Installing GCC

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 8 [Specific], page 27. 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 Downloading GCC

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

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

The full distribution includes the C, C++, Objective-C, Fortran, Java, and Ada (in 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, 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.

3 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 CVS, *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.

Note that the bootstrap compiler and the resulting GCC must be link compatible, else the bootstrap will fail with linker errors about incompatible object file formats. Several multilibed targets are affected by this requirement, see Chapter 8 [Specific], page 27.

To configure GCC:

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

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 i960-rtems, 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.

`--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.

These additional options control where certain parts of the distribution are installed. 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 parts of GCC. The default is '*`exec-prefix`*`/lib`'.

`--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/g++-v3`'.

`--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, except for '`libobjc`' which is built as a static library only by default.

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`' and '`libjava`'. Note that '`libobjc`' does not recognize itself by any name, so, if you list package names in '`--enable-shared`', you will only get static Objective-C libraries. '`libf2c`' and '`libiberty`' do 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`*'.

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`'
- '*`any`*`-lynx-lynxos`'
- '`mips-`*`any`*'
- '`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, and for '`mips-sgi-irix5.*`'), 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:

- Check the '*`exec_prefix`*`/lib/gcc-lib/`*`target`*`/`*`version`*' directory, where *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.

- Check operating system specific directories (e.g. '`/usr/ccs/bin`' on Sun Solaris 2).

Note that these rules do not check for the value of `PATH`. 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.

  `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 thread support.

  `pthreads`
  : Same as '`posix`' on arm*-*-linux*, *-*-chorusos* and *-*-freebsd* only. A future release of gcc might remove this alias or extend it to all platforms.

  `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.

`--with-cpu=`*cpu*
: Specify which cpu variant the compiler should generate code for by default. This is currently only supported on some ports, specifically arm, powerpc, and SPARC. If configure does not recognize the model name (e.g. arm700, 603e, or ultrasparc) you provide, please check the '`gcc/config.gcc`' script for a complete list of supported models.

`--enable-altivec`
: Specify that the target supports AltiVec vector enhancements. This option will adjust the ABI for AltiVec enhancements, as well as generate AltiVec code when appropriate. This option is only available for PowerPC systems.

`--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.

`--enable-version-specific-runtime-libs`
: Specify that runtime libraries should be installed in the compiler specific subdirectory ('*libsubdir*') rather than the usual places. In addition, '`libstdc++`''s include files will be installed in '*libsubdir*`/include/g++`' 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 '`libf2c`' and '`libstdc++`', and is the default for '`libobjc`' which cannot be changed in this case.

`--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: `ada`, `c`, `c++`, `f77`, `java`, `objc`. Building the Ada compiler has special requirements, see below.
If you do not pass this flag, all languages available in the '`gcc`' sub-tree will be configured. Re-defining `LANGUAGES` when calling '`make bootstrap`' **does not** work anymore, as those language sub-directories might not have been configured!

`--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.

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

`--enable-win32-registry`
`--enable-win32-registry=`*key*
`--disable-win32-registry`
: The '`--enable-win32-registry`' option enables 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-checking`
`--enable-checking=`*list*
: When you specify this option, the compiler is built to perform checking of tree node types when referencing fields of that node, and some other internal consistency checks. 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 on by default when building from CVS or snapshots, but off for releases. More control over the checks may be had by specifying *list*; the categories of checks available are '`misc`', '`tree`', '`gc`', '`rtl`', '`rtlflag`', '`gcac`' and '`valgrind`'.

The check '`valgrind`' requires the external `valgrind` simulator, available from `http://developer.kde.org/~sewardj/`. The default when *list* is not specified is '`misc,tree,gc,rtlflag`'; the checks '`rtl`', '`gcac`' and '`valgrind`' are very expensive.

`--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-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`'.

`--with-system-zlib`
Use installed zlib rather than that included with GCC. This option only applies if the Java front end is being built.

`--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.

Some options which only apply to building cross compilers:

`--with-headers`
`--with-headers=`*dir*
: 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.

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

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

4 Building

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

We **highly** recommend that GCC be built using GNU make; other versions may work, then again they might not. GNU make is required for compiling GNAT (the Ada compiler) and the Java runtime library.

(For example, many broken versions of make will fail if you use the recommended setup where *objdir* is different from *srcdir*. Other broken versions may recompile parts of the compiler when installing the compiler.)

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.

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 CVS or snapshots, or if you modify parser sources, you need the Bison parser generator installed. Any version 1.25 or later should work; older versions may also work. 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 CVS or snapshots, or if you modify Texinfo documentation, you need version 4.2 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.

4.1 Building a native compiler

For a native build issue the command '`make bootstrap`'. This will build the entire GCC system, which includes the following steps:

- Build host tools necessary to build the compiler such as texinfo, bison, gperf.
- 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.

- Perform a 3-stage bootstrap of the compiler.
- 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. This is identical to '`make bootstrap`' except that 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 bootstrap`'. 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.

