GNU Source Release Collection

for version 2014.10.11, 6 January 2014

bug-gsrc@gnu.org

This manual is for the GNU Source Release Collection (version 2014.10.11, updated 6 January 2014).

Copyright © 2011, 2012, 2013, 2014 Free Software Foundation, Inc.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 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."

Table of Contents

1 Introduction

The GNU Source Release Collection (GSRC) provides a simple way to install the latest GNU packages on an existing distribution. By using GSRC, the GNU source packages from `ftp.gnu.org` are automatically downloaded, compiled and installed, either in your home directory or a system-wide directory such as `/opt`.

At its core, it is a presentation of the current state of the GNU system, in the most appropriate form: buildable and installable source code. GSRC makes it easy to discover great new software from the GNU system, as well as providing other benefits over standard software distributions. It allows you, for example, to install easily GNU software for yourself on a system on which you do not have permission to install software system-wide; or to install the latest, unpatched packages when those distributed with your operating system are outdated or not configured to your liking.

GSRC is based on the GAR build system by Nick Moffitt and the GARstow enhancements by Adam Sampson. GAR was inspired by BSD Ports, a Makefile-based build system, and is written in GNU Make. The GARNOME build system for GNOME was another example of a system using GAR.

Note that GSRC is not intended to be a full package-management system or source distribution. It is just a more convenient way to compile GNU packages from source on an existing system.

Because GSRC is not a full distribution you will sometimes need to install other packages from your distribution to build and run GNU programs. For example, GSRC itself does not include Perl or Python, so you will need to make sure these are already installed for GNU programs which use them.

1.1 Building GNU packages

If you have never built a GNU package by hand, this section will briefly show the process so you will have an idea of what GSRC is doing. If you are already familiar with this, you may skip this section.

The build process is performed via commands entered into a shell, which is generally done in a terminal or a terminal emulator. The dollar sign in the following examples represents the *shell prompt*, denoting the point at which you enter commands, while the characters following the prompt show the commands that you must enter. While much of the build process is conveniently automated, such that you do not need to manually compile every file, you still must take a few steps.

For example, to build the package “hello” version 2.9, you must perform the following steps in your terminal:

1. Download the package and unpack it

```
$ wget http://ftpmirror.gnu.org/gnu/hello/hello-2.9.tar.gz
$ tar xvfz hello-2.9.tar.gz
```

2. Run the `configure` script

```
$ cd hello-2.9
$ ./configure
```

3. Compile the source code

```
$ make
```

4. Install it

```
$ make install
```

In some unfortunate cases, the process is not as straight-forward and may require some extra intervention on your part. GSRC abstracts away most of these steps so that all you need to enter to install a program is `make install`.

2 Getting started

GSRC is distributed directly using the Bazaar version control system or via a compressed archive. You can check out the latest version from the Bazaar repository using

```
$ bzr checkout bzr://bzr.savannah.gnu.org/gsrc/trunk/ gsrc
```

This will create a directory `gsrc`. The build definitions for GNU packages are in the `gnu` subdirectory. Large sub-projects, such as GNOME, have their own subdirectory containing packages (i.e. `gnome`). The `external` subdirectory contains references to dependencies which you may have to install outside of GSRC, such as via your GNU/Linux distribution's package manager (APT, Pacman, Yum, etc.). If these dependencies are required for a given package and are not found on your system, you will be automatically notified. Finally, the `decommissioned` directory contains former GNU packages that have been decommissioned.

Each package has its own subdirectory within its parent directory, for example `gnu/emacs` or `gnome/evince`. Package directories contain a `config.mk` file for configuring the package and a `Makefile` for building it. This `Makefile` will automate the usual `./configure` and `make` commands needed to build a GNU package.

To stay up-to-date with the latest releases of GNU software, you can pull in recent changes to your local copy of GSRC:

```
$ bzr update
```

Alternatively, quarter-annual "snapshots" of GSRC are made available for download at http://ftpmirror.gnu.org/gnu/gsrc.

