

Information For Maintainers of GNU Software

Richard Stallman
last updated April 25, 2008

Information for maintainers of GNU software, last updated April 25, 2008.

Copyright © 1992, 1993, 1994, 1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008 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, with no Front-Cover Texts, and with no Back-Cover Texts. A copy of the license is included in the section entitled “GNU Free Documentation License”.

Table of Contents

1	About This Document	1
2	Stepping Down.....	1
3	Recruiting Developers.....	2
4	Legal Matters	2
4.1	Copyright Papers	2
4.2	Legally Significant Changes	4
4.3	Recording Contributors	5
4.4	Copying from Other Packages	6
4.5	Copyright Notices	7
4.6	License Notices.....	8
4.6.1	Canonical License Sources	8
4.6.2	License Notices for Code	8
4.6.3	License Notices for Documentation	9
4.6.4	License Notices for Other Files	10
4.7	External Libraries.....	10
5	Cleaning Up Changes	10
6	Platforms to Support	11
7	Dealing With Mail	12
8	Recording Old Versions.....	13
9	Distributions.....	13
9.1	Distribution tar Files.....	13
9.2	Distribution Patches	14
9.3	Distribution on ftp.gnu.org	15
9.4	Test Releases	15
9.5	Automated FTP Uploads	15
9.5.1	Automated Upload Registration	16
9.5.2	Automated Upload Procedure	16
9.5.3	FTP Upload Directive File - v1.1	17
9.5.4	FTP Upload Directive File - v1.0	18
9.6	Announcing Releases	18

10	Web Pages	19
10.1	Invoking <code>gendocs.sh</code>	20
10.2	CVS Keywords in Web Pages	21
11	Ethical and Philosophical Consideration ...	21
12	Terminology Issues	22
12.1	Free Software and Open Source	22
12.2	GNU and Linux	22
13	Hosting	23
14	Free Software Directory	23
15	Using the Proofreaders List	24
Appendix A	GNU Free Documentation License	24
Index		31

1 About This Document

This file contains guidelines and advice for someone who is the maintainer of a GNU program on behalf of the GNU Project. Everyone is entitled to change and redistribute GNU software; you need not pay attention to this file to get permission. But if you want to maintain a version for widespread distribution, we suggest you follow these guidelines; if you would like to be a GNU maintainer, then it is essential to follow these guidelines.

Please send corrections or suggestions for this document to maintainers@gnu.org. If you make a suggestion, please include a suggested new wording for it, to help us consider the suggestion efficiently. We prefer a context diff to the ‘`maintain.texi`’ file, but if you don’t have that file, you can make a context diff for some other version of this document, or propose it in any way that makes it clear.

This document uses the gender-neutral third-person pronouns “person”, “per”, “pers” and “perself” which were promoted, and perhaps invented, by Marge Piercy in *Woman on the Edge of Time*. They are used just like “she”, “her”, “hers” and “herself”, except that they apply equally to males and females. For example, “Person placed per new program under the GNU GPL, to let the public benefit from per work, and to enable per to feel person has done the right thing.”

The directory ‘`/gd/gnuorg`’ is found on the GNU file server, currently fencepost.gnu.org; if you are the maintainer of a GNU package, you should have an account there. Contact accounts@gnu.org if you don’t have one. (You can also ask for accounts for people who help you a large amount in working on the package.)

This release of the GNU Maintenance Instructions was last updated April 25, 2008.

2 Stepping Down

With good fortune, you will continue maintaining your package for many decades. But sometimes for various reasons maintainers decide to step down.

If you’re the official maintainer of a GNU package and you decide to step down, please inform the GNU Project (maintainers@gnu.org). We need to know that the package no longer has a maintainer, so we can look for and appoint a new maintainer.

If you have an idea for who should take over, please tell maintainers@gnu.org your suggestion. The appointment of a new maintainer needs the GNU Project’s confirmation, but your judgment that a person is capable of doing the job will carry a lot of weight.

As your final act as maintainer, it would be helpful to set up the package under savannah.gnu.org (see [Chapter 8 \[Old Versions\]](#), page 13). This will make it much easier for the new maintainer to pick up where you left off and will ensure that the source tree is not misplaced if it takes us a while to find a new maintainer.

3 Recruiting Developers

Unless your package is a fairly small, you probably won't do all the work on it yourself. Most maintainers recruit other developers to help.

Sometimes people will offer to help. Some of them will be capable, while others will not. It's up to you to determine who provides useful help, and encourage those people to participate more.

Some of the people who offer to help will support the GNU Project, while others may be interested for other reasons. Some will support the goals of the Free Software Movement, but some may not. They are all welcome to help with the work—we don't ask people's views or motivations before they contribute to GNU packages.

As a consequence, you cannot expect all contributors to support the GNU Project, or to have a concern for its policies and standards. So part of your job as maintainer is to exercise your authority on these points when they arise. No matter how much of the work other people do, you are in charge of what goes in the release. When a crucial point arises, you should calmly state your decision and stick to it.

Sometimes a package has several co-maintainers who share the role of maintainer. Unlike developers who help, co-maintainers have actually been appointed jointly as the maintainers of the package, and they carry out the maintainer's functions together. If you would like to propose some of your developers as co-maintainers, please contact maintainers@gnu.org.

4 Legal Matters

This chapter describes procedures you should follow for legal reasons as you maintain the program, to avoid legal difficulties.

4.1 Copyright Papers

If you maintain an FSF-copyrighted package certain legal procedures are required when incorporating legally significant changes written by other people. This ensures that the FSF has the legal right to distribute the package, and the standing to defend its GPL-covered status in court if necessary.

Before incorporating significant changes, make sure that the person who wrote the changes has signed copyright papers and that the Free Software Foundation has received and signed them. We may also need an employer's disclaimer from the person's employer.

To check whether papers have been received, look in `/gd/gnuorg/copyright.list`. If you can't look there directly, fsf-records@gnu.org can check for you. Our clerk can also check for papers that are waiting to be entered and inform you when expected papers arrive.

The directory `/gd/gnuorg` is found on the GNU machines, currently fencepost.gnu.org; if you are the maintainer of a GNU package, you should have an account on them. Contact accounts@gnu.org if you don't have one. (You can also ask for accounts for people who help you a large amount in working on the package.)