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 bootstrap`' **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`'.)

4.2 Building a cross compiler

We recommend reading the crossgcc FAQ for information about building cross compilers.

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 such as texinfo, bison, gperf.

- 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.

4.3 Building in parallel

You can use '`make bootstrap MAKE="make -j 2" -j 2`', or just '`make -j 2 bootstrap`' for GNU Make 3.79 and above, instead of '`make bootstrap`' to build GCC in parallel. 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.

4.4 Building the Ada compiler

In order to build GNAT, the Ada compiler, you need a working GNAT compiler (GNAT version 3.13 or later, or GCC version 3.1 or later), since the Ada front end is written in Ada (with some GNAT-specific extensions), and GNU make.

However, you do not need a full installation of GNAT, just the GNAT binary '`gnat1`', a copy of '`gnatbind`', and a compiler driver which can deal with Ada input (by invoking the '`gnat1`' binary). You can specify this compiler driver by setting the `ADAC` environment variable at the configure step. `configure` can detect the driver automatically if it has got a common name such as `gcc` or `gnatgcc`. Of course, you still need a working C compiler (the compiler driver can be different or not). `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.

Additional build tools (such as `gnatmake`) or a working GNAT run-time library installation are usually *not* required. However, if you want to bootstrap the compiler using a minimal version of GNAT, you have to issue the following commands before invoking '`make bootstrap`' (this assumes that you start with an unmodified and consistent source distribution):

```
cd srcdir/gcc/ada
touch treeprs.ads [es]info.h nmake.ad[bs]
```

At the moment, the GNAT library and several tools for GNAT are not built by '`make bootstrap`'. You have to invoke '`make gnatlib_and_tools`' in the '*`objdir`*`/gcc`' subdirectory before proceeding with the next steps.

For example, you can build a native Ada compiler by issuing the following commands (assuming `make` is GNU make):

```
cd objdir
srcdir/configure --enable-languages=c,ada
```

```
cd srcdir/gcc/ada
touch treeprs.ads [es]info.h nmake.ad[bs]
cd objdir
make bootstrap
cd gcc
make gnatlib_and_tools
cd ..
```

Currently, when compiling the Ada front end, you cannot use the parallel build feature described in the previous section.

5 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 1.4.1 or 1.4.3 and later, 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.

5.5 How can I run the test suite 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.

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

5.6 Additional testing for Java Class Libraries

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 test suite 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`'.

5.7 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 our testing harness does not allow fine grained control over whether or not a test is expected to fail. We expect to fix this problem in future releases.

5.8 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.

6 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-lib`'; 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 built a released version of GCC using '`make bootstrap`' 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 us 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 8 [Specific], page 27 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` telling us how the information should be changed.

If you find a bug, please report it following our 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 4.2) 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`. You can also buy printed manuals from the Free Software Foundation, though such manuals may not be for the most recent version of GCC.

7 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.
- Sinix/Reliant Unix—Siemens.
- 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, 2.7, 8, and 9,

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.

8 Host/target specific installation notes for GCC

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

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`'.

The '`--enable-threads`' options isn't supported yet. A patch is in preparation for a future release.

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`'.

You absolutely **must** use GNU make on this platform. Also, 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 during '`make bootstrap`' 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-*-aout

This configuration is obsoleted in GCC 3.3.

Advanced RISC Machines ARM-family processors. These are often used in embedded applications. There are no standard Unix configurations. This configuration corresponds to the basic instruction sequences and will produce '`a.out`' format object modules.

You may need to make a variant of the file '`arm.h`' for your particular configuration.

arm-*-elf

This configuration is intended for embedded systems.

arm*-*-linux-gnu

We require GNU binutils 2.10 or newer.

avr

ATMEL AVR-family micro controllers. These are used in embedded applications. There are no standard Unix configurations. See section "AVR Options" in *Using and Porting 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.openavr.org
- 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.

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 and Porting 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 and Porting 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/`.

DOS

Please have a look at our 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.

dsp16xx

A port to the AT&T DSP1610 family of processors.

--freebsd*

The version of binutils installed in '`/usr/bin`' is known to work unless otherwise specified in any per-architecture notes. However, binutils 2.12.1 or greater is known to improve overall testsuite results.

Support for FreeBSD 1 was discontinued in GCC 3.2.

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 4.8-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-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 our 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 versions 7, 8, and 9 is obsoleted in GCC 3.3.

We *highly* recommend using gas/binutils 2.8 or newer on all hppa platforms; you may encounter a variety of problems when using 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 and configure GCC with the '`--with-gnu-as`' and '`--with-as=...`' options.

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

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.

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

hppa*-hp-hpux9

Support for this system is obsoleted in GCC 3.3.

The HP assembler has major problems on this platform. We've tried to work around the worst of the problems. However, those workarounds may be causing linker crashes in some circumstances; the workarounds also probably prevent shared libraries from working. Use the GNU assembler to avoid these problems.

