

GNU-Build-System HOWTO

Giridhar Appaji Nag Y

appaji{at}ibiblio{dot}org

Mark Hoebeke

mark{dot}hoebeke{at}orange{dot}fr

Copyright © 2003 Mark HoebekeY Giridhar Appaji Nag
None Yet!

Revision History

Revision 0.1 2003-11-05 Revised by: mh
first formatting of consensus TOC

This HOWTO aims to help beginning developers to leverage the power the the GNU Build System (GBS). The GBS is composed of well-known tools such as **make** and of less-used utilities such as **autoconf**, **automake** or **libtool**. The primary goal of the HOWTO is to ring developers up to speed with each of these tools, and, at another level to explain some of their inner workings.

Document last updated \$Date: 2007/03/05 07:11:36 \$.

1. Introduction

1.1. About this HOWTO

If you ever installed packages from source on a Unix environment, you will have encountered `Makefile` that is used to compile the package. When there was no **automake** or **autoconf**, it was a non trivial task to get big source packages to compile, leave alone getting them to work for you. It often involved tweaking of the `Makefile` and making changes to source. Finding the necessary things that you need

(like libraries, the right version of the library needed) before you could compile a particular package, was a difficult task.

Another major problem that many software developers face is that of maintaining portability, achieving platform independence and building libraries. This is important if the code written should run on a variety of platforms. Finding the specifics of a machine and the operating system, and configuring the source of a package for successful compilation and working, is also a problem of interest.

The GNU Autotools address these issues. This HOWTO is a resource for first time users of the *autotools*. It is intended to be an introductory guide, to help with GNU **make**, **autoconf**, **automake** **libtool** etc. It also tries to provide an explanation as to how these tools work. To be able to follow the explanations given in here, the reader is expected to be familiar with Unix like operating systems and the basics of programming.

1.2. Keeping up-to-date

New versions of this document that are released would be available from The LDP website and its mirrors. Since a few mirrors are broken or not up-to-date, it is highly recommended that the latest released version be accessed from <http://tldp.org/HOWTO/GNU-Build-System-HOWTO/> (<http://tldp.org/HOWTO/GNU-Build-System-HOWTO/>). The latest in-progress version of this document is available at the GNU-Build-System HOWTO home page (<http://www.appaji.net/stuff/gbsh.html>).

1.3. Document license

This document, the *GNU-Build-System HOWTO*, is distributed under the terms of the *GNU General Public License*. The word *Program* in the license is to be interpreted as *document*. The term *source code* refers to the DocBook/XML format of this document and the term *object code* refers to one, many or all the other formats that this document is available in, after conversion from the *source code*.

This program is free software; you can redistribute it and/or modify it under the terms of the *GNU General Public License* as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version. A copy of the license is available at <http://www.gnu.org/licenses/gpl.txt> (<http://www.gnu.org/licenses/gpl.txt>) and can be found in Appendix A.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

1.4. Disclaimer

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY;

without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

All copyrights are held by their by their respective owners, unless specifically noted otherwise. Use of a term in this document should not be regarded as affecting the validity of any trademark or service mark. Naming of particular products or brands should not be seen as endorsements.

2. Hello build system!

The GNU Build System encompasses a number of components that assist a developer in each stage of the configure, compile and distribute process. Projects using the suite of the GBS are recognizable by the ease with which they are installed. The ubiquitous `./configure && make && make install` does it all.

This section of the document introduces the build tools and explains how to use them. Though the treatment would not be in depth, it should help you understand the files that these tools operate on. The aim is to take the reader step by step, through the stages turning a set of source files of a “project” to conform to the `./configure && make && make install` mantra. We will work through each of these components using a `Hello World!` styled example.

2.1. Hello GNU make!

If you are reading this document, it is possible that you already have prior experience with the `make` utility. However, any discussion of the build system would be incomplete without a mention of `make`.

2.1.1. What is make?

`make` is a utility for automatic compilation of a project’s source code. It is used in projects to find out if the files (typically C sources) that a certain “target” (typically object files and executables) depends on have changed, and re-generates the target. It makes compiling only the necessary files of a big project as simple as typing `make`, when a few of them have been modified. The input to the `make` utility is a `Makefile`.