In order for the contributor to know person should sign papers, you need to ask per for the necessary papers. If you don't know per well, and you don't know that person is used

to our ways of handling copyright papers, then it might be a good idea to raise the subject with a message like this:

Would you be willing to assign the copyright to the Free Software Foundation,
so that we could install it in *program*?

or

Would you be willing to sign a copyright disclaimer to put this change in the
public domain, so that we can install it in *program*?

If the contributor wants more information, you can send per `/gd/gnuorg/conditions.text`, which explains per options (assign vs. disclaim) and their consequences.

Once the conversation is under way and the contributor is ready for more details, you should send one of the templates that are found in the directory `/gd/gnuorg/Copyright/`; they are also available from the `doc/Copyright/` directory of the `gnulib` project at <http://savannah.gnu.org/projects/gnulib>. This section explains which templates you should use in which circumstances. **Please don't use any of the templates except for those listed here, and please don't change the wording.**

Once the conversation is under way, you can send the contributor the precise wording and instructions by email. Before you do this, make sure to get the current version of the template you will use! We change these templates occasionally—don't keep using an old version.

For large changes, ask the contributor for an assignment. Send per a copy of the file `request-assign.changes`. (Like all the `request-` files, it is in `/gd/gnuorg/Copyright` and in `gnulib`.)

For medium to small changes, request a personal disclaimer by sending per the file `request-disclaim.changes`.

If the contributor is likely to keep making changes, person might want to sign an assignment for all per future changes to the program. So it is useful to offer per that alternative. If person wants to do it that way, send per the `request-assign.future`.

When you send a `request-` file, you don't need to fill in anything before sending it. Just send the file verbatim to the contributor. The file gives per instructions for how to ask the FSF to mail per the papers to sign. The `request-` file also raises the issue of getting an employer's disclaimer from the contributor's employer.

When the contributor emails the form to the FSF, the FSF sends per papers to sign. If person signs them right away, the whole process takes about two weeks—mostly waiting for letters to go back and forth.

For less common cases, we have template files you should send to the contributor. Be sure to fill in the name of the person and the name of the program in these templates, where it says `NAME OF PERSON` and `NAME OF PROGRAM`, before sending; otherwise person might sign without noticing them, and the papers would be useless. Note that in some templates there is more than one place to put the name of the program or the name of the person; be sure to change all of them. All the templates raise the issue of an employer's disclaimer as well.

You do not need to ask for separate papers for a manual that is distributed only in the software package it describes. But if we sometimes distribute the manual separately (for instance, if we publish it as a book), then we need separate legal papers

for changes in the manual. For smaller changes, use `'disclaim.changes.manual'`; for larger ones, use `'assign.changes.manual'`. To cover both past and future changes to a manual, you can use `'assign.future.manual'`. For a translation of a manual, use `'assign.translation.manual'`.

For translations of program strings (as used by GNU Gettext, for example; see [Section “Internationalization” in GNU Coding Standards](#)), use `'disclaim.translation'`. If you make use of the Translation Project (<http://translationproject.org>) facilities, please check with the TP coordinators that they have sent the contributor the papers; if they haven't, then you should send the papers. In any case, you should wait for the confirmation from the FSF that the signed papers have been received and accepted before integrating the new contributor's material, as usual.

If a contributor is reluctant to sign an assignment for a large change, and is willing to sign a disclaimer instead, that is acceptable, so you should offer this alternative if it helps you reach agreement. We prefer an assignment for a larger change, so that we can enforce the GNU GPL for the new text, but a disclaimer is enough to let us use the text.

If you maintain a collection of programs, occasionally someone will contribute an entire separate program or manual that should be added to the collection. Then you can use the files `'request-assign.program'`, `'disclaim.program'`, `'assign.manual'`, and `'disclaim.manual'`. We very much prefer an assignment for a new separate program or manual, unless it is quite small, but a disclaimer is acceptable if the contributor insists on handling the matter that way.

If a contributor wants the FSF to publish only a pseudonym, that is ok. The contributor should say this, and state the desired pseudonym, when answering the `'request-'` form. The actual legal papers will use the real name, but the FSF will publish only the pseudonym. When using one of the other forms, fill in the real name but ask the contributor to discuss the use of a pseudonym with assign@gnu.org before sending back the signed form.

Although there are other templates besides the ones listed here, they are for special circumstances; please do not use them without getting advice from assign@gnu.org.

If you are not sure what to do, then please ask assign@gnu.org for advice; if the contributor asks you questions about the meaning and consequences of the legal papers, and you don't know the answers, you can forward them to assign@gnu.org and we will answer.

Please do not try changing the wording of a template yourself. If you think a change is needed, please talk with assign@gnu.org, and we will work with a lawyer to decide what to do.

4.2 Legally Significant Changes

If a person contributes more than around 15 lines of code and/or text that is legally significant for copyright purposes, we need copyright papers for that contribution, as described above.

A change of just a few lines (less than 15 or so) is not legally significant for copyright. A regular series of repeated changes, such as renaming a symbol, is not legally significant even if the symbol has to be renamed in many places. Keep in mind, however, that a series of minor changes by the same person can add up to a significant contribution. What counts

is the total contribution of the person; it is irrelevant which parts of it were contributed when.

Copyright does not cover ideas. If someone contributes ideas but no text, these ideas may be morally significant as contributions, and worth giving credit for, but they are not significant for copyright purposes. Likewise, bug reports do not count for copyright purposes.

When giving credit to people whose contributions are not legally significant for copyright purposes, be careful to make that fact clear. The credit should clearly say they did not contribute significant code or text.

When people's contributions are not legally significant because they did not write code, do this by stating clearly what their contribution was. For instance, you could write this:

```
/*
 * Ideas by:
 *   Richard Mlynarik <mly@adoc.xerox.com> (1997)
 *   Masatake Yamato <masata-y@is.aist-nara.ac.jp> (1999)
 */
```

Ideas by: makes it clear that Mlynarik and Yamato here contributed only ideas, not code. Without the **Ideas by:** note, several years from now we would find it hard to be sure whether they had contributed code, and we might have to track them down and ask them.

When you record a small patch in a change log file, first search for previous changes by the same person, and see if per past contributions, plus the new one, add up to something legally significant. If so, you should get copyright papers for all per changes before you install the new change.