The configuration scripts for GCC will also trigger a bug in the hpux9 shell. To avoid this problem set `CONFIG_SHELL` to '`/bin/ksh`' and `SHELL` to '`/bin/ksh`' in your environment.

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 is much better than the hpux9 assembler, but still 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 '`make bootstrap`'. You should be able to continue by saying '`make all`' after getting the failure from '`make bootstrap`'.

hppa*-hp-hpux11

GCC 3.0 and up support HP-UX 11. On 64-bit capable systems, there are two distinct ports. The '`hppa2.0w-hp-hpux11*`' port generates code for the 32-bit pa-risc runtime architecture. It uses the HP linker. The '`hppa64-hp-hpux11*`' port generates 64-bit code for the pa-risc 2.0 architecture. The script config.guess now selects the port type based on the type compiler detected during configuration. You must set your `PATH` or define `CC` so that configure finds an appropriate compiler for the initial bootstrap. Different prefixes must be used if both ports are to be installed on the same system.

It is best to explicitly configure the '`hppa64-hp-hpux11*`' target with the '`--with-ld=...`' option. We support both the HP and GNU linkers for this target. The two linkers require different link commands. Thus, it's not possible to switch linkers during a GCC build. This has been been reported to occur in a unified build of binutils and GCC.

GCC 2.95.x is not supported under HP-UX 11 and cannot be used to compile GCC 3.0 and up. Refer to binaries for information about obtaining precompiled GCC binaries for HP-UX.

You must use GNU binutils 2.11 or above with the 32-bit port. Thread support is not currently implemented, so '`--enable-threads`' does not work. See:

- `http://gcc.gnu.org/ml/gcc-prs/2002-01/msg00551.html`
- `http://gcc.gnu.org/ml/gcc-bugs/2002-01/msg00663.html`

GCC 3.3 and later support weak symbols on the 32-bit port using SOM secondary definition symbols. This feature is not enabled for earlier versions of HP-UX since there have been bugs in the linker support for secondary symbols. The HP linker patches `PHSS_26559` and `PHSS_24304` for HP-UX 11.00 and 11.11, respectively, correct the problem of linker core dumps creating C++ libraries. Earlier patches may work but they have not been tested.

GCC 3.3 nows uses the ELF DT_INIT_ARRAY and DT_FINI_ARRAY capability to run initializers and finalizers on the 64-bit port. The feature requires CVS binutils as of January 2, 2003, or a subsequent release to correct a problem arising from HP's non-standard use of the .init and .fini sections. The 32-bit port uses the linker '`+init`' and '`+fini`' options. As with the support for secondary symbols, there have been bugs in the order in which these options are executed by the HP linker. So, again a recent linker patch is recommended.

The HP assembler has many limitations and is not recommended for either the 32 or 64-bit ports. For example, it does not support weak symbols or alias definitions. As a result, explicit template instantiations are required when using C++. This will make it difficult if not impossible to build many C++ applications. You also can't generate debugging information when using the HP assembler with GCC.

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.

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.

When starting with a HP compiler, it is preferable to use the ANSI compiler as the bundled compiler only supports traditional C. Bootstrapping with the bundled compiler is tested infrequently and problems often arise because of the subtle differences in semantics between traditional and ISO C.

This port still is undergoing significant development.

i370-*-*

This port is very preliminary and has many known bugs. We hope to have a higher-quality port for this machine soon.

--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.

If you use glibc 2.2 (or 2.1.9x), GCC 2.95.2 won't install out-of-the-box. You'll get compile errors while building '`libstdc++`'. The patch glibc-2.2.patch, that is to be applied in the GCC source tree, fixes the compatibility problems.

Currently Glibc 2.2.3 (and older releases) and GCC 3.0 are out of sync since the latest exception handling changes for GCC. Compiling glibc with GCC 3.0 will give a binary incompatible glibc and therefore cause lots of problems and might make your system completely unusable. This will definitely need fixes in glibc but might also need fixes in GCC. We strongly advise to wait for glibc 2.2.4 and to read the release notes of glibc 2.2.4 whether patches for GCC 3.0 are needed. You can use glibc 2.2.3 with GCC 3.0, just do not try to recompile it.

i?86-*-linux*aout

Use this configuration to generate '`a.out`' binaries on Linux-based GNU systems. This configuration is being superseded. You must use gas/binutils version 2.5.2 or later.

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-*-sco

Compilation with RCC is recommended. Also, it may be a good idea to link with GNU malloc instead of the malloc that comes with the system.

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 the latest version of the Supplement Graphics, Web and X11 Libraries (GWXLIBS) package. 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 and GWXLIBS). 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. GWXLIBS 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 GWXLIBS is significantly updated in Maintenance Pack 1. Please visit ftp://ftp.sco.com/pub/openserver5 and ftp://ftp.sco.com/pub/openserver5/opensrc 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 2.14 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 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-*-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 bootstrap`' 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.