2.1.2. make in action

A very minimal `Makefile` to `make` a program called `foo` from its C source file, `foo.c` would look like:

```
[jrh ~/gbs]$ cat Makefile
foo: foo.c
    gcc -o foo foo.c
```

```
[jrh ~/gbs]$
```

The target in this Makefile is `foo` and it depends on the `foo.c` dependency. The second line (command) tells make how to build the target in case the dependency changes. The command should be on its own line with a *tab* character (and *not* spaces) before it.

The `foo` executable can then be created by issuing a **make foo** command in the directory where the above Makefile is present.

```
[jrh ~/gbs]$ make
gcc -o foo foo.c
[jrh ~/gbs]$
```

A slightly more useful example would be:

```
[jrh ~/gbs]$ cat Makefile

all: hw

hw: main.o hello.o world.o
    gcc main.o hello.o world.o -o hw

world.o: world.c hello.h world.h
    gcc -Wall -c hello.c

hello.o: hello.c hello.h
    gcc -Wall -c hello.c

world.o: world.c world.h
    gcc -Wall -c world.c

clean:
    rm -f hw hello.o world.o

[jrh ~/gbs]$
```

The default target in this Makefile (which is the first target that appears in the Makefile) is `all`. `all` depends on `hw` which in-turn depends on `main.o`, `hello.o` and `world.o`. These can be “made” using the rules that have been specified for them respectively. For example, `hello.o` is created using the command **gcc -Wall -c hello.c** whenever `hello.c` or `hello.h` are modified.

```
[jrh ~/gbs]$ make
gcc -Wall -c main.c
gcc -Wall -c hello.c
gcc -Wall -c world.c
gcc main.o hello.o world.o -o hw
[jrh ~/gbs]$
```

Another invocation of **make** (or **make all**) does nothing. This is because none of the files that **make** looks at have been modified.

```
[jrh ~/gbs]$ make all
make: Nothing to be done for `all'.
[jrh ~/gbs]$
```

However, if one of the files is changed (say `hello.h`), **make** will figure out the right files to be re-compiled (based on the dependencies in the `Makefile`) and generate a new `hw`.

```
[jrh ~/gbs]$ touch hello.h
[jrh ~/gbs]$ make all
gcc -Wall -c main.c
gcc -Wall -c hello.c
gcc main.o hello.o world.o -o hw
[jrh ~/gbs]$
```

The other target `clean` in the `Makefile` can be invoked explicitly to remove the target files that have been generated.

```
[jrh ~/gbs]$ make clean
rm -f hw main.o hello.o world.o
[jrh ~/gbs]$
```

2.1.3. Variables in Makefiles

A `Makefile` can also be written using variables and wild-cards making it easy to modify and maintain. For example, the above `Makefile` can be re-written as:

```
[jrh ~/gbs]$ cat Makefile

TARGET = hw
OBJS = main.o hello.o world.o
CC = gcc
CCOPTS = -Wall -c
RM = rm
RMOPTS = -f

all: $(TARGET)

$(TARGET): $(OBJS)
    $(CC) $(OBJS) -o $(TARGET)

main.o: main.c hello.h world.h
    $(CC) $(CCOPTS) $<

%.o: %.c %.h
    $(CC) $(CCOPTS) $<
```

```
clean:
    $(RM) $(RMOPTS) $(TARGET) $(OBJS)

[jrh ~/gbs]$
```

Every `$(VARIABLE)` in the `Makefile` is replaced with its value. The generic rule `%.o: %.c %.h`¹ says that each `.o` file is dependent on the corresponding `.c` and the `.h` files. `$$` refers to the current target and `$$<` refers to the first one among the list of dependencies of the current target. Note that this version of the `Makefile` does not need a lot of modifications if a single source file and the corresponding header files are added.

GNU **make** provides several other useful features like the `VPATH` directive to locate the source files that are not co-located with the `Makefile`, facility for sophisticated wildcards, etc. These are described in the *official GNU **make** manual*.