If that is not so, you can install the small patch. Write '(tiny change)' after the patch author's name, like this:

```
2002-11-04  Robert Fenk  <Robert.Fenk@gmx.de>  (tiny change)
```

4.3 Recording Contributors

Keep correct records of which portions were written by whom. This is very important. These records should say which files or parts of files were written by each person, and which files or parts of files were revised by each person. This should include installation scripts as well as manuals and documentation files—everything.

These records don't need to be as detailed as a change log. They don't need to distinguish work done at different times, only different people. They don't need describe changes in more detail than which files or parts of a file were changed. And they don't need to say anything about the function or purpose of a file or change—the Register of Copyrights doesn't care what the text does, just who wrote or contributed to which parts.

The list should also mention if certain files distributed in the same package are really a separate program.

Only the contributions that are legally significant for copyright purposes (see [Section 4.2 \[Legally Significant\]](#), page 4) need to be listed. Small contributions, bug reports, ideas, etc., can be omitted.

For example, this would describe an early version of GAS:

Dean Elsner first version of all files except gdb-lines.c and m68k.c.
 Jay Fenlason entire files gdb-lines.c and m68k.c, most of app.c,
 plus extensive changes in messages.c, input-file.c, write.c
 and revisions elsewhere.

Note: GAS is distributed with the files obstack.c and obstack.h, but they are considered a separate package, not part of GAS proper.

Please keep these records in a file named ‘AUTHORS’ in the source directory for the program itself.

You can use the change log as the basis for these records, if you wish. Just make sure to record the correct author for each change (the person who wrote the change, *not* the person who installed it), and add ‘(tiny change)’ for those changes that are too trivial to matter for copyright purposes. Later on you can update the ‘AUTHORS’ file from the change log. This can even be done automatically, if you are careful about the formatting of the change log entries.

4.4 Copying from Other Packages

When you copy legally significant code from another free software package with a GPL-compatible license, you should look in the package’s records to find out the authors of the part you are copying, and list them as the contributors of the code that you copied. If all you did was copy it, not write it, then for copyright purposes you are *not* one of the contributors of *this* code.

Especially when code has been released into the public domain, authors sometimes fail to write a license statement in each file. In this case, please first be sure that all the authors of the code have disclaimed copyright interest. Then, when copying the new files into your project, add a brief note at the beginning of the files recording the authors, the public domain status, and anything else relevant.

On the other hand, when merging some public domain code into an existing file covered by the GPL (or LGPL or other free software license), there is no reason to indicate the pieces which are public domain. The notice saying that the whole file is under the GPL (or other license) is legally sufficient.

Using code that is released under a GPL-compatible free license, rather than being in the public domain, may require preserving copyright notices or other steps. Of course, you should do what is needed.

If you are maintaining an FSF-copyrighted package, please verify we have papers for the code you are copying, *before* copying it. If you are copying from another FSF-copyrighted package, then we presumably have papers for that package’s own code, but you must check whether the code you are copying is part of an external library; if that is the case, we don’t have papers for it, so you should not copy it. It can’t hurt in any case to double-check with the developer of that package.

When you are copying code for which we do not already have papers, you need to get papers for it. It may be difficult to get the papers if the code was not written as a contribution to your package, but that doesn’t mean it is ok to do without them. If you cannot get papers for the code, you can only use it as an external library (see [Section 4.7 \[External Libraries\]](#), page 10).

4.5 Copyright Notices

You should maintain a proper copyright notice and a license notice in each nontrivial file in the package. (Any file more than ten lines long is nontrivial for this purpose.) This includes header files and interface definitions for building or running the program, documentation files, and any supporting files. If a file has been explicitly placed in the public domain, then instead of a copyright notice, it should have a notice saying explicitly that it is in the public domain.

Even image files and sound files should contain copyright notices and license notices, if they can. Some formats do not have room for textual annotations; for these files, state the copyright and copying permissions in a README file in the same directory.

Change log files should have a copyright notice and license notice at the end, since new material is added at the beginning but the end remains the end.

When a file is automatically generated from some other file in the distribution, it is useful for the automatic procedure to copy the copyright notice and permission notice of the file it is generated from, if possible. Alternatively, put a notice at the beginning saying which file it is generated from.

A copyright notice looks like this:

```
Copyright (C) year1, year2, year3  copyright-holder
```

The *copyright-holder* may be the Free Software Foundation, Inc., or someone else; you should know who is the copyright holder for your package.

Replace the ‘(C)’ with a C-in-a-circle symbol if it is available. For example, use ‘@copyright{’ in a Texinfo file. However, stick with parenthesized ‘C’ unless you know that C-in-a-circle will work. For example, a program’s standard ‘--version’ message should use parenthesized ‘C’ by default, though message translations may use C-in-a-circle in locales where that symbol is known to work.

To update the list of year numbers, add each year in which you have made nontrivial changes to the package. (Here we assume you’re using a publicly accessible revision control server, so that every revision installed is also immediately and automatically published.) When you add the new year, it is not required to keep track of which files have seen significant changes in the new year and which have not. It is recommended and simpler to add the new year to all files in the package, and be done with it for the rest of the year.

Don’t delete old year numbers, though; they are significant since they indicate when older versions might theoretically go into the public domain, if the movie companies don’t continue buying laws to further extend copyright. If you copy a file into the package from some other program, keep the copyright years that come with the file.

Do not abbreviate the year list using a range; for instance, do not write ‘1996--1998’; instead, write ‘1996, 1997, 1998’.

For files which are regularly copied from another project (such as ‘gnulib’), leave the copyright notice as it is in the original.

The copyright statement may be split across multiple lines, both in source files and in any generated output. This often happens for files with a long history, having many different years of publication.

For an FSF-copyrighted package, if you have followed the procedures to obtain legal papers, each file should have just one copyright holder: the Free Software Foundation, Inc. You should edit the file's copyright notice to list that name and only that name.

But if contributors are not all assigning their copyrights to a single copyright holder, it can easily happen that one file has several copyright holders. Each contributor of nontrivial text is a copyright holder.

In that case, you should always include a copyright notice in the name of main copyright holder of the file. You can also include copyright notices for other copyright holders as well, and this is a good idea for those who have contributed a large amount and for those who specifically ask for notices in their names. (Sometimes the license on code that you copy in may require preserving certain copyright notices.) But you don't have to include a notice for everyone who contributed to the file (which would be rather inconvenient).

Sometimes a program has an overall copyright notice that refers to the whole program. It might be in the 'README' file, or it might be displayed when the program starts up. This copyright notice should mention the year of completion of the most recent major version; it can mention years of completion of previous major versions, but that is optional.