2.1 Initial setup

If you have checked out the source tree from the Bazaar repository you will need to create the build files with the following command,

```
$ ./bootstrap
```

Before building any packages you will need to run the top-level configure script. There is only one configuration parameter, the installation prefix, specified with `--prefix`. For example, to install all the compiled packages under `/gnu` use:

```
$ ./configure --prefix=/gnu
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for a thread-safe mkdir -p... /usr/bin/mkdir -p
checking for gawk... gawk
checking whether make sets $(MAKE)... yes
checking whether make supports nested variables... yes
checking for recsel... /usr/bin/recsel
checking for recfmt... /usr/bin/recfmt
checking that generated files are newer than configure... done
configure: creating ./config.status
config.status: creating gsrc
config.status: creating config.mk
config.status: creating setup.sh
config.status: creating GNUmakefile
```

```
config.status: creating doc/Makefile
```

You can optionally install the documentation and the `gsrc` script (see Section 2.7 [Finding packages], page 7). Note that these are installed to the directory specified in the previous step. Be sure to set your environment to be able to use them (see Section 2.4 [Setting your environment], page 4).

```
$ make install
```

2.2 Building a simple package

All interaction with GSRC is performed via the program Make. When you execute Make via the `make` command, you generally must provide a *target* that tells Make which *recipe*, consisting of a series of pre-defined commands, to execute. For example, the `build` target will tell Make to execute a recipe to build the software, while the `install` target will execute a recipe for installing it. Often, a default recipe will be available that will typically build the software, allowing you to omit the `build` target.

Thus, in GSRC, to build any package, type `make build` (or, simply `make`) in the package's subdirectory. You can change to the directory with the `cd` command in the shell, or with the `-C` option of `make`. For example, to build the *hello* package in the `gnu/hello` subdirectory from the root GSRC directory use:

```
$ make -C gnu/hello
```

This will download, unpack, configure and build the *hello* package. The package will be built in the subdirectory `gnu/hello/work`.

```
$ ./gnu/hello/work/hello-2.9/src/hello
Hello, world!
```

2.3 Installing a package

You are now ready to install the package. If you are installing to a new directory tree, first create the directory specified in the top-level configure `--prefix` option if necessary,

```
$ mkdir /gnu
```

Then to install the package use the `install` target,

```
$ make -C gnu/hello install
```

The package should be automatically installed under `/gnu`, with any executable programs under `/gnu/bin/`.

```
$ /gnu/bin/hello --version
hello (GNU hello) 2.9
```

2.4 Setting your environment

If you want to use the newly installed package without having to specify its full path, you will need to modify the relevant variables in your environment, such as `PATH`, `LD_LIBRARY_PATH`, `INFOPATH`, etc. These variables inform your system of the locations of relevant files on it. For example, `PATH` contains a list of all directories that contain executable files.

There is a sample script `setup.sh` in the top-level GSRC directory which can be used to set the main environment variables.

```
$ source setup.sh
```

Note that you need to load this file into the current shell with the `source` command, instead of executing it (which would only apply the definitions temporarily in a subshell).

After loading this file, your environment variables should include the target directory so you can run the new packages directly:

```
$ echo $PATH
/gnu/bin:/usr/local/bin:/usr/bin:/bin
$ which hello
/gnu/bin/hello
```

If you want to restore your original environment variables they are saved in the variables `ORIG_PATH`, `ORIG_LD_LIBRARY_PATH`, etc.

```
$ PATH=$ORIG_PATH
$ LD_LIBRARY_PATH=$ORIG_LD_LIBRARY_PATH
```

2.5 Useful targets

To clean up the build directory and delete any downloaded files, use the `clean` target:

```
$ make -C gnu/hello clean
```

There are other useful targets. For example, the whole build sequence can be broken down into stages as follows:

```
$ make -C gnu/hello fetch checksum extract configure build install
```

Each target depends on the previous one, so typing `make -C gnu/hello install` executes all the earlier targets first.

You can install the source code of a package (to, i.e., `/gnu/src/hello-2.9`) using the `install-src` target. Likewise, the source can be removed using the `uninstall-src` target.

To see some information about a package, use the target `pkg-info`.

```
$ make -C gnu/hello pkg-info
make: Entering directory ‘/home/gnu/gsrc/gnu/hello’
Name:        GNU Hello
Version:     2.9
URL:         http://www.gnu.org/software/hello/manual/
Description:
 GNU Hello prints the message "Hello, world!" and then exits.  It
 serves as an example of standard GNU coding practices.  As such, it
 supports command-line arguments, multiple languages, and so on.
Status:      installed (stowed)
make: Leaving directory ‘/home/gnu/gsrc/gnu/hello’
```

The “Status” can be any of: “not installed”, “installed (not stowed)” or “installed (stowed)” (see Section 3.4 [Package versions], page 11).

