated link editors and assemblers, as well as updated standard C and math libraries. The C startup modules are also updated to support the System V gABI draft, and GCC relies on that behavior. OSS631 provides a collection of commonly used open source libraries, some of which GCC depends on (such as GNU gettext and zlib). SCO OpenServer Release 5.0.7 has all of this built in by default, but OSS631C and later also apply to that release. Please visit ftp://ftp.sco.com/pub/openserver5 for the latest versions of these (and other potentially useful) supplements.

Although there is support for using the native assembler, it is recommended that you configure GCC to use the GNU assembler. You do this by using the flags '`--with-gnu-as`'. You should use a modern version of GNU binutils. Version 2.13.2.1 was used for all testing. In general, only the '`--with-gnu-as`' option is tested. A modern bintuils (as well as a plethora of other development related GNU utilities) can be found in Support Level

Supplement OSS658A, the "GNU Development Tools" package. See the SCO web and ftp sites for details. That package also contains the currently "officially supported" version of GCC, version 2.95.3. It is useful for bootstrapping this version.

i?86-*-solaris2.10

Use this for Solaris 10 or later on x86 and x86-64 systems. This configuration is supported by GCC 4.0 and later versions only.

It is recommended that you configure GCC to use the GNU assembler in '`/usr/sfw/bin/gas`' but the Sun linker, using the options '`--with-gnu-as --with-as=/usr/sfw/bin/gas --without-gnu-ld --with-ld=/usr/ccs/bin/ld`'.

i?86-*-udk

This target emulates the SCO Universal Development Kit and requires that package be installed. (If it is installed, you will have a '`/udk/usr/ccs/bin/cc`' file present.) It's very much like the '`i?86-*-unixware7*`' target but is meant to be used when hosting on a system where UDK isn't the default compiler such as OpenServer 5 or Unixware 2. This target will generate binaries that will run on OpenServer, Unixware 2, or Unixware 7, with the same warnings and caveats as the SCO UDK.

This target is a little tricky to build because we have to distinguish it from the native tools (so it gets headers, startups, and libraries from the right place) while making the tools not think we're actually building a cross compiler. The easiest way to do this is with a configure command like this:

```
CC=/udk/usr/ccs/bin/cc /your/path/to/gcc/configure \
  --host=i686-pc-udk --target=i686-pc-udk --program-prefix=udk-
```

You should substitute '`i686`' in the above command with the appropriate processor for your host.

After the usual '`make`' and '`make install`', you can then access the UDK-targeted GCC tools by adding `udk-` before the commonly known name. For example, to invoke the C compiler, you would use `udk-gcc`. They will coexist peacefully with any native-target GCC tools you may have installed.

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.

"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.

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 4 and required for bootstrapping on AIX 5L. The GNU Assembler reports that it supports WEAK symbols on AIX 4, which causes GCC to try to utilize weak symbol functionality although it is not supported. The GNU Assembler and Linker do not support AIX 5L sufficiently to bootstrap GCC. The native AIX tools do interoperate with GCC.

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 a 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
```

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`'.

By default, GCC for AIX 4.1 and above produces code that can be used on both Power or PowerPC processors.

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.

m32c-*-elf

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

m32r-*-elf

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

m6811-elf

Motorola 68HC11 family micro controllers. These are used in embedded applications. There are no standard Unix configurations.

m6812-elf

Motorola 68HC12 family micro controllers. These are used in embedded applications. There are no standard Unix configurations.

m68k-*-*

By default, '`m68k-*-aout`', '`m68k-*-coff*`', '`m68k-*-elf*`', '`m68k-*-rtems`' and '`m68k-*-uclinux`' 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 code 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 code for the 547x family of processors when configured with '`--with-arch=cf`' and 68020 code otherwise. (Note that code for the 547x family can also run on the 548x family.) The '`m68k-*-linux-gnu`' target is the same, but its ColdFire variant builds both 547x-compatible and 5445x-compatible libraries.

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`'.

m68k-hp-hpux

HP 9000 series 300 or 400 running HP-UX. HP-UX version 8.0 has a bug in the assembler that prevents compilation of GCC. This bug manifests itself during the first stage of compilation, while building '`libgcc2.a`':