4.6 License Notices

Every nontrivial file needs a license notice as well as the copyright notice. (Without a license notice giving permission to copy and change the file, the file is non-free.)

The package itself should contain a full copy of GPL (conventionally in a file named 'COPYING') and the GNU Free Documentation License (included within your documentation). If the package contains any files distributed under the Lesser GPL, it should contain a full copy of that as well (conventionally in a file named 'COPYING.LESSER').

If you have questions about license issues for your GNU package, please write licensing@gnu.org.

4.6.1 Canonical License Sources

You can get the official versions of these files from several places. You can use whichever is the most convenient for you.

- <http://www.gnu.org/licenses/>.
- The `gnulib` project on savannah.gnu.org, which you can access via anonymous Git or CVS. See <http://savannah.gnu.org/projects/gnulib>.

The official Texinfo sources for the licenses are also available in those same places, so you can include them in your documentation. A GFDL-covered manual should include the GFDL in this way. See [Section "GNU Sample Texts" in *Texinfo*](#), for a full example in a Texinfo manual.

4.6.2 License Notices for Code

Typically the license notice for program files (including build scripts, configure files and makefiles) should cite the GPL, like this:

This file is part of GNU *program*.

GNU *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 3 of the License, or (at your option) any later version. GNU *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, see <http://www.gnu.org/licenses/>.

But in a small program which is just a few files, you can use this instead:

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 3 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, see <http://www.gnu.org/licenses/>.

4.6.3 License Notices for Documentation

Documentation files should have license notices also. Manuals should use the GNU Free Documentation License. Following is an example of the license notice to use after the copyright line(s) using all the features of the GFDL.

```
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 the
Invariant Sections being ‘‘GNU General Public License’’, with the
Front-Cover Texts being ‘‘A GNU Manual’’, and with the Back-Cover Texts
as in (a) below. A copy of the license is included in the section
entitled ‘‘GNU Free Documentation License’’.
```

```
(a) The FSF’s Back-Cover Text is: ‘‘You have the freedom to
copy and modify this GNU manual. Buying copies from the FSF
supports it in developing GNU and promoting software freedom.’’
```

If the FSF does not publish this manual on paper, then omit the last sentence in (a) that talks about copies from GNU Press. If the FSF is not the copyright holder, then replace ‘FSF’ with the appropriate name.

Please adjust the list of invariant sections as appropriate for your manual. If there are none, then say “with no Invariant Sections”. If your manual is not published by the FSF, and under 400 pages, you can omit both cover texts and the inclusion of the GPL.

See Section “GNU Sample Texts” in *Texinfo*, for a full example in a Texinfo manual, and see <http://www.gnu.org/licenses/fdl-howto.html> for more advice about how to use the GNU FDL.

If the manual is over 400 pages, or if the FSF thinks it might be a good choice for publishing on paper, then please include our standard invariant section which explains the importance of free documentation. Write to assign@gnu.org to get a copy of this section.

When you distribute several manuals together in one software package, their on-line forms can share a single copy of the GFDL (see section 6). However, the printed (‘.dvi’, ‘.pdf’, ...) forms should each contain a copy of the GFDL, unless they are set up to be printed and published only together. Therefore, it is usually simplest to include the GFDL in each manual.

4.6.4 License Notices for Other Files

Small supporting files, short manuals (under 300 lines long) and rough documentation (README files, INSTALL files, etc) can use a simple all-permissive license like this one:

```
Copying and distribution of this file, with or without modification,
are permitted in any medium without royalty provided the copyright
notice and this notice are preserved.
```

If your package distributes Autoconf macros that are intended to be used (hence distributed) by third-party packages under possibly incompatible licenses, you may also use the above all-permissive license for these macros.

4.7 External Libraries

When maintaining an FSF-copyrighted GNU package, you may occasionally want to use a general-purpose free software module which offers a useful functionality, as a “library” facility (though the module is not always packaged technically as a library).

In a case like this, it would be unreasonable to ask the author of that module to assign the copyright to the FSF. After all, person did not write it specifically as a contribution to your package, so it would be impertinent to ask per, out of the blue, “Please give the FSF your copyright.”

So the thing to do in this case is to make your program use the module, but not consider it a part of your program. There are two reasonable methods of doing this:

1. Assume the module is already installed on the system, and use it when linking your program. This is only reasonable if the module really has the form of a library.
2. Include the module in your package, putting the source in a separate subdirectory whose ‘README’ file says, “This is not part of the GNU FOO program, but is used with GNU FOO.” Then set up your makefiles to build this module and link it into the executable.

For this method, it is not necessary to treat the module as a library and make a ‘.a’ file from it. You can link with the ‘.o’ files directly in the usual manner.

Both of these methods create an irregularity, and our lawyers have told us to minimize the amount of such irregularity. So consider using these methods only for general-purpose modules that were written for other programs and released separately for general use. For anything that was written as a contribution to your package, please get papers signed.

5 Cleaning Up Changes

Don’t feel obligated to include every change that someone asks you to include. You must judge which changes are improvements—partly based on what you think the users will like,

and partly based on your own judgment of what is better. If you think a change is not good, you should reject it.

If someone sends you changes which are useful, but written in an ugly way or hard to understand and maintain in the future, don't hesitate to ask per to clean up their changes before you merge them. Since the amount of work we can do is limited, the more we convince others to help us work efficiently, the faster GNU will advance.

If the contributor will not or can not make the changes clean enough, then it is legitimate to say "I can't install this in its present form; I can only do so if you clean it up." Invite per to distribute per changes another way, or to find other people to make them clean enough for you to install and maintain.

The only reason to do these cleanups yourself is if (1) it is easy, less work than telling the author what to clean up, or (2) the change is an important one, important enough to be worth the work of cleaning it up.

The GNU Coding Standards are a good thing to send people when you ask them to clean up changes (see [Section "Contents" in GNU Coding Standards](#)). The Emacs Lisp manual contains an appendix that gives coding standards for Emacs Lisp programs; it is good to urge authors to read it (see [Section "Tips and Standards" in The GNU Emacs Lisp Reference Manual](#)).

6 Platforms to Support

Most GNU packages run on a wide range of platforms. These platforms are not equally important.