To view a more concise summary, ideal for producing a list of packages in script, use the target `pkg-info-curt`.

```
$ make -C gnu/hello pkg-info-curt
make: Entering directory ‘/home/gnu/gsrc/gnu/hello’
gnu/hello 2.9
```

```
 A program that produces a familiar, friendly greeting
make: Leaving directory ‘/home/gnu/gsrc/gnu/hello’
```

To get a better idea of what files will be downloaded and which dependencies must be built in order to use a package, use the `fetch-list` target.

```
$ make -C gnu/hello fetch-list
make: Entering directory ‘/home/gnu/gsrc/gnu/hello’
Name: hello
Version: 2.9
Location: http://ftpmirror.gnu.org/hello/
Distribution files:
hello-2.9.tar.gz
Patch files:
Signature files:
hello-2.9.tar.gz.sig
Dependencies:
make: Leaving directory ‘/home/gnu/gsrc/gnu/hello’
```

Most GNU packages are highly configurable. To see which configuration options are available to you, you may invoke the `help-config` target.

Finally, if you choose to remove a package, you may use the `uninstall` target. This target "un-stows" the package; if you were to re-install it, the package would not need to be re-built. Instead, it would merely be re-stowed. To completely remove a package, use the `uninstall-pkg` target. When you update a package to a new version, the old version is merely un-stowed and the new version is installed alongside it (see Section 3.4 [Package versions], page 11). In order to clean out old package versions, use the `uninstall-pkg-old` target.

2.6 Complex packages

If building or using a package depends on other GNU packages, these will be built automatically in the correct order. To see the dependencies of any package use the `dep-list` target.

```
$ make -C gnu/gnupg dep-list
make: Entering directory ‘/home/gnu/gsrc/gnu/gnupg’
libgpg-error libgcrypt libassuan libksba pth zlib readline
make: Leaving directory ‘/home/gnu/gsrc/gnu/gnupg’
```

The dependencies are searched for in the `gnu`, `gnustep` and `gnome` subdirectories by default. Of course, packages might depend on software that does not belong to the GNU project. In those cases, GSRC will try to determine whether these external packages are installed on your system. If one is not present, you will have to install it separately, for example via your distribution's software repositories.

Note that the dependencies can be more than one level deep,

```
$ make -C gnu/readline dep-list
make: Entering directory ‘/home/gnu/gsrc/gnu/readline’
ncurses
make: Leaving directory ‘/home/gnu/gsrc/gnu/readline’
```

So, to install a complex package like *gnupg* use the same commands as for a simple package,

```
$ make -C gnu/gnupg
$ make -C gnu/gnupg install
```

All of the dependencies (and the dependencies' dependencies) will be built and installed first, as needed.

2.7 Finding packages

GSRC provides build recipes for several hundred packages. So, how can you find or discover a package relevant to your needs? Fortunately, the build recipes are described by metadata, which can help you in searching. For example, you can use standard GNU tools such as `grep` to search the text of the build recipes for key words.

A template script is installed, called `gsrc`, that provides a simple means for searching for packages via keywords, printing information about a package, and printing its location. Since `gsrc` is installed to the same location as executables installed by GSRC, if you have set up your environment to use GSRC packages (see Section 2.4 [Setting your environment], page 4), you can use the `gsrc` script to access GSRC from outside the GSRC directory.

For example, here we search for an editor, discover the program *moe*, read information about it, and then install it.

```
$ gsrc search editor
gnu/denemo 1.0.0
 A music notation editor
gnu/ed 1.7
 An implementation of the standard Unix editor
gnu/emacs 24.3
 The extensible, customizable text editor
gnu/global 6.2.8
 A source code tag system
gnu/gnusound 0.7.5
 A multitrack sound editor
gnu/leg
 Libraries for game engines and game development
gnu/less 451
 A pager
gnu/mc 4.6.1
 A two-paned file manager
gnu/mit-scheme 9.1.1
 An implementation of the Scheme programming language
gnu/moe 1.5
 A simple-to-use text editor
gnu/nano 2.3.2
 A simple text editor
gnu/sed 4.2.2
 A text stream editor
$ gsrc info moe
```

```
Name:        Moe
Version:     1.5
URL:         http://www.gnu.org/software/moe
Description:
 GNU Moe is a powerful-but-simple-to-use text editor.  It works in a
 modeless manner, and features an intuitive set of key-bindings that
 assign a degree of "severity" to each key; for example key
 combinations with the Alt key are for harmless commands like cursor
 movements while combinations with the Control key are for commands
 that will modify the text.  Moe features multiple windows, unlimited
 undo/redo, unlimited line length, global search and replace, and
 more.
Status:      not installed
$ make -C $(gsrc path moe) install
```

If you view the `gsrc` script's code, you will find that it is very simple and, indeed, can be used as a template to be expanded to include the functionality that you desire.