```
_floatdisf
cc1: warning: '-g' option not supported on this version of GCC
cc1: warning: '-g1' option not supported on this version of GCC
./xgcc: Internal compiler error: program as got fatal signal 11
```

A patched version of the assembler is available as the file ftp://altdorf.ai.mit.edu/archive/cph/hpux-8 If you have HP software support, the patch can also be obtained directly from HP, as described in the following note:

> This is the patched assembler, to patch SR#1653-010439, where the assembler aborts on floating point constants.
>
> The bug is not really in the assembler, but in the shared library version of the function "cvtnum(3c)". The bug on "cvtnum(3c)" is SR#4701-078451. Anyway, the attached assembler uses the archive library version of "cvtnum(3c)" and thus does not exhibit the bug.

This patch is also known as PHCO_4484.

In addition gdb does not understand that native HP-UX format, so you must use gas if you wish to use gdb.

On HP-UX version 8.05, but not on 8.07 or more recent versions, the `fixproto` shell script triggers a bug in the system shell. If you encounter this problem, upgrade your operating system or use BASH (the GNU shell) to run `fixproto`. This bug will cause the fixproto program to report an error of the form:

```
./fixproto: sh internal 1K buffer overflow
```

To fix this, you can also change the first line of the fixproto script to look like:

```
#!/bin/ksh
```

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. However, you can still use the original ABI by configuring for '`m68k-uclinuxoldabi`' or '`m68k-`*`vendor`*`-uclinuxoldabi`'.

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.

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.

Cross-compilers for the MIPS as target using the MIPS assembler currently do not work, because the auxiliary programs '`mips-tdump.c`' and '`mips-tfile.c`' can't be compiled on anything but a MIPS. It does work to cross compile for a MIPS if you use the GNU assembler and linker.

The assembler from GNU binutils 2.17 and earlier has a bug in the way it sorts relocations for REL targets (o32, o64, EABI). This can cause bad code to be generated for simple C++ programs. Also the linker from GNU binutils versions prior to 2.17 has a bug which causes the runtime linker stubs in very large programs, like '`libgcj.so`', to be incorrectly generated. Binutils CVS snapshots and releases made after Nov. 9, 2006 are thought to be free from both of these problems.

mips-sgi-irix5

In order to compile GCC on an SGI running IRIX 5, the '`compiler_dev.hdr`' subsystem must be installed from the IDO CD-ROM supplied by SGI. It is also available for download from `ftp://ftp.sgi.com/sgi/IRIX5.3/iris-development-option-5.3.tardist`.

If you use the MIPS C compiler to bootstrap, it may be necessary to increase its table size for switch statements with the '`-Wf,-XNg1500`' option. If you use the '`-O2`' optimization option, you also need to use '`-Olimit 3000`'.

To enable debugging under IRIX 5, you must use GNU binutils 2.15 or later, and use the '`--with-gnu-ld`' `configure` option when configuring GCC. You need to use GNU `ar` and `nm`, also distributed with GNU binutils.

Some users have reported that `/bin/sh` will hang during bootstrap. This problem can be avoided by running the commands:

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

before starting the build.

mips-sgi-irix6

If you are using SGI's MIPSpro `cc` as your bootstrap compiler, you must ensure that the N32 ABI is in use. To test this, compile a simple C file with `cc` and then run `file` on the resulting object file. The output should look like:

```
test.o: ELF N32 MSB ...
```

If you see:

```
test.o: ELF 32-bit MSB ...
```

or

```
test.o: ELF 64-bit MSB ...
```

then your version of `cc` uses the O32 or N64 ABI by default. You should set the environment variable `CC` to '`cc -n32`' before configuring GCC.

If you want the resulting `gcc` to run on old 32-bit systems with the MIPS R4400 CPU, you need to ensure that only code for the '`mips3`' instruction set architecture (ISA) is

generated. While GCC 3.x does this correctly, both GCC 2.95 and SGI's MIPSpro `cc` may change the ISA depending on the machine where GCC is built. Using one of them as the bootstrap compiler may result in '`mips4`' code, which won't run at all on '`mips3`'-only systems. For the test program above, you should see:

```
test.o: ELF N32 MSB mips-3 ...
```

If you get:

```
test.o: ELF N32 MSB mips-4 ...
```

instead, you should set the environment variable `CC` to '`cc -n32 -mips3`' or '`gcc -mips3`' respectively before configuring GCC.

MIPSpro C 7.4 may cause bootstrap failures, due to a bug when inlining `memcmp`. Either add `-U__INLINE_INTRINSICS` to the `CC` environment variable as a workaround or upgrade to MIPSpro C 7.4.1m.

GCC on IRIX 6 is usually built to support the N32, O32 and N64 ABIs. If you build GCC on a system that doesn't have the N64 libraries installed or cannot run 64-bit binaries, you need to configure with '`--disable-multilib`' so GCC doesn't try to use them. This will disable building the O32 libraries, too. Look for '`/usr/lib64/libc.so.1`' to see if you have the 64-bit libraries installed.

To enable debugging for the O32 ABI, you must use GNU `as` from GNU binutils 2.15 or later. You may also use GNU `ld`, but this is not required and currently causes some problems with Ada.

The '`--enable-threads`' option doesn't currently work, a patch is in preparation for a future release. The '`--enable-libgcj`' option is disabled by default: IRIX 6 uses a very low default limit (20480) for the command line length. Although `libtool` contains a workaround for this problem, at least the N64 '`libgcj`' is known not to build despite this, running into an internal error of the native `ld`. A sure fix is to increase this limit ('`ncargs`') to its maximum of 262144 bytes. If you have root access, you can use the `systune` command to do this.

`wchar_t` support in '`libstdc++`' is not available for old IRIX 6.5.x releases, $x < 19$. The problem cannot be autodetected and in order to build GCC for such targets you need to configure with '`--disable-wchar_t`'.

See http://freeware.sgi.com/ for more information about using GCC on IRIX platforms.

powerpc-*-*

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

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 http://developer.apple.com/darwin/projects/compiler/ (free registration required).

This version of GCC requires at least cctools-590.7.

The version of GCC shipped by Apple typically includes a number of extensions not available in a standard GCC release. These extensions are generally for backwards compatibility and best avoided.

powerpc-*-elf, powerpc-*-sysv4

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

powerpc*-*-linux-gnu*

You will need binutils 2.15 or newer for a working GCC.

powerpc-*-netbsd*

PowerPC system in big endian mode running NetBSD. To build the documentation you will need Texinfo version 4.4 (NetBSD 1.5.1 included Texinfo version 3.12).

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, powerpcle-*-sysv4

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.

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*

Sun does not ship a C compiler with Solaris 2. To bootstrap and install GCC you first have to install a pre-built compiler, see the binaries page for details.

The Solaris 2 `/bin/sh` will often fail to configure '`libstdc++-v3`', '`boehm-gc`' or '`libjava`'. 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.

Solaris 2 comes with a number of optional OS packages. Some of these are needed to use GCC fully, namely `SUNWarc`, `SUNWbtool`, `SUNWesu`, `SUNWhea`, `SUNWlibm`, `SUNWsprot`, and `SUNWtoo`. If you did not install all optional packages when installing Solaris 2, you will need to verify that the packages that GCC needs are installed.

To check whether an optional package is installed, use the `pkginfo` command. To add an optional package, use the `pkgadd` command. For further details, see the Solaris 2 documentation.

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 Sun 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.

All releases of GNU binutils prior to 2.11.2 have known bugs on this platform. We recommend the use of GNU binutils 2.11.2 or later, or the vendor tools (Sun `as`, Sun `ld`). Note that your mileage may vary if you use a combination of the GNU tools and the Sun tools: while the combination GNU `as` + Sun `ld` should reasonably work, the reverse combination Sun `as` + GNU `ld` is known to cause memory corruption at runtime in some cases for C++ programs.

The stock GNU binutils 2.15 release is broken on this platform because of a single bug. It has been fixed on the 2.15 branch in the CVS repository. You can obtain a working version by checking out the binutils-2_15-branch from the CVS repository or applying the patch `http://sourceware.org/ml/binutils-cvs/2004-09/msg00036.html` to the release.

We recommend using GNU binutils 2.16 or later in conjunction with GCC 4.x, or the vendor tools (Sun `as`, Sun `ld`). However, for Solaris 10 and above, an additional patch is required in order for the GNU linker to be able to cope with a new flavor of shared libraries. You can obtain a working version by checking out the binutils-2_16-branch from the CVS repository or applying the patch `http://sourceware.org/ml/binutils-cvs/2005-07/msg00122.html` to the release.

Sun bug 4296832 turns up when compiling X11 headers with GCC 2.95 or newer: `g++` will complain that types are missing. These headers assume that omitting the type means `int`; this assumption worked for C89 but is wrong for C++, and is now wrong for C99 also.

`g++` accepts such (invalid) constructs with the option '`-fpermissive`'; it will assume that any missing type is `int` (as defined by C89).

There are patches for Solaris 2.6 (105633-56 or newer for SPARC, 106248-42 or newer for Intel), Solaris 7 (108376-21 or newer for SPARC, 108377-20 for Intel), and Solaris 8 (108652-24 or newer for SPARC, 108653-22 for Intel) that fix this bug.

Sun bug 4927647 sometimes causes random spurious testsuite failures related to missing diagnostic output. This bug doesn't affect GCC itself, rather it is a kernel bug triggered by the `expect` program which is used only by the GCC testsuite driver. When the bug causes the `expect` program to miss anticipated output, extra testsuite failures appear.

There are patches for Solaris 8 (117350-12 or newer for SPARC, 117351-12 or newer for Intel) and Solaris 9 (117171-11 or newer for SPARC, 117172-11 or newer for Intel) that address this problem.

sparc-sun-solaris2*

When GCC is configured to use binutils 2.11.2 or later the binaries produced are smaller than the ones produced using Sun's native tools; this difference is quite significant for binaries containing debugging information.

Sun `as` 4.x is broken in that it cannot cope with long symbol names. A typical error message might look similar to the following:

```
/usr/ccs/bin/as: "/var/tmp/ccMsw135.s", line 11041: error:
  can't compute value of an expression involving an external symbol.
```

This is Sun bug 4237974. This is fixed with patch 108908-02 for Solaris 2.6 and has been fixed in later (5.x) versions of the assembler, starting with Solaris 7.

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 on a Solaris 7 or later system that is running a kernel that supports only 32-bit binaries, one must configure with '`--disable-multilib`', since we will not be able to build the 64-bit target libraries.

GCC 3.3 and GCC 3.4 trigger code generation bugs in earlier versions of the GNU compiler (especially GCC 3.0.x versions), which lead to the miscompilation of the stage1 compiler and the subsequent failure of the bootstrap process. A workaround is to use GCC 3.2.3 as an intermediary stage, i.e. to bootstrap that compiler with the base compiler and then use it to bootstrap the final compiler.

GCC 3.4 triggers a code generation bug in versions 5.4 (Sun ONE Studio 7) and 5.5 (Sun ONE Studio 8) of the Sun compiler, which causes a bootstrap failure in form of a miscompilation of the stage1 compiler by the Sun compiler. This is Sun bug 4974440. This is fixed with patch 112760-07.

GCC 3.4 changed the default debugging format from STABS to DWARF-2 for 32-bit code on Solaris 7 and later. If you use the Sun assembler, this change apparently runs afoul of Sun bug 4910101 (which is referenced as a x86-only problem by Sun, probably because they do not use DWARF-2). A symptom of the problem is that you cannot compile C++ programs like `groff` 1.19.1 without getting messages similar to the following:

```
ld: warning: relocation error: R_SPARC_UA32: ...
  external symbolic relocation against non-allocatable section
```

```
.debug_info cannot be processed at runtime: relocation ignored.
```

To work around this problem, compile with '`-gstabs+`' instead of plain '`-g`'.

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

```
% ./configure --build=sparc-sun-solaris2.7 --prefix=xxx
```

sparc-sun-solaris2.7

Sun patch 107058-01 (1999-01-13) for Solaris 7/SPARC triggers a bug in the dynamic linker. This problem (Sun bug 4210064) affects GCC 2.8 and later, including all EGCS releases. Sun formerly recommended 107058-01 for all Solaris 7 users, but around 1999-09-01 it started to recommend it only for people who use Sun's compilers.

Here are some workarounds to this problem:

- Do not install Sun patch 107058-01 until after Sun releases a complete patch for bug 4210064. This is the simplest course to take, unless you must also use Sun's C compiler. Unfortunately 107058-01 is preinstalled on some new Solaris 7-based hosts, so you may have to back it out.
- Copy the original, unpatched Solaris 7 `/usr/ccs/bin/as` into `/usr/local/libexec/gcc/sparc-sun-solaris2.7/3.4/as`, adjusting the latter name to fit your local conventions and software version numbers.
- Install Sun patch 106950-03 (1999-05-25) or later. Nobody with both 107058-01 and 106950-03 installed has reported the bug with GCC and Sun's dynamic linker. This last course of action is riskiest, for two reasons. First, you must install 106950 on all hosts that run code generated by GCC; it doesn't suffice to install it only on the hosts that run GCC itself. Second, Sun says that 106950-03 is only a partial fix for bug 4210064, but Sun doesn't know whether the partial fix is adequate for GCC. Revision -08 or later should fix the bug. The current (as of 2004-05-23) revision is -24, and is included in the Solaris 7 Recommended Patch Cluster.

GCC 3.3 triggers a bug in version 5.0 Alpha 03/27/98 of the Sun assembler, which causes a bootstrap failure when linking the 64-bit shared version of libgcc. A typical error message is:

```
ld: fatal: relocation error: R_SPARC_32: file libgcc/sparcv9/_muldi3.o:
  symbol <unknown>:  offset 0xffffffff7ec133e7 is non-aligned.
```

This bug has been fixed in the final 5.0 version of the assembler.

A similar problem was reported for version Sun WorkShop 6 99/08/18 of the Sun assembler, which causes a bootstrap failure with GCC 4.0.0:

```
ld: fatal: relocation error: R_SPARC_DISP32:
  file .libs/libstdc++.lax/libsupc++convenience.a/vterminate.o:
    symbol <unknown>: offset 0xfccd33ad is non-aligned
```

This bug has been fixed in more recent revisions of the assembler.

sparc-*-linux*

GCC versions 3.0 and higher require binutils 2.11.2 and glibc 2.2.4 or newer on this platform. All earlier binutils and glibc releases mishandled unaligned relocations on `sparc-*-*` targets.

sparc64-*-solaris2*

When configuring the GNU Multiple Precision Library (GMP) or the MPFR library, the canonical target triplet must be specified as the `build` parameter on the configure line. For example on a Solaris 7 system:

```
% ./configure --build=sparc64-sun-solaris2.7 --prefix=xxx
```

The following compiler flags must be specified in the configure step in order to bootstrap this target with the Sun compiler:

```
% CC="cc -xarch=v9 -xildoff" srcdir/configure [options] [target]
```

'`-xarch=v9`' specifies the SPARC-V9 architecture to the Sun toolchain and '`-xildoff`' turns off the incremental linker.

sparcv9-*-solaris2*

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

--sysv*

On System V release 3, you may get this error message while linking:

```
ld fatal: failed to write symbol name something
 in strings table for file whatever
```

This probably indicates that the disk is full or your ulimit won't allow the file to be as large as it needs to be.

This problem can also result because the kernel parameter `MAXUMEM` is too small. If so, you must regenerate the kernel and make the value much larger. The default value is reported to be 1024; a value of 32768 is said to work. Smaller values may also work.

On System V, if you get an error like this,

```
/usr/local/lib/bison.simple: In function 'yyparse':
/usr/local/lib/bison.simple:625: virtual memory exhausted
```

that too indicates a problem with disk space, ulimit, or `MAXUMEM`.

On a System V release 4 system, make sure '`/usr/bin`' precedes '`/usr/ucb`' in `PATH`. The `cc` command in '`/usr/ucb`' uses libraries which have bugs.

vax-dec-ultrix

Don't try compiling with VAX C (`vcc`). It produces incorrect code in some cases (for example, when `alloca` is used).

--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).

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 (32-bit)

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.

OS/2

GCC does not currently support OS/2. However, Andrew Zabolotny has been working on a generic OS/2 port with pgcc. The current code can be found at http://www.goof.com/pcg/os2/.

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 63, 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, m6811, m6812, 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. If you want to install your own homemade configuration files, you can use '`local`' as the company

name to access them. If you use configuration '*cpu*`-local`', the configuration name without the cpu prefix is used to form the configuration file names.

Thus, if you specify '`m68k-local`', configuration uses files '`m68k.md`', '`local.h`', '`m68k.c`', '`xm-local.h`', '`t-local`', and '`x-local`', all in the directory '`config/m68k`'.

GNU Free Documentation License

Version 1.2, November 2002

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.

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 for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

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.

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 1.2
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.