The most important platforms for a GNU package to support are GNU and GNU/Linux. Developing the GNU operating system is the whole point of the GNU Project; a GNU package exists to make the whole GNU system more powerful. So please keep that goal in mind and let it shape your work. For instance, every new feature you add should work on GNU, and GNU/Linux if possible too. If a new feature only runs on GNU and GNU/Linux, it could still be acceptable. However, a feature that runs only on other systems and not on GNU or GNU/Linux makes no sense in a GNU package.

You will naturally want to keep the program running on all the platforms it supports. But you personally will not have access to most of these platforms—so how should you do it?

Don't worry about trying to get access to all of these platforms. Even if you did have access to all the platforms, it would be inefficient for you to test the program on each platform yourself. Instead, you should test the program on a few platforms, including GNU or GNU/Linux, and let the users test it on the other platforms. You can do this through a pretest phase before the real release; when there is no reason to expect problems, in a package that is mostly portable, you can just make a release and let the users tell you if anything unportable was introduced.

It is important to test the program personally on GNU or GNU/Linux, because these are the most important platforms for a GNU package. If you don't have access to one of these platforms, please ask maintainers@gnu.org to help you out.

Supporting other platforms is optional—we do it when that seems like a good idea, but we don't consider it obligatory. If the users don't take care of a certain platform, you may

have to desupport it unless and until users come forward to help. Conversely, if a user offers changes to support an additional platform, you will probably want to install them, but you don't have to. If you feel the changes are complex and ugly, if you think that they will increase the burden of future maintenance, you can and should reject them. This includes both free or mainly-free platforms such as OpenBSD, FreeBSD, and NetBSD, and non-free platforms such as Windows.

7 Dealing With Mail

Once a program is in use, you will get bug reports for it. Most GNU programs have their own special lists for sending bug reports. The advertised bug-reporting email address should always be `'bug-program@gnu.org'`, to help show users that the program is a GNU package, but it is ok to set up that list to forward to another site for further forwarding. The package distribution should state the name of the bug-reporting list in a prominent place, and ask users to help us by reporting bugs there.

We also have a catch-all list, `bug-gnu-utils@gnu.org`, which is used for all GNU programs that don't have their own specific lists. But nowadays we want to give each program its own bug-reporting list and move away from using `bug-gnu-utils`.

If you are the maintainer of a GNU package, you should have an account on the GNU servers; contact `accounts@gnu.org` if you don't have one. (You can also ask for accounts for people who help you a large amount in working on the package.) With this account, you can edit `'/com/mailler/aliases'` to create a new unmanaged list or add yourself to an existing unmanaged list. A comment near the beginning of that file explains how to create a Mailman-managed mailing list.

But if you don't want to learn how to do those things, you can alternatively ask `alias-file@gnu.org` to add you to the bug-reporting list for your program. To set up a new list, contact `new-mailing-list@gnu.org`. You can subscribe to a list managed by Mailman by sending mail to the corresponding `'-request'` address.

You should moderate postings from non-subscribed addresses on your mailing lists, to prevent propagation of unwanted messages ("spam") to subscribers and to the list archives. For lists controlled by Mailman, you can do this by setting `Privacy Options - Sender Filter - generic_nonmember_action` to `Hold`, and then periodically (daily is best) reviewing the held messages, accepting the real ones and discarding the junk.

When you receive bug reports, keep in mind that bug reports are crucial for your work. If you don't know about problems, you cannot fix them. So always thank each person who sends a bug report.

You don't have an obligation to give more response than that, though. The main purpose of bug reports is to help you contribute to the community by improving the next version of the program. Many of the people who report bugs don't realize this—they think that the point is for you to help them individually. Some will ask you to focus on that *instead of* on making the program better. If you comply with their wishes, you will have been distracted from the job of maintaining the program.

For example, people sometimes report a bug in a vague (and therefore useless) way, and when you ask for more information, they say, "I just wanted to see if you already knew the solution" (in which case the bug report would do nothing to help improve the program).

When this happens, you should explain to them the real purpose of bug reports. (A canned explanation will make this more efficient.)

When people ask you to put your time into helping them use the program, it may seem “helpful” to do what they ask. But it is much *less* helpful than improving the program, which is the maintainer’s real job.

By all means help individual users when you feel like it, if you feel you have the time available. But be careful to limit the amount of time you spend doing this—don’t let it eat away the time you need to maintain the program! Know how to say no; when you are pressed for time, just “thanks for the bug report—I will fix it” is enough response.

Some GNU packages, such as Emacs and GCC, come with advice about how to make bug reports useful. If you want to copy and adapt that, it could be a very useful thing to do.

8 Recording Old Versions

It is very important to keep backup files of all source files of GNU. You can do this using a source control system (such as RCS, CVS, Git, . . .) if you like. The easiest way to use RCS or CVS is via the Version Control library in Emacs (see [Section “Concepts of Version Control”](#) in *The GNU Emacs Manual*).

The history of previous revisions and log entries is very important for future maintainers of the package, so even if you do not make it publicly accessible, be careful not to put anything in the repository or change log that you would not want to hand over to another maintainer some day.

The GNU Project provides a server that GNU software packages can use for source control and other package needs: savannah.gnu.org. You don’t have to use this repository, but if you plan to allow public read-only access to your development sources, it is convenient for people to be able to find various GNU packages in a central place. Savannah is managed by savannah-hackers@gnu.org.

All GNU maintainers are encouraged to take advantage of Savannah, as sharing such a central point can serve to foster a sense of community among GNU developers and help in keeping up with project management.

9 Distributions

It is important to follow the GNU conventions when making GNU software distributions.

9.1 Distribution tar Files

The tar file for version *m.n* of program *foo* should be named ‘*foo-m.n.tar*’. It should unpack into a subdirectory named ‘*foo-m.n*’. Tar files should not unpack into files in the current directory, because this is inconvenient if the user happens to unpack into a directory with other files in it.

Here is how the ‘*Makefile*’ for Bison creates the tar file. This method is good for other programs.

```

dist: bison.info
    echo bison-'sed -e '/version_string/!d' \
        -e 's/[~0-9.]*\([0-9.]*\)*/\1/' -e q version.c' > .fname
    -rm -rf 'cat .fname'
    mkdir 'cat .fname'
    dst='cat .fname'; for f in $(DISTFILES); do \
        ln $(srcdir)/$$f $$dst/$$f || { echo copying $$f; \
            cp -p $(srcdir)/$$f $$dst/$$f ; } \
    done
    tar --gzip -chf 'cat .fname'.tar.gz 'cat .fname'
    -rm -rf 'cat .fname' .fname

```

Source files that are symbolic links to other file systems cannot be installed in the temporary directory using `ln`, so use `cp` if `ln` fails.