2.1.4. A GNU `Makefile`

A `Makefile` that goes with software packages that follow the GNU `Makefile` conventions has a few standard targets and provides useful and commonly used functionality like:

- `all`: to build the package
- `clean`: which removes all the generated files
- `tags`: generates the TAGS file for source browsing
- `check`: which executes any automated test cases that have been included with the package.
- `dist`: creates a `.tar.gz` distribution of the software package.
- `install`: installs the compiled binaries, manual pages, documents at their appropriate location.
- `uninstall`: removes the files that have been installed

and many more ...

Writing the dependencies and rules for these targets is a non-trivial task and involves some effort. **automake** solves this problem by generating all the standard targets.

2.2. Hello autoconf!

2.2.1. What is autoconf?

We need to rely on other tools to ensure that a project is built and runs correctly on most flavors of Unix (including Linux of course). These platform specific parameters are detected by the `configure` script

generated by **autoconf**. **autoconf** also generates a `Makefile`. The input to the `configure` script are the `Makefile.in` and the `configure.in` files.

2.3. Hello automake!

2.3.1. What is automake?

The final (?) step to build a well-behaved and distributable software project is to supply **make** with a set of standard targets (Section 2.1.4) and to use a set of thoroughly tested recipes to build them. This is where **automake** comes into play. Starting from a high-level description of the project's components (source files, binaries to be built, documentation, additional data files) found in a `Makefile.am`, **automake** is capable of generating a suitable `Makefile.in`. This can be used by the `configure` script to generate the `Makefile`.

2.4. Hello libtool!

So that's it? Now we've got our GBS compliant project architecture, isn't it? Indeed, but our toolkit would be incomplete if we omitted to mention **libtool**.

2.4.1. What is libtool?

The **libtool** component favors the use of program libraries (sets of related object files grouped into a single archive). These libraries can be used by all executables of a project, and in some cases, without being copied in every single executable. Ways of building libraries vary much from platform to platform, fortunately, **libtool** simplifies this process, lifting the burden from the programmer.

3. The real world

After the entry-level guided tour, a real world programming project will be subjected to the whole above slew of procedures here.

3.1. A real GNU make example

3.2. A real autoconf example

3.3. A real automake example

3.4. A real libtool example

3.5. Accompanying tools

A note on **makedepend**, **autoheader**, **autoscan**, **ifnames**, **aclocal**, **autoproject**, **autoupdate**, **autoreconf** etc.

4. An inside look

To the survivors of the former section, this is an inside look at the workings of all the GBS components. For starters, it describes the innards of the **make** before covering a rather ill-known macro programming language, **m4** upon which several of the GBS tools are built. It then scrutinizes **autoconf**, **automake** and **libtool** to answer *what* and *how*.

4.1. Inside GNU make

4.2. The m4 macro processor

4.3. Inside autoconf

4.4. Inside automake

4.5. Inside libtool

5. Installing the tools

This section of the HOWTO covers the installation of the GBS components, either from pre-packaged files specific to Linux distributions, or from source `.tar.gz` (or `.tar.bz2`) archives (also called *tarballs*).

5.1. From distribution packages

5.1.1. Fedora

5.1.2. SuSE

5.1.3. Debian

5.1.4. Mandrake

5.2. Installation from source

6. Final Words

6.1. Translations

6.2. Comments and Suggestions

Comments and suggestions regarding this document may be sent to [<mark{dot}hoebeke{at}orange{dot}fr>](mailto:mark{dot}hoebeke{at}orange{dot}fr) and [<appaaji@ibiblio.org>](mailto:appaaji@ibiblio.org). Please point out any mistakes, omissions, broken links and possible improvements so that they can be incorporated in future revisions of the document. In case you have additions and improvements that would change major portions of the document, it would be great if you could send them as a patch against the latest DocBook/XML version.

6.3. Acknowledgements

The authors of this document have the pleasure of acknowledging the following people who have contributed to this document by sending in comments and suggestions for improvements.