The toolchain is not completely finished, so requirements will continue to change. GCC 3.0.1 and later require glibc 2.2.4. GCC 3.0.2 requires binutils from 2001-09-05 or later. GCC 3.0.1 requires binutils 2.11.1 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: 3.0.2, 3.0.1, 3.0, Red Hat 2.96, and Trillian 000717. This primarily affects C++ programs and programs that create shared libraries. Because of these ABI incompatibilities, GCC 3.0.2 is not recommended for user programs on GNU/Linux systems built using earlier compiler releases. GCC 3.0.2 is recommended for compiling linux, the kernel. GCC 3.0.2 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.

*-lynx-lynxos

Support for SPARC LynxOS is obsoleted in GCC 3.3.

LynxOS 2.2 and earlier comes with GCC 1.x already installed as '`/bin/gcc`'. You should compile with this instead of '`/bin/cc`'. You can tell GCC to use the GNU assembler and linker, by specifying '`--with-gnu-as --with-gnu-ld`' when configuring. These will produce COFF format object files and executables; otherwise GCC will use the installed tools, which produce '`a.out`' format executables.

-ibm-aix

Support for AIX versions 1, 2, and 3 is obsoleted in GCC 3.3.

AIX Make frequently has problems with GCC makefiles. GNU Make 3.76 or newer is recommended to build on this platform.

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).

'`libstdc++`' in GCC 3.2 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 GCC 3.1 version of the shared library. Applications either need to be re-linked against the new shared library or the GCC 3.1 version of the '`libstdc++`' shared object needs to be available to the AIX runtime loader. The GCC 3.1 '`libstdc++.so.4`' shared object 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 object from each the GCC 3.1 '`libstdc++.a`' archive:

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

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
```

Archive the runtime-only shared object in the GCC 3.2 '`libstdc++.a`' archive:

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

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`*'.

ip2k-*-elf

Ubicom IP2022 micro controller. This configuration is intended for embedded systems. There are no standard Unix configurations.

Use '`configure --target=ip2k-elf --enable-languages=c`' to configure GCC.

m32r-*-elf

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

m68000-hp-bsd

Support for this system is obsoleted in GCC 3.3.

HP 9000 series 200 running BSD. Note that the C compiler that comes with this system cannot compile GCC; contact `law@cygnus.com` to get binaries of GCC for bootstrapping.

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-att-sysv

Support for this system is obsoleted in GCC 3.3.

AT&T 3b1, a.k.a. 7300 PC. This version of GCC cannot be compiled with the system C compiler, which is too buggy. You will need to get a previous version of GCC and use it to bootstrap. Binaries are available from the OSU-CIS archive, at `ftp://ftp.uu.net/systems/att7300/`.

m68k-crds-unos

Support for this system is obsoleted in GCC 3.3.

Use '`configure unos`' for building on Unos.

The Unos assembler is named `casm` instead of `as`. For some strange reason linking '`/bin/as`' to '`/bin/casm`' changes the behavior, and does not work. So, when installing GCC, you should install the following script as '`as`' in the subdirectory where the passes of GCC are installed:

```
#!/bin/sh
casm $*
```

The default Unos library is named '`libunos.a`' instead of '`libc.a`'. To allow GCC to function, either change all references to '`-lc`' in '`gcc.c`' to '`-lunos`' or link '`/lib/libc.a`' to '`/lib/libunos.a`'.

When compiling GCC with the standard compiler, to overcome bugs in the support of `alloca`, do not use '`-O`' when making stage 2. Then use the stage 2 compiler with '`-O`' to make the stage 3 compiler. This compiler will have the same characteristics as the usual stage 2 compiler on other systems. Use it to make a stage 4 compiler and compare that with stage 3 to verify proper compilation.

(Perhaps simply defining `ALLOCA` in '`x-crds`' as described in the comments there will make the above paragraph superfluous. Please inform us of whether this works.)

Unos uses memory segmentation instead of demand paging, so you will need a lot of memory. 5 Mb is barely enough if no other tasks are running. If linking '`cc1`' fails, try putting the object files into a library and linking from that library.

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, if you wish to use gas, you must use gas version 2.1 or later, and you must use the GNU linker version 2.1 or later. Earlier versions of gas relied upon a program which converted the gas output into the native HP-UX format, but that program has not been kept up to date. 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-ncr-*

Support for this system is obsoleted in GCC 3.3.

On the Tower models 4*n*0 and 6*n*0, by default a process is not allowed to have more than one megabyte of memory. GCC cannot compile itself (or many other programs) with '`-O`' in that much memory.

To solve this problem, reconfigure the kernel adding the following line to the configuration file:

```
MAXUMEM = 4096
```

m68k-sun

Support for this system is obsoleted in GCC 3.3.

Sun 3. We do not provide a configuration file to use the Sun FPA by default, because programs that establish signal handlers for floating point traps inherently cannot work with the FPA.

m68k-sun-sunos4.1.1

Support for this system is obsoleted in GCC 3.3.

It is reported that you may need the GNU assembler on this platform.

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-sgi-irix5

This configuration has considerable problems, which will be fixed in a future release.

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 Silicon Graphics. It is also available for download from http://www.sgi.com/developers/devtools/apis/ido.html.

'`make compare`' may fail on version 5 of IRIX 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 do you '`-save-temps`', you will have to manually delete the '`.i`' and '`.s`' files after each series of compilations.

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 `as` 2.11.2 or later, and use the '`--with-gnu-as`' configure option when configuring GCC. GNU `as` is distributed as part of the binutils package. When using release 2.11.2, you need to apply a patch http://sources.redhat.com/ml/binutils/2001-07/msg00352.html which will be included in the next release of binutils.

When building GCC, the build process loops rebuilding `cc1` over and over again. This happens on '`mips-sgi-irix5.2`', and possibly other platforms. It has been reported that this is a known bug in the `make` shipped with IRIX 5.2. We recommend you use GNU `make` instead of the vendor supplied `make` program; however, you may have success with `smake` on IRIX 5.2 if you do not have GNU `make` available.

mips-sgi-irix6

If you are using IRIX `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.

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

You must *not* use GNU `as` (which isn't built anyway as of binutils 2.11.2) on IRIX 6 platforms; doing so will only cause problems.

GCC does not currently support generating O32 ABI binaries in the '`mips-sgi-irix6`' configurations. It is possible to create a GCC with O32 ABI only support by configuring it for the '`mips-sgi-irix5`' target and using a patched GNU `as` 2.11.2 as documented in the '`mips-sgi-irix5`' section above. Using the native assembler requires patches to GCC which will be included in a future release. It is expected that O32 ABI support will be available again in a future release.

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.

GCC does not correctly pass/return structures which are smaller than 16 bytes and which are not 8 bytes. The problem is very involved and difficult to fix. It affects a number of other targets also, but IRIX 6 is affected the most, because it is a 64-bit target, and 4 byte structures are common. The exact problem is that structures are being padded at the wrong end, e.g. a 4 byte structure is loaded into the lower 4 bytes of the register when it should be loaded into the upper 4 bytes of the register.

GCC is consistent with itself, but not consistent with the SGI C compiler (and the SGI supplied runtime libraries), so the only failures that can happen are when there are library functions that take/return such structures. There are very few such library functions. Currently this is known to affect `inet_ntoa`, `inet_lnaof`, `inet_netof`, `inet_makeaddr`, and `semctl`. Until the bug is fixed, GCC contains workarounds for the known affected functions.

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/tools/compilers.html` (free registration required).

The default stack limit of 512K is too small, which may cause compiles to fail with 'Bus error'. Set the stack larger, for instance by doing '`limit stack 800`'. It's a good idea to use the GNU preprocessor instead of Apple's '`cpp-precomp`' during the first stage of bootstrapping; this is automatic when doing '`make bootstrap`', but to do it from the toplevel objdir you will need to say '`make CC='cc -no-cpp-precomp' bootstrap`'.

The version of GCC shipped by Apple typically includes a number of extensions not available in a standard GCC release. These extensions are generally specific to Mac programming.

powerpc-*-elf, powerpc-*-sysv4

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

powerpc-*-linux-gnu*

You will need binutils 2.13.90.0.10 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 4.2 (NetBSD 1.5.1 included Texinfo version 3.12).

powerpc-*-eabiaix

Embedded PowerPC system in big endian mode with '`-mcall-aix`' selected as the default.

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 Linux for S/390.

s390x-*-linux*

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

--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 our binaries page for details.

The Solaris 2 `/bin/sh` will often fail to configure '`libstdc++-v3`', '`boehm-gc`' or '`libjava`'. We therefore recommend to use the following sequence of commands to bootstrap and install GCC:

```
% CONFIG_SHELL=/bin/ksh
% export CONFIG_SHELL
% srcdir/configure [options] [target]
% gmake bootstrap
% gmake install
```

As explained in the build instructions, we recommend to use GNU make, which we call `gmake` here to distinguish it from Sun make.

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 the vendor tools (Sun `as`, Sun `ld`).

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.

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.

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/lib/gcc-lib/sparc-sun-solaris2.7/3.1/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 2001-09-24) revision is -14, 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.

sparc-sun-sunos4*

Support for this system is obsoleted in GCC 3.3.

A bug in the SunOS 4 linker will cause it to crash when linking '`-fPIC`' compiled objects (and will therefore not allow you to build shared libraries).

To fix this problem you can either use the most recent version of binutils or get the latest SunOS 4 linker patch (patch ID 100170-10) from Sun's patch site.

Sometimes on a Sun 4 you may observe a crash in the program `genflags` or `genoutput` while building GCC. This is said to be due to a bug in `sh`. You can probably get around it by running `genflags` or `genoutput` manually and then retrying the `make`.

sparc-unknown-linux-gnulibc1

Support for this system is obsoleted in GCC 3.3.

It has been reported that you might need binutils 2.8.1.0.23 for this platform, too.

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*

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

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

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

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).

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 ‘`gcc/config/xtensa/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)

A port of GCC 2.95.2 and 3.x is included with the Cygwin environment.

Current (as of early 2001) snapshots of GCC will build under Cygwin without modification.

GCC does not currently 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/.

An older copy of GCC 2.8.1 is included with the EMX tools available at ftp://ftp.leo.org/pub/comp/os/os2/leo/devtools/emx+gcc/.

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 CVS 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 sources.redhat.com 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.

9 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 GNU CC on a GNU or Unix system. See Section 9.11 [VMS Install], page 56, for VMS systems.

1. If you have chosen a configuration for GNU CC 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. (If you are building a cross-compiler, see Section 9.10 [Cross-Compiler], page 53.)

   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 9.9 [Configurations], page 51, 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 GNU CC.

9.9 Configurations Supported by GNU CC

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 GNU CC 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. GNU CC 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`'.

Here is a list of configurations that have special treatment or special things you must know:

'`vax-dec-vms`'
: See Section 9.11 [VMS Install], page 56, for details on how to install GNU CC on VMS.

9.10 Building and Installing a Cross-Compiler

GNU CC can function as a cross-compiler for many machines, but not all.

- 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.
- Cross-compilers between machines with different floating point formats have not all been made to work. GNU CC now has a floating point emulator with which these can work, but each target machine description needs to be updated to take advantage of it.
- Cross-compilation between machines of different word sizes is somewhat problematic and sometimes does not work.

Since GNU CC generates assembler code, you probably need a cross-assembler that GNU CC can run, in order to produce object files. If you want to link on other than the target machine, you need a cross-linker as well. You also need header files and libraries suitable for the target machine that you can install on the host machine.

9.10.1 Steps of Cross-Compilation

To compile and run a program using a cross-compiler involves several steps:

- Run the cross-compiler on the host machine to produce assembler files for the target machine. This requires header files for the target machine.
- Assemble the files produced by the cross-compiler. You can do this either with an assembler on the target machine, or with a cross-assembler on the host machine.
- Link those files to make an executable. You can do this either with a linker on the target machine, or with a cross-linker on the host machine. Whichever machine you use, you need libraries and certain startup files (typically '`crt....o`') for the target machine.

It is most convenient to do all of these steps on the same host machine, since then you can do it all with a single invocation of GNU CC. This requires a suitable cross-assembler and cross-linker. For some targets, the GNU assembler and linker are available.

9.10.2 Configuring a Cross-Compiler

To build GNU CC as a cross-compiler, you start out by running '`configure`'. Use the '`--target=`*`target`*' to specify the target type. If '`configure`' was unable to correctly identify the system you are running on, also specify the '`--build=`*`build`*' option. For example, here is how to configure for a cross-compiler that produces code for an HP 68030 system running BSD on a system that '`configure`' can correctly identify:

```
./configure --target=m68k-hp-bsd4.3
```

9.10.3 Tools and Libraries for a Cross-Compiler

If you have a cross-assembler and cross-linker available, you should install them now. Put them in the directory '`/usr/local/`*`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 GNU CC 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 and GAS. Configure them with the same '`--host`' and '`--target`' options that you use for configuring GNU CC, then build and install them. They install their executables automatically into the proper directory. Alas, they do not support all the targets that GNU CC supports.

If you want to install libraries to use with the cross-compiler, such as a standard C library, put them in the directory '`/usr/local/`*`target`*`/lib`'; installation of GNU CC copies all the files in that subdirectory into the proper place for GNU CC to find them and link with them. Here's an example of copying some libraries from a target machine:

```
ftp target-machine
lcd /usr/local/target/lib
cd /lib
get libc.a
cd /usr/lib
get libg.a
get libm.a
quit
```

The precise set of libraries you'll need, and their locations on the target machine, vary depending on its operating system.

Many targets require "start files" such as '`crt0.o`' and '`crtn.o`' which are linked into each executable; these too should be placed in '`/usr/local/`*`target`*`/lib`'. 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. Here's an example of copying these files from a target machine:

```
ftp target-machine
lcd /usr/local/target/lib
prompt
cd /lib
mget *crt*.o
cd /usr/lib
mget *crt*.o
quit
```

9.10.4 Cross-Compilers and Header Files

If you are cross-compiling a standalone program or a program for an embedded system, then you may not need any header files except the few that are part of GNU CC (and those of your program). However, if you intend to link your program with a standard C library such as '`libc.a`', then you probably need to compile with the header files that go with the library you use.

The GNU C compiler does not come with these files, because (1) they are system-specific, and (2) they belong in a C library, not in a compiler.

If the GNU C library supports your target machine, then you can get the header files from there (assuming you actually use the GNU library when you link your program).

If your target machine comes with a C compiler, it probably comes with suitable header files also. If you make these files accessible from the host machine, the cross-compiler can use them also.

Otherwise, you're on your own in finding header files to use when cross-compiling.

When you have found suitable header files, you should put them in the directory '`/usr/local/target/include`', before building the cross compiler. Then installation will run fixincludes properly and install the corrected versions of the header files where the compiler will use them.

Provide the header files before you build the cross-compiler, because the build stage actually runs the cross-compiler to produce parts of '`libgcc.a`'. (These are the parts that *can* be compiled with GNU CC.) Some of them need suitable header files.

Here's an example showing how to copy the header files from a target machine. On the target machine, do this:

```
(cd /usr/include; tar cf - .) > tarfile
```

Then, on the host machine, do this:

```
ftp target-machine
lcd /usr/local/target/include
get tarfile
quit
tar xf tarfile
```

9.10.5 Actually Building the Cross-Compiler

Now you can proceed just as for compiling a single-machine compiler through the step of building stage 1.

Do not try to build stage 2 for a cross-compiler. It doesn't work to rebuild GNU CC as a cross-compiler using the cross-compiler, because that would produce a program that runs on the target machine, not on the host. For example, if you compile a 386-to-68030 cross-compiler with itself, the result will not be right either for the 386 (because it was compiled into 68030 code) or for the 68030 (because it was configured for a 386 as the host). If you want to compile GNU CC into 68030 code, whether you compile it on a 68030 or with a cross-compiler on a 386, you must specify a 68030 as the host when you configure it.

To install the cross-compiler, use '`make install`', as usual.

9.11 Installing GNU CC on VMS

The VMS version of GNU CC is distributed in a backup saveset containing both source code and precompiled binaries.

To install the '`gcc`' command so you can use the compiler easily, in the same manner as you use the VMS C compiler, you must install the VMS CLD file for GNU CC as follows:

1. Define the VMS logical names '`GNU_CC`' and '`GNU_CC_INCLUDE`' to point to the directories where the GNU CC executables ('`gcc-cpp.exe`', '`gcc-cc1.exe`', etc.) and the C include files are kept respectively. This should be done with the commands:

   ```
   $ assign /system /translation=concealed -
     disk:[gcc.] gnu_cc
   $ assign /system /translation=concealed -
     disk:[gcc.include.] gnu_cc_include
   ```

   with the appropriate disk and directory names. These commands can be placed in your system startup file so they will be executed whenever the machine is rebooted. You may, if you choose, do this via the '`GCC_INSTALL.COM`' script in the '`[GCC]`' directory.

2. Install the '`GCC`' command with the command line:

   ```
   $ set command /table=sys$common:[syslib]dcltables -
     /output=sys$common:[syslib]dcltables gnu_cc:[000000]gcc
   $ install replace sys$common:[syslib]dcltables
   ```

3. To install the help file, do the following:

   ```
   $ library/help sys$library:helplib.hlb gcc.hlp
   ```

   Now you can invoke the compiler with a command like '`gcc /verbose file.c`', which is equivalent to the command '`gcc -v -c file.c`' in Unix.

If you wish to use GNU C++ you must first install GNU CC, and then perform the following steps:

1. Define the VMS logical name '`GNU_GXX_INCLUDE`' to point to the directory where the preprocessor will search for the C++ header files. This can be done with the command:

   ```
   $ assign /system /translation=concealed -
     disk:[gcc.gxx_include.] gnu_gxx_include
   ```

   with the appropriate disk and directory name. If you are going to be using a C++ runtime library, this is where its install procedure will install its header files.

2. Obtain the file '`gcc-cc1plus.exe`', and place this in the same directory that '`gcc-cc1.exe`' is kept.

   The GNU C++ compiler can be invoked with a command like '`gcc /plus /verbose file.cc`', which is equivalent to the command '`g++ -v -c file.cc`' in Unix.

We try to put corresponding binaries and sources on the VMS distribution tape. But sometimes the binaries will be from an older version than the sources, because we don't always have time to update them. (Use the '`/version`' option to determine the version number of the binaries and compare it with the source file '`version.c`' to tell whether this is so.) In this case, you should use the binaries you get to recompile the sources. If you must recompile, here is how:

1. Execute the command procedure '`vmsconfig.com`' to set up the files '`tm.h`', '`config.h`', '`aux-output.c`', and '`md.`', and to create files '`tconfig.h`' and '`hconfig.h`'. This procedure also creates several linker option files used by '`make-cc1.com`' and a data file used by '`make-l2.com`'.

   ```
   $ @vmsconfig.com
   ```

2. Setup the logical names and command tables as defined above. In addition, define the VMS logical name '`GNU_BISON`' to point at the to the directories where the Bison executable is kept. This should be done with the command:

   ```
   $ assign /system /translation=concealed -
     disk:[bison.] gnu_bison
   ```

   You may, if you choose, use the '`INSTALL_BISON.COM`' script in the '`[BISON]`' directory.

3. Install the '`BISON`' command with the command line:

   ```
   $ set command /table=sys$common:[syslib]dcltables -
     /output=sys$common:[syslib]dcltables -
     gnu_bison:[000000]bison
   $ install replace sys$common:[syslib]dcltables
   ```

4. Type '`@make-gcc`' to recompile everything, or submit the file '`make-gcc.com`' to a batch queue. If you wish to build the GNU C++ compiler as well as the GNU CC compiler, you must first edit '`make-gcc.com`' and follow the instructions that appear in the comments.

5. In order to use GCC, you need a library of functions which GCC compiled code will call to perform certain tasks, and these functions are defined in the file '`libgcc2.c`'. To compile this you should use the command procedure '`make-l2.com`', which will generate the library '`libgcc2.olb`'. '`libgcc2.olb`' should be built using the compiler built from the same distribution that '`libgcc2.c`' came from, and '`make-gcc.com`' will automatically do all of this for you.

   To install the library, use the following commands:

   ```
   $ library gnu_cc:[000000]gcclib/delete=(new,eprintf)
   $ library gnu_cc:[000000]gcclib/delete=L_*
   $ library libgcc2/extract=*/output=libgcc2.obj
   $ library gnu_cc:[000000]gcclib libgcc2.obj
   ```

   The first command simply removes old modules that will be replaced with modules from '`libgcc2`' under different module names. The modules `new` and `eprintf` may not actually be present in your '`gcclib.olb`'—if the VMS librarian complains about those modules not being present, simply ignore the message and continue on with the next command. The second command removes the modules that came from the previous version of the library '`libgcc2.c`'.

   Whenever you update the compiler on your system, you should also update the library with the above procedure.

6. You may wish to build GCC in such a way that no files are written to the directory where the source files reside. An example would be the when the source files are on

a read-only disk. In these cases, execute the following DCL commands (substituting your actual path names):