Using Automake is a good way to take care of writing the `dist` target.

9.2 Distribution Patches

If the program is large, it is useful to make a set of diffs for each release, against the previous important release.

At the front of the set of diffs, put a short explanation of which version this is for and which previous version it is relative to. Also explain what else people need to do to update the sources properly (for example, delete or rename certain files before installing the diffs).

The purpose of having diffs is that they are small. To keep them small, exclude files that the user can easily update. For example, exclude info files, DVI files, tags tables, output files of Bison or Flex. In Emacs diffs, we exclude compiled Lisp files, leaving it up to the installer to recompile the patched sources.

When you make the diffs, each version should be in a directory suitably named—for example, ‘gcc-2.3.2’ and ‘gcc-2.3.3’. This way, it will be very clear from the diffs themselves which version is which.

If you use GNU `diff` to make the patch, use the options ‘-rc2P’. That will put any new files into the output as “entirely different.” Also, the patch’s context diff headers should have dates and times in Universal Time using traditional Unix format, so that patch recipients can use GNU `patch`’s ‘-Z’ option. For example, you could use the following Bourne shell command to create the patch:

```

LC_ALL=C TZ=UTC0 diff -rc2P gcc-2.3.2 gcc-2.3.3 | \
gzip -9 >gcc-2.3.2-2.3.3.patch.gz

```

If the distribution has subdirectories in it, then the diffs probably include some files in the subdirectories. To help users install such patches reliably, give them precise directions for how to run patch. For example, say this:

```

To apply these patches, cd to the main directory of the program
and then use 'patch -p1'. '-p1' avoids guesswork in choosing
which subdirectory to find each file in.

```

It’s wise to test your patch by applying it to a copy of the old version, and checking that the result exactly matches the new version.

9.3 Distribution on `ftp.gnu.org`

GNU packages are distributed through directory ‘/gnu’ on `ftp.gnu.org`. Each package should have a subdirectory named after the package, and all the distribution files for the package should go in that subdirectory.

See [Section 9.5 \[Automated FTP Uploads\], page 15](#), for procedural details of putting new versions on `ftp.gnu.org`.

9.4 Test Releases

When you release a greatly changed new major version of a program, you might want to do so as a pretest. This means that you make a tar file, but send it only to a group of volunteers that you have recruited. (Use a suitable GNU mailing list/newsgroup to recruit them.)

We normally use the FTP server `alpha.gnu.org` for pretests and prerelease versions. See [Section 9.5 \[Automated FTP Uploads\], page 15](#), for procedural details of putting new versions on `alpha.gnu.org`.

Once a program gets to be widely used and people expect it to work solidly, it is a good idea to do pretest releases before each “real” release.

There are two ways of handling version numbers for pretest versions. One method is to treat them as versions preceding the release you are going to make.

In this method, if you are about to release version 4.6 but you want to do a pretest first, call it 4.5.90. If you need a second pretest, call it 4.5.91, and so on. If you are really unlucky and ten pretests are not enough, after 4.5.99 you could advance to 4.5.990 and so on. (You could also use 4.5.100, but 990 has the advantage of sorting in the right order.)

The other method is to attach a date to the release number that is coming. For a pretest for version 4.6, made on Dec 10, 2002, this would be 4.6.20021210. A second pretest made the same day could be 4.6.20021210.1.

For development snapshots that are not formal pretests, using just the date without the version numbers is ok too.

One thing that you should never do is to release a pretest with the same version number as the planned real release. Many people will look only at the version number (in the tar file name, in the directory name that it unpacks into, or wherever they can find it) to determine whether a tar file is the latest version. People might look at the test release in this way and mistake it for the real release. Therefore, always change the number when you release changed code.

9.5 Automated FTP Uploads

In order to upload new releases to `ftp.gnu.org` or `alpha.gnu.org`, you first need to register the necessary information. Then, you can perform uploads yourself, with no intervention needed by the system administrators.

The general idea is that releases should be cryptographically signed before they are made publicly available.

9.5.1 Automated Upload Registration

Here is how to register your information so you can perform uploads for your GNU package:

1. Create an account for yourself at <http://savannah.gnu.org>, if you don't already have one. By the way, this is also needed to maintain the web pages at www.gnu.org for your project (see [Chapter 10 \[Web Pages\]](#), page 19).
2. In the 'My Account Conf' page on [savannah](http://savannah.gnu.org), upload the GPG key you will use to sign your packages. You can create a key with the command `gpg --gen-key`. (For full information about GPG, see <http://www.gnu.org/software/gpg/>).
3. Send a message, preferably GPG-signed, to ftp-upload@gnu.org with the following:
 1. Name of package(s) that you are the maintainer for, and your preferred email address.
 2. An ASCII armored copy of your GnuPG key, as an attachment. ('`gpg --export -a your_key_id >mykey.asc`' should give you this.)
 3. A list of names and preferred email addresses of other individuals you authorize to make releases for which packages, if any (in the case that you don't make all releases yourself).
 4. ASCII armored copies of GnuPG keys for any individuals listed in (3).

The administrators will acknowledge your message when they have added the proper GPG keys as authorized to upload files for the corresponding packages.

9.5.2 Automated Upload Procedure

Once you have registered your information as described in the previous section, you will be able to do unattended ftp uploads using the following procedure.

For each upload destined for ftp.gnu.org or alpha.gnu.org, three files (a *triplet*) need to be uploaded via ftp to the host ftp-upload.gnu.org.

1. The file to be distributed (for example, 'foo.tar.gz').
2. Detached GPG binary signature for (1), made using '`gpg -b`' (for example, 'foo.tar.gz.sig').
3. A clearsigned *directive file*, made using '`gpg --clearsign`' (for example, 'foo.tar.gz.directive.asc').

The names of the files are important. The signature file must have the same name as the file to be distributed, with an additional '.sig' extension. The directive file must have the same name as the file to be distributed, with an additional '.directive.asc' extension. If you do not follow this naming convention, the upload *will not be processed*.

Since v1.1 of the upload script, it is also possible to upload a *directive file* on its own to perform certain operations on uploaded files. See [Section 9.5.3 \[FTP Upload Directive File - v1.1\]](#), page 17, for more information.