More robust searching can be performed with the file `MANIFEST.rec`. If you have acquired GSRC by downloading it as a `tar.gz` archive, this file should be present in the package's root directory. If you have acquired GSRC by cloning its code repository, you will have to generate this file. Simply navigate to the package's root directory and enter `make manifest`; you will want to run this every time you pull updates to the repository. The resulting file is a *recfile*, which can be queried as a database using GNU Recutils, which must be installed (see Section "recsel" in *Recutils*).

3 Advanced configuration

The default behavior of GSRC may be configured both globally and for individual packages. All configuration is done in simple Makefiles, so some familiarity with GNU Make, while not required, is recommended for more advanced changes.

3.1 Global configuration

Building a package loads the following configuration files:

`config.mk`
: Specifies the installation directory prefix. Created by the configure script from `config.mk.in`

`gar.conf.mk`
: Specifies general configuration variables

`gar.env.mk`
: Defines the environment variables that are set during each build step.

`gar.master.mk`
: Defines the list of mirror sites used to download the source tarballs. It is recommended to modify this to use local mirrors.

`gar.site.mk`
: An optional file that you can create to load extra recipes to use on packages. This file must be created by the user (however, it is not an eroror if the file does not exist).

Much of the behavior of GSRC is defined by variables that can be customized. Generally speaking, you should override these variables in your `config.mk` file rather than in the `gar.*.mk` files. That way, you do not have to worry about updates to GSRC overwriting your changes.

Some of the more important configuration variables are:

`BOOTSTRAP`
: If defined (the default), the environment variables `C_INCLUDE_PATH`, `CPLUS_INCLUDE_PATH` and `LDFLAGS` point to the `include` and `lib` subdirectories of the installation directory. This forces the use of any previously installed libraries in preference to the normal system libraries. To disable this feature, remove the definition `BOOTSTRAP=1` in `config.mk.in` and rerun configure, or build with `BOOTSTRAP` undefined on the command-line:

```
$ make -C gnu/gnupg BOOTSTRAP=
```

Set in `conf.mk`

`IGNORE_DEPS`
: Specifies any packages that should be skipped as dependencies (for example, if you prefer to use existing system packages instead). A space separated list. Set in `gar.conf.mk`.

`GARCHIVEDIR`
`GARBALLDIR`
: Specifies the directories used to cache downloaded source code archives (`GARCHIVEDIR`) and the archives of the installed packages (`GARBALLDIR`). Set in `gar.conf.mk`.

`MAKE_ARGS_PARALLEL`
: Set this to `-j N` to allow N parallel processes in the build. Note that multiple dependencies are built one-by-one; only the commands within each build are performed in parallel. Set in `gar.conf.mk`

`USE_COLOR`
: It's easy to miss the messages printed by GSRC amongst all the output of the build process. Set this to "y" to enable colorized output of GSRC messages, which may make them more visible. Set it to anything else to disable color. In either case, four more variables are defined: `MSG`, `MSG2`, `ERR`, `OK` and `OFF`. The first four define strings to insert at the beginning of a normal message (`MSG`, `MSG2`), an error message (`ERR`), or a message indicating success (`OK`). The `OFF` code is inserted at the end of the message. When `USE_COLOR` is "y", these variables contain ANSI escape sequences to change properties of the text (i.e. to set colors or text weight). Otherwise, they may contain textual indicators, such as "==> " to begin a message. Some sensible default values for both cases are included. Set in `gar.conf.mk`.

`REDIRECT_OUTPUT`
: A typical build process produces a lot of textual output. In some cases, you may wish to redirect this output to somewhere other than your screen. In this case, you may set the variable `REDIRECT_OUTPUT` to any value other than "n". To edit where the output will be redirected, set the `OUTPUT` variable. By default, if you set `REDIRECT_OUTPUT`, standard text output will be redirected to `/dev/null`, which means it is thrown away, while errors will be printed to the screen. You can instead, for example, redirect to log files of your choosing (see Section "Redirections" in *Bash* for more details on redirection). Set in `gar.conf.mk`

3.2 Package configuration

Each package can be customized to your liking. Because GNU packages follow a standardized build process, customizing the GSRC build for one is straightforward.