```
$ assign dua0:[gcc.build_dir.]/translation=concealed, -
         dua1:[gcc.source_dir.]/translation=concealed  gcc_build
$ set default gcc_build:[000000]
```

where the directory '`dua1:[gcc.source_dir]`' contains the source code, and the directory '`dua0:[gcc.build_dir]`' is meant to contain all of the generated object files and executables. Once you have done this, you can proceed building GCC as described above. (Keep in mind that '`gcc_build`' is a rooted logical name, and thus the device names in each element of the search list must be an actual physical device name rather than another rooted logical name).

7. **If you are building GNU CC with a previous version of GNU CC, you also should check to see that you have the newest version of the assembler**. In particular, GNU CC version 2 treats global constant variables slightly differently from GNU CC version 1, and GAS version 1.38.1 does not have the patches required to work with GCC version 2. If you use GAS 1.38.1, then `extern const` variables will not have the read-only bit set, and the linker will generate warning messages about mismatched psect attributes for these variables. These warning messages are merely a nuisance, and can safely be ignored.
8. If you want to build GNU CC with the VAX C compiler, you will need to make minor changes in '`make-cccp.com`' and '`make-cc1.com`' to choose alternate definitions of `CC`, `CFLAGS`, and `LIBS`. See comments in those files. However, you must also have a working version of the GNU assembler (GNU as, aka GAS) as it is used as the back end for GNU CC to produce binary object modules and is not included in the GNU CC sources. GAS is also needed to compile '`libgcc2`' in order to build '`gcclib`' (see above); '`make-l2.com`' expects to be able to find it operational in '`gnu_cc:[000000]gnu-as.exe`'.

   To use GNU CC on VMS, you need the VMS driver programs '`gcc.exe`', '`gcc.com`', and '`gcc.cld`'. They are distributed with the VMS binaries ('`gcc-vms`') rather than the GNU CC sources. GAS is also included in '`gcc-vms`', as is Bison.

   Once you have successfully built GNU CC with VAX C, you should use the resulting compiler to rebuild itself. Before doing this, be sure to restore the `CC`, `CFLAGS`, and `LIBS` definitions in '`make-cccp.com`' and '`make-cc1.com`'. The second generation compiler will be able to take advantage of many optimizations that must be suppressed when building with other compilers.

Under previous versions of GNU CC, the generated code would occasionally give strange results when linked with the sharable '`VAXCRTL`' library. Now this should work.

Even with this version, however, GNU CC itself should not be linked with the sharable '`VAXCRTL`'. The version of `qsort` in '`VAXCRTL`' has a bug (known to be present in VMS versions V4.6 through V5.5) which causes the compiler to fail.

The executables are generated by '`make-cc1.com`' and '`make-cccp.com`' use the object library version of '`VAXCRTL`' in order to make use of the `qsort` routine in '`gcclib.olb`'. If you wish to link the compiler executables with the shareable image version of '`VAXCRTL`', you should edit the file '`tm.h`' (created by '`vmsconfig.com`') to define the macro `QSORT_WORKAROUND`.

`QSORT_WORKAROUND` is always defined when GNU CC is compiled with VAX C, to avoid a problem in case '`gcclib.olb`' is not yet available.

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 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 Warrany 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.