Upload the file(s) via anonymous ftp to ftp-upload.gnu.org. If the upload is destined for ftp.gnu.org, place the file(s) in the '/incoming/ftp' directory. If the upload is destined for alpha.gnu.org, place the file(s) in the '/incoming/alpha' directory.

Uploads are processed every five minutes. Uploads that are in progress while the upload processing script is running are handled properly, so do not worry about the timing of your

upload. Uploaded files that belong to an incomplete triplet are deleted automatically after 24 hours.

Your designated upload email addresses (see [Section 9.5.1 \[Automated Upload Registration\]](#), page 16) are sent a message if there are any problems processing an upload for your package. You also receive a message when your upload has been successfully processed.

If you have difficulties processing an upload, email ftp-upload@gnu.org.

9.5.3 FTP Upload Directive File - v1.1

The directive file name must end in `'directive.asc'`.

When part of a triplet, the directive file must always contain the directives `version`, `directory` and `filename`, as described. In addition, a `'comment'` directive is allowed.

The `version` directive must always have the value `'1.1'`.

The `directory` directive specifies the final destination directory where the uploaded file and its `'.sig'` companion are to be placed.

The `filename` directive must contain the name of the file to be distributed (item (1) above).

For example, as part of an uploaded triplet, a `'foo.tar.gz.directive.asc'` file might contain these lines (before being gpg clearsigned):

```
version: 1.1
directory: bar/v1
filename: foo.tar.gz
comment: hello world!
```

This directory line indicates that `'foo.tar.gz'` and `'foo.tar.gz.sig'` are part of package `bar`. If you uploaded this triplet to `'/incoming/ftp'` and the system positively authenticates the signatures, the files `'foo.tar.gz'` and `'foo.tar.gz.sig'` will be placed in the directory `'gnu/bar/v1'` of the `ftp.gnu.org` site.

The directive file can be used to create currently non-existent directory trees, as long as they are under the package directory for your package (in the example above, that is `bar`).

If you upload a file that already exists in the FTP directory, the original will simply be archived and replaced with the new upload.

Standalone directives

When uploaded by itself, the directive file must contain one or more of the directives `symlink`, `rmsymlink` or `archive`, in addition to the obligatory `directory` and `version` directives. A `filename` directive is not allowed, and a `comment` directive is optional.

If you use more than one directive, the directives are executed in the sequence they are specified in.

Here are a few examples. The first removes a symlink:

```
version: 1.1
directory: bar/v1
rmsymlink: foo-latest.tgz
comment: remove a symlink
```

Archive an old file, taking it offline:

```

version: 1.1
directory: bar/v1
archive: foo-1.1.tar.gz
comment: archive an old file; it will not be available through FTP anymore

```

Archive an old directory (with all contents), taking it offline:

```

version: 1.1
directory: bar/v1
archive: foo
comment: archive an old directory; it will not be available through FTP anymore

```

Create a new symlink:

```

version: 1.1
directory: bar/v1
symlink: foo-1.2.tar.gz foo-latest.tgz
comment: create a new symlink

```

Do everything at once:

```

version: 1.1
directory: bar/v1
rmsymlink: foo-latest.tgz
symlink: foo-1.2.tar.gz foo-latest.tgz
archive: foo-1.1.tar.gz
comment: now do everything at once

```

9.5.4 FTP Upload Directive File - v1.0

As of June 2006, the upload script is running in compatibility mode, allowing uploads with either version 1.1 or version 1.0 of the directive file syntax. Support for v1.0 uploads will be phased out by the end of 2006, so please upgrade to v1.1.

The directive file should contain one line, excluding the clearsigned data GPG that inserts, which specifies the final destination directory where items (1) and (2) are to be placed.

For example, the ‘foo.tar.gz.directive.asc’ file might contain the single line:

```

directory: bar/v1

```

This directory line indicates that ‘foo.tar.gz’ and ‘foo.tar.gz.sig’ are part of package **bar**. If you were to upload the triplet to ‘/incoming/ftp’, and the system can positively authenticate the signatures, then the files ‘foo.tar.gz’ and ‘foo.tar.gz.sig’ will be placed in the directory ‘gnu/bar/v1’ of the ftp.gnu.org site.

The directive file can be used to create currently non-existent directory trees, as long as they are under the package directory for your package (in the example above, that is **bar**).

9.6 Announcing Releases

When you have a new release, please make an announcement. For official new releases, including those made just to fix bugs, we recommend using the (moderated) general GNU announcements list, info-gnu@gnu.org. Doing so makes it easier for users and developers to find the latest GNU releases.

You can maintain your own mailing list for announcements as well if you like. For your own list, you can decide as you see fit what events are worth announcing.

10 Web Pages

Please write web pages about your package for installation on www.gnu.org. They should follow our usual standards for web pages (see <http://www.gnu.org/server>); we chose them in order to support a wide variety of browsers, to focus on information rather than flashy eye candy, and to keep the site simple and uniform.

The simplest way to maintain the web pages for your project is to register the project on savannah.gnu.org. Then you can edit the pages using CVS. You can keep the source files there too, but if you want to use savannah.gnu.org only for the web pages, simply register a “web-only” project.

If you don’t want to use that method, please talk with webmasters@gnu.org about other possible methods. For instance, you can mail them pages to install, if necessary. But that is more work for them, so please use CVS if you can.

Some GNU packages have just simple web pages, but the more information you provide, the better. So please write as much as you usefully can, and put all of it on www.gnu.org. However, pages that access databases (including mail logs and bug tracking) are an exception; set them up on whatever site is convenient for you, and make the pages on www.gnu.org link to that site.

Historically, web pages for GNU packages did not include GIF images, because of patent problems (see [Chapter 11 \[Ethical and Philosophical Consideration\]](#), page 21). Although the GIF patents expired in 2006, using GIF images is still not recommended, as the PNG and JPEG formats are generally superior. See <http://www.gnu.org/philosophy/gif.html>.

If you use a site other than www.gnu.org, please make sure that the site runs on free software alone. (It is ok if the site uses unreleased custom software, since that is free in a trivial sense: there’s only one user and it has the four freedoms.) If the web site for a GNU package runs on non-free software, the public will see this, and it will have the effect of granting legitimacy to the non-free program.

If you use multiple sites, they should all follow that criterion. Please don’t link to a site that is about your package, which the public might perceive as connected with it and reflecting the position of its developers, unless it follows that criterion.