GNU packages take most of their configuration in the form of options passed to the `configure` script. One may easily customize these options in a GSRC Makefile by setting the `CONFIGURE_OPTS` variable. Any options added to this variable will be appended to the options set by default by GSRC.

```
CONFIGURE_OPTS = --disable-gtk --without-png
```

For convenience, every package has a file called `config.mk` in its directory which is imported by its build script. Typically, all user configuration should be done here. By default, it contains the `CONFIGURE_OPTS` and `BUILD_OPTS` variables. In some special cases, package-specific, user-customize-able variables are also defined in this file.

Generally speaking, user configuration is done exclusively in `config.mk` while `Makefile` contains the information and recipes necessary for the package to build correctly. Thus, you should not need to modify the `Makefile` unless you have special requirements. Note that most configuration options relating to directory locations (such as where to install, where to search for libraries, etc.) are set in the `Makefile`, because they are necessary for proper building and installation in GSRC. Therefore, you do not need to worry about setting them correctly in `config.mk`.

3.3 Patching packages

If you have a patch that you would like to apply to a package, the process can be automated by GSRC. First, in the package's directory, make a subdirectory called `files` and move the patch file(s) there. Next, create two variables in the package's `Makefile`:

```
PATCHFILES = my-patch.diff my-patch2.diff
PATCHOPTS = -p0
```

`PATCHFILES` holds a list of all the patch files in the `files` subdirectory. `PATCHOPTS` contains the option switches to pass to the `patch` program.

Next, the patch file's checksum is added to the checksums file for the package.

```
$ make makesum
```

Note that if the `make makesums` command fails due to GPG verification and you trust the source from which the package or patch was downloaded, you may instead use `make makesums GPGV=true` to skip this key verification step.

Finally, you may build the package as normal. The patch(es) will be applied automatically in the process.

```
$ make install
```

If the patching process fails and you are sure that the patch is for the version of the package contained in GSRC, then you may have to modify the `-p` option in the `PATCHOPTS` variable (see Section "patch Options" in *patch*).

If the package requires a patch to even build properly, then this is a bug in GSRC. Please report such build problems to `bug-gsrc@gnu.org`. You should also contact the maintainers of the software package to make them aware of the problem.

3.4 Package versions

What is actually happening "under the hood" when GSRC installs a package is slightly more complicated than what has been described so far.

When you install a package, it is first actually installed to the `/gnu/packages` directory in a sub-directory with the name <package>-<version> (i.e. `/gnu/packages/hello-2.8`). In the example of the package *hello*, the executable `hello` is installed to `/gnu/packages/hello-2.8/bin/hello` instead of `/gnu/bin/hello`. All other files installed by the package are installed in a similar manner. Next, GSRC makes symbolic links to those files inside the parent `/gnu` directory. Thus, `/gnu/bin/hello` is ultimately a symlink to `/gnu/packages/hello-2.8/bin/hello`. This is referred to as *stowing*; a package with symlinks to its files installed in the system is said to be *stowed*.

When a new version of a package is released, you do not have to uninstall the previous version first. When *hello 2.9* is built and installed, it is put into its own package directory,

`/gnu/packages/hello-2.9` and the directory of *hello 2.8* is left untouched. When GSRC finalizes the installation, the old symlinks are removed and new ones are created to the latest version's files. Thus, while there would then actually be two versions of the package installed, only the latest one would be stowed.

If you want to stow a particular version of the package, you may pass the `GARVERSION` variable to `make install`. Be sure to update the checksums when you do so, otherwise the process will fail!

```
$ make -C gnu/hello makesum install GARVERSION=2.8
```

If you had previously built version 2.8, then GSRC will merely re-stow those files. Of course, if you have not previously built it, or if you have previously run `make clean`, the package will be built from scratch.

Note: this method may fail if the package naming format or compression algorithm has changed between versions (i.e. a change from tar.gz to tar.xz); in this case you must also modify `DISTFILES`.

Users wishing to maintain different configurations of a package may take advantage of the `GARPROFILE` variable. Its value is merely appended to the package directory name, allowing you to have multiple configurations of the same package version installed. For example:

```
$ make -C gnu/hello install CONFIGURE_OPTS="--disable-nls" GARPROFILE="-no-nls"
```

This would install the newly configured package to `/gnu/packages/hello-2.9-no-nls`.

Appendix A Technical information

This appendix gives detailed information on the GSRC build system. This information is not necessary for most users but it may be of interest to developers and GSRC maintainers.

A.1 The GSRC build system

The GSRC build system is based on a system called GARstow by Adam Sampson, which, in turn, was based on an earlier system called GAR by Nick Moffitt. In this section, the basic architecture of the GSRC build system will be described.