7. References and Resources

No example is good enough to expose the hurdles that inevitably show up when things get a bit more complex. The documents and references provided as references can be used to further experiment with the features that the tools described have to offer.

- Learning the GNU Development Tools (<http://www.st-andrews.ac.uk/~iam/docs/tutorial.html>) by Eleftherios Gkioulekas is a good tutorial to understand the GNU build system and the documentation files associated with GNU style packages.
- The “autotools” referred to in the above tutorial can be downloaded from <ftp://ftp.ugcs.caltech.edu/pub/elef/autotools> (<ftp://ftp.ugcs.caltech.edu/pub/elef/autotools>). Though the author seems to be planning a rewrite, it is worth taking a look at.
- autoproject: (<http://www.mv.com/ipusers/vanzandt/>) In the authors words, “autoproject - a script to start a programming project using **autoconf**, **automake**, and optionally a command line parser generator”
- GNU Autoconf, Automake and Libtool (<http://www.newriders.com/autoconf>) is probably the best guide for mastering these tools. You can also read it online at <http://sources.redhat.com/autobook/download.html> (<http://sources.redhat.com/autobook/download.html>)
- The official GNU **make** manual <http://www.gnu.org/software/make/manual/index.html> (<http://www.gnu.org/software/make/manual/index.html>). You are encouraged to explore and write more complex rules, commands in the rules, conditional parts, transformation of text in *Makefiles* etc., all of which are detailed in the manual.
- The official **autoconf** manual <http://www.gnu.org/software/autoconf/manual/autoconf-2.57/autoconf.html> (<http://www.gnu.org/software/autoconf/manual/autoconf-2.57/autoconf.html>). Take a look at the

documentation of existing tests and learn how to write new **autoconf** macros and tests. The indices of the manual are great for reference.

- Manuals of **automake** <http://www.gnu.org/software/automake/manual/index.html> (<http://www.gnu.org/software/automake/manual/index.html>). A few examples, alternative approaches to handling sub-directories, how various targets work, support for test suites etc.
- The official **libtool** manual <http://www.gnu.org/software/libtool/manual.html> (<http://www.gnu.org/software/libtool/manual.html>)
- Yet another language to pick up. The **m4** manual with lots of examples is at <http://www.gnu.org/software/m4/manual/index.html> (<http://www.gnu.org/software/m4/manual/index.html>)

A. GNU General Public License

A.1. Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software - to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps:

1. copyright the software, and
2. offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

A.2. TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

A.2.1. Section 0

This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

A.2.2. Section 1

You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

A.2.3. Section 2

You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

1. You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.
2. You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.
3. If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. ¹

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

A.2.4. Section 3

You may copy and distribute the Program (or a work based on it, under Section 2 in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the

following:

1. Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
2. Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
3. Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

A.2.5. Section 4

You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program 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.

A.2.6. Section 5

You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

A.2.7. Section 6

Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

A.2.8. Section 7

If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

A.2.9. Section 8

If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

A.2.10. Section 9

The Free Software Foundation may publish revised and/or new versions of the General Public 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.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

A.2.11. Section 10

If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

A.2.12. NO WARRANTY Section 11

BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

A.2.13. Section 12

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO

OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

A.3. How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the program's name and a brief idea of what it does.> Copyright © <year> <name of author>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

Gnomovision version 69, Copyright (C) year name of author Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type 'show w'. This is free software, and you are welcome to redistribute it under certain conditions; type 'show c' for details.

The hypothetical commands 'show w' and 'show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than 'show w' and

'show c'; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program 'Gnomovision' (which makes passes at compilers) written by James Hacker.

<signature of Ty Coon>, 1 April 1989 Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

Notes

1. In general, one source file would depend on more than one header file (like the `main.o` dependency) and the dependency rule is not as simple as this. It is in fact difficult to keep track of dependencies on header files and their dependencies with nested includes. There are ways of handling this (see the manual page of **makedepend** on your Linux machine and the `-M` option to **gcc**), but using **automake** makes this a cake-walk.
1. Exception: If the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.