GSRC consists of several system Makefiles plus the Makefile for each package. When the user calls `make` on a package's Makefile, the GSRC system Makefiles are pulled in. There are several of these system Makefiles:

File	Description
`gar.mk`	This file contains the top-level targets such as `build` or `install`.
`gar.lib.mk`	This file contains recipes to perform the sub-tasks for each top-level target (see below).
`gar.master.mk`	This file contains master URLs for downloading packages (i.e. `http://ftp.gnu.org/gnu`).
`gar.lib`	This directory contains further Makefiles to define common variable values for typical build systems, such as the standard GNU Autotools process.
`gar.conf.mk`	This file contains the general configuration of GSRC.
`gar.env.mk`	The variables in this file are used to properly set the build environment for GSRC.
`config.mk`	This file contains the user's particular GSRC configuration.

The typical user-level GSRC Make targets, such as `fetch`, `build` or `install`, come from `gar.mk`. Depending on the package's build requirements, as defined in the package's GSRC Makefile, these user-level targets will depend on lower-level targets that actually perform the required tasks.

For example, in a typical GNU package, configuration is done with a `configure` script while building and installing are done with a `Makefile`. So, for the package *hello*, the `build` target will depend on a target called `build-work/hello-2.9/Makefile` (`build-` plus the location of the `Makefile` distributed with the package). For a Python-based package that is installed via a `setup.py`, the `install` target will depend on the target `install-work/foo-1.0/setup.py`. The file `gar.lib.mk` contains many generalized Make recipes to handle each of these different scenarios.

The directory `gar.lib` contains Makefiles that set common variable values for packages that share similar build systems. It has a file called `auto.mk`, for example, that defines the settings for a package that uses the standard Autotools process.

A.2 Anatomy of a GSRC Makefile

GSRC Makefiles are the point of entry for the user into the GSRC system. Since GSRC supplies GNU software and there are GNU coding standards that dictate how package installation is supposed to work, the GSRC Makefiles for most GNU software packages are similar.

In order to facilitate working with the GSRC Makefiles in an automated way, such as searching them via a script, they all share a common structure, split into three sections: metadata variables, build variables, and the build recipes. By convention, these three sections are separated by lines of seventy hash symbols ("#"). This helps to visually separate the sections, as well as to provide convenient stopping points when scanning or searching the files.

A.2.1 Metadata variables

This section consists of variable declarations that describe the package itself. The following variables should be present:

Variable name	Description
`NAME`	This is the common-language, official name of the package. It may contain multiple words and any character. Example: "GNU Source-highlight"
`GARNAME`	This is the internal GSRC name of the package. It should match the name of the directory containing the package and, by convention, for GNU packages it is the name of the package's HTTP subdirectory on `http://www.gnu.org/software`. It should consist of only lower case letters, numbers, hyphens or underscores. Example: "src-highlite"
`UPSTREAMNAME`	[optional] If the package maintainers ever use a different name for the package, for example a different spelling or capitalization, include it here. This is often useful in specifying URLs or package arcive names. Example: "source-highlight"
`GARVERSION`	This is the current version number of the package. Example: "3.1.7"
`DISTNAME`	[optional] This variable contains the distribution name of the package. This variable is automatically constructed and by default it is `$(GARNAME)-$(GARVERSION)`. Example: "src-highlite-3.1.7"

`HOME_URL`	This is the home URL of the package, where a user might find more information about it. Example: "http://www.gnu.org/software/src-highlite"
`DESCRIPTION`	This variable should have a short, one-line description of the package.
`BLURB`	[optional] This should contain a longer, multi-line description of the package. To achieve this, its value needs to be declared using the Make `define` statement.

A.2.2 Build variables

The second section of a GSRC Makefile holds variable definitions that are used in the build process. When possible, it is preferable to use the metadata variables in the build variable definitions, to minimize the number of items that need to be modified should anything change.

Variable name	**Description**
`MASTER_SITES`	This variable defines the top-level URL from where the package files should be retrieved. Many URLs are already defined in variables in the file `gar.master.mk`. Most GNU packages are retrievable from `http://ftp.gnu.org/gnu`, which is assigned to the variable `MASTER_GNU` in `gar.master.mk`, so for a GNU package, `MASTER_SITES` would be set to `$(MASTER_GNU)`. Multiple sites may be listed; attempts to download a files will proceed for each site listed until one succeeds.
`MASTER_SUBDIR`	This is the directory of the master site under which the package files can be found. For most GNU packages, this can simply be `$(GARNAME)/`.
`DISTFILE_SITES`	This variable contains URL(s) from which source distribution archives only are to be downloaded.
`DISTFILE_SUBDIR`	This variable contains the sub-directory of `DISTFILE_SITES` where the source distributions can be found.
`SIGFILE_SITES`	This variable contains URL(s) from which signature files only are to be downloaded.
`SIGFILE_SUBDIR`	This variable contains the sub-directory of `SIGFILE_SITES` where the signature files can be found.

`PATCHFILE_SITES` This variable contains URL(s) from which patch files only are to be downloaded.

`PATCHFILE_SUBDIR` This variable contains the sub-directory of `DISTFILE_SITES` where the source distributions can be found.

`FILE_SITES` This variable lists file URIs where files can be found locally. By default this contains the `files` sub-directory of the package's GSRC directory and the location specified by the variable `GARCHIVEDIR`. Note that these URIs should be prefaced with "file://".

`DISTFILES` This variable contains a space-separated list of all of the source distribution archives to be fetched.

`SIGFILES` This variable contains a space-separated list of all the signature files to fetch.

`PATCHFILES` This variable contains a space-separated list of all the patch files to fetch.

`WORKSRC` This variable contains the name of the directory where all of the work should take place. Its default value is `$(WORKDIR)/$(DISTNAME)`, which should be sufficient for most cases, so it is normally not necessary to set this variable. If, however, the package's source archive extracts to a directory with some other name, you should set it here. This should always begin with `$(WORKDIR)`, which by default is the `work` subdirectory of the GSRC package's sub-directory.

`WORKOBJ` This variable defines the location where the build process should take place. Normally, and by default, this is the same as `WORKSRC`, however some packages recommend building in a directory separate from the location of the source code.

`CONFIGURE_SCRIPTS` This variable contains a list of the scripts or files that need to be run during the configuration step of the build process. Phony targets may also be included.

`BUILD_SCRIPTS` This variable contains a list of the scripts or files that need to be run during the build step of the build process. Phony targets may also be included.

`INSTALL_SCRIPTS` This variable contains a list of the scripts or files that need to be run during the install step of the build process. Phony targets may also be included.

`INFO_FILES`	This variable contains a list of all of the Info documentation files installed by a program. To use this variable, you must include the `info.mk` file from the `gar.lib` directory. If this variable is not defined and `info.mk` is included, then it will have a default value of `$(GARNAME).info`
`BUILDDEPS`	This variable contains a space-separated list of the programs required to build the package, using their GARNAMEs.
`LIBDEPS`	This variable is slightly a misnomer. It is a space-separated list of all the programs and/or libraries required at run-time by the package.

A.2.3 Build recipes

The final section of the GSRC Makefile contains the specifics of building the package. For most cases, it is sufficient to just add `include ../../gar.lib/auto.mk`, which will work for any package that follows the GNU building and installation standards. This will, among other actions, automatically define the `CONFIGURE_SCRIPTS`, `BUILD_SCRIPTS` and `INSTALL_SCRIPTS` variables and it will include the `gar.mk` Makefile. If the package does not follow this building standard, then add `include ../../gar.mk` directly. Following this, the user's package configuration should be loaded with `include config.mk`.

Because there is the possibility that the user specify some configuration options, any further options that must be set within the Makefile should be done after the user configuration has been loaded. By convention, whereas the user specifies options with the `CONFIGURE_OPTS` and `BUILD_OPTS` variables, inside the GSRC Makefile options should be included by *appending* to the `CONFIGURE_ARGS` and `BUILD_ARGS` variables:

```
CONFIGURE_ARGS += --some-option
```

Finally, if necessary, the actual recipes are written. Note that if `gar.lib/auto.mk` was included, no recipes should need to be written. In general, there are two kinds of targets for which recipes may need to be written.

The first correspond to the files listed under `CONFIGURE_SCRIPTS`, `BUILD_SCRIPTS` and `INSTALL_SCRIPTS`. As mentioned previously, user-level targets, such as `build`, depend on lower-level targets such as `build-work/hello-2.9/Makefile`. These are the targets that must be implemented for each of the designated configure/build/install scripts. For each target, a recipe is written using the normal Make syntax to perform the necessary task. Recall that phony targets may be specified as configure/build/install scripts. So, if `INSTALL_SCRIPTS = java`, then a target named `install-java` must be written.

The second kind of targets that may be written are pre- and post- rules. These recipes are run before or after the specified top-level target. For example, a target called `pre-build` is run immediately before the `build` target. These targets are convenient for performing pre- or post-processing on files. Note that there are also `pre-everything` and `post-everything` targets that can be written.

A.2.4 A simple example

```
NAME = GNU Hello
GARNAME = hello
```

```
GARVERSION = 2.9
HOME_URL = http://www.gnu.org/software/hello/manual/
DESCRIPTION = A program that produces a familiar, friendly greeting
define BLURB
GNU Hello prints the message "Hello, world!" and then exits.  It
serves as an example of standard GNU coding practices.  As such, it
supports command-line arguments, multiple languages, and so on.
endef

###############################################################################

MASTER_SITES = $(MASTER_GNU)
MASTER_SUBDIR = $(GARNAME)/
DISTFILES = $(DISTNAME).tar.gz
SIGFILES = $(DISTNAME).tar.gz.sig

BUILDDEPS =
LIBDEPS =

###############################################################################

include ../../gar.lib/auto.mk
include config.mk
```

A.2.5 A complex example

```
NAME = Linux Libre
GARNAME = linux-libre
GARVERSION = 3.8.5
HOME_URL = http://www.fsfla.org/svnwiki/selibre/linux-libre/
DESCRIPTION = A free version of the Linux kernel
define BLURB
Linux Libre is a free (as in freedom) variant of the Linux kernel.
It has been modified to remove any non-free binary blobs.
endef

###############################################################################

MASTER_SITES = http://linux-libre.fsfla.org/pub/
MASTER_SUBDIR = $(GARNAME)/releases/$(GARVERSION)-gnu/
DISTFILES = $(DISTNAME)-gnu.tar.xz
SIGFILES = $(DISTNAME)-gnu.tar.xz.sign

WORKSRC = $(WORKDIR)/linux-$(GARVERSION)
CONFIGURE_SCRIPTS = $(WORKSRC)/Makefile
BUILD_SCRIPTS = $(WORKSRC)/Makefile
INSTALL_SCRIPTS = kernel
```

```
BUILDDEPS =
LIBDEPS =

##############################################################################

include ../../gar.mk
include config.mk

CONFIGURE_ARGS = $(CONFIGURE_OPTS)
BUILD_ARGS += $(if $(USE_PARALLEL),$(MAKE_ARGS_PARALLEL),)

CREATED_MERGE_DIRS = \
sysconf $(sysconfdir) \
var $(vardir) \
rootlib /lib

pre-configure:
make -C $(WORKSRC) mrproper
$(MAKECOOKIE)

configure-%/Makefile:
$(CONFIGURE_ENV) make -C $* $(MAKE_ARGS) $(CONFIGURE_ARGS) $(CONFIGURE_TARGET)
$(MAKECOOKIE)

post-configure:
cd $(WORKSRC) && make $(MAKE_ARGS) prepare
$(MAKECOOKIE)

build-%/Makefile:
$(BUILD_ENV) make -C $* $(BUILD_ARGS)
$(MAKECOOKIE)

install-kernel:
make -C $(WORKOBJ) $(MAKE_ARGS) \
INSTALL_MOD_PATH=$(packageprefix) \
INSTALL_HDR_PATH=$(packageprefix) \
modules_install \
headers_install \
firmware_install
@install -m755 -D $(WORKSRC)/arch/$(ARCH)/boot/bzImage $(packageprefix)/boot/vmlinuz-$
@install -m755 $(WORKSRC)/System.map $(packageprefix)/boot/System.map-$(GARVERSION)
@install -m755 $(WORKSRC)/.config $(packageprefix)/boot/config-$(GARVERSION)
$(MAKECOOKIE)
```

Appendix B GNU Free Documentation License

Version 1.3, 3 November 2008

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

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

0. PREAMBLE

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

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

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

1. APPLICABILITY AND DEFINITIONS

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

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

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

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

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

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

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

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

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

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

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

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

2. VERBATIM COPYING

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

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

3. COPYING IN QUANTITY

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

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

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

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

4. MODIFICATIONS

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

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

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

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

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

D. Preserve all the copyright notices of the Document.

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

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

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

H. Include an unaltered copy of this License.

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

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

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

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

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

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

O. Preserve any Warranty Disclaimers.

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

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

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

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

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

5. COMBINING DOCUMENTS

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

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

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

6. COLLECTIONS OF DOCUMENTS

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

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

7. AGGREGATION WITH INDEPENDENT WORKS

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

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

8. TRANSLATION

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

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

9. TERMINATION

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

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

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

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

10. FUTURE REVISIONS OF THIS LICENSE

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

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

11. RELICENSING

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

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

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

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

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

ADDENDUM: How to use this License for your documents

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

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

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

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

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

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.