The web pages for the package should include its manuals, in HTML, DVI, Info, PostScript, PDF, plain ASCII, and Texinfo format (source). (All of these can be generated automatically from the Texinfo source using Makeinfo and other programs.) When there is only one manual, put it in a subdirectory called ‘`manual`’; the file ‘`manual/index.html`’ should have a link to the manual in each of its forms.

If the package has more than one manual, put each one in a subdirectory of ‘`manual`’, set up ‘`index.html`’ in each subdirectory to link to that manual in all its forms, and make ‘`manual/index.html`’ link to each manual through its subdirectory.

See the section below for details on a script to make the job of creating all these different formats and index pages easier.

We would like to include links to all these manuals in the page <http://www.gnu.org/manual>. Just send mail to webmasters@gnu.org telling them the name of your package and asking them to edit <http://www.gnu.org/manual>, and they will do so based on the contents of your ‘manual’ directory.

10.1 Invoking gendocs.sh

The script `gendocs.sh` eases the task of generating the Texinfo documentation output for your web pages section above. It has a companion template file, used as the basis for the HTML index pages. Both are available from the Texinfo CVS sources:

<http://savannah.gnu.org/cgi-bin/viewcvs/texinfo/texinfo/util/gendocs.sh>

http://savannah.gnu.org/cgi-bin/viewcvs/texinfo/texinfo/util/gendocs_template

There is also a “minimalistic” template version, available from:

http://savannah.gnu.org/cgi-bin/viewcvs/texinfo/texinfo/util/gendocs_template_min

Invoke the script like this, in the directory containing the Texinfo source:

```
gendocs.sh yourmanual "GNU yourmanual manual"
```

where *yourmanual* is the short name for your package. The script processes the file ‘*yourmanual.texinfo*’ (or ‘.texi’ or ‘.txi’). For example:

```
cd ../emacs/man
# download gendocs.sh and gendocs_template
gendocs.sh emacs "GNU Emacs manual"
```

`gendocs.sh` creates a subdirectory ‘manual/’ containing the manual generated in all the standard output formats: Info, HTML, DVI, and so on, as well as the Texinfo source. You then need to move all those files, retaining the subdirectories, into the web pages for your package.

You can specify the option ‘`-o outdir`’ to override the name ‘manual’. Any previous contents of *outdir* will be deleted.

The second argument, with the description, is included as part of the HTML <title> of the overall ‘manual/index.html’ file. It should include the name of the package being documented, as shown. ‘manual/index.html’ is created by substitution from the file ‘gendocs_template’. (Feel free to modify the generic template for your own purposes.)

If you have several manuals, you’ll need to run this script several times with different arguments, specifying a different output directory with ‘`-o`’ each time, and moving all the output to your web page. Then write (by hand) an overall index.html with links to them all. For example:

```
cd ../texinfo/doc
gendocs.sh -o texinfo texinfo "GNU Texinfo manual"
gendocs.sh -o info info "GNU Info manual"
gendocs.sh -o info-stdn info-stdn "GNU info-stdn manual"
```

By default, the script uses `makeinfo` for generating HTML output. If you prefer to use `texi2html`, use the ‘`--texi2html`’ command line option, e.g.:

```
gendocs --texi2html -o texinfo texinfo "GNU Texinfo manual"
```


The template files will automatically produce entries for additional HTML output generated by `texi2html` (i.e., split by sections and chapters).

You can set the environment variables `MAKEINFO`, `TEXI2DVI`, `TEXI2HTML` and `DVIPS` to control the programs that get executed, and `GENDOCS_TEMPLATE_DIR` to control where the ‘`gendocs_template`’ file is found.

Please email bug reports, enhancement requests, or other correspondence to bug-texinfo@gnu.org.

10.2 CVS Keywords in Web Pages

Since www.gnu.org works through CVS, CVS keywords in your manual, such as `Log`, need special treatment (even if you don’t happen to maintain your manual in CVS).

If these keywords end up in the generated output as literal strings, they will be expanded. The most robust way to handle this is to turn off keyword expansion for such generated files. For existing files, this is done with:

```
cvs admin -ko file1 file2 ...
```

For new files:

```
cvs add -ko file1 file2 ...
```

See [Section “Keyword substitution” in *Version Management with CVS*](#).

In Texinfo source, the recommended way to literally specify a “dollar” keyword is:

```
@w{$}Log$
```

The `@w` prevents keyword expansion in the Texinfo source itself. Also, `makeinfo` notices the `@w` and generates output avoiding the literal keyword string.

11 Ethical and Philosophical Consideration

The GNU project takes a strong stand for software freedom. Many times, this means you’ll need to avoid certain technologies when their use would conflict with our long-term goals.

Software patents threaten the advancement of free software and freedom to program. There are so many software patents in the US that any large program probably implements hundreds of patented techniques, unknown to the program’s developers. It would be futile and self-defeating to try to find and avoid all these patents. But there are some patents which we know are likely to be used to threaten free software, so we make an effort to avoid the patented techniques. If you are concerned about the danger of a patent and would like advice, write to maintainers@gnu.org, and we will try to help you get advice from a lawyer.

Sometimes the GNU project takes a strong stand against a particular patented technology in order to encourage society to reject it.

For example, the MP3 audio format is covered by a software patent in the USA and some other countries. A patent holder has threatened lawsuits against the developers of free programs (these are not GNU programs) to produce and play MP3, and some GNU/Linux distributors are afraid to include them. Development of the programs continues, but we campaign for the rejection of MP3 format in favor of Ogg Vorbis format.

A GNU package should not recommend use of any non-free program, nor should it require a non-free program (such as a non-free compiler or IDE) to build. Thus, a GNU package cannot be written in a programming language that does not have a free software implementation. Now that GNU/Linux systems are widely available, all GNU packages should provide full functionality on a 100% free GNU/Linux system, and should not require any non-free software to build or function. The GNU Coding Standards say a lot more about this issue.

A GNU package should not refer the user to any non-free documentation for free software. The need for free documentation to come with free software is now a major focus of the GNU project; to show that we are serious about the need for free documentation, we must not contradict our position by recommending use of documentation that isn't free.

Finally, new issues concerning the ethics of software freedom come up frequently. We ask that GNU maintainers, at least on matters that pertain specifically to their package, stand with the rest of the GNU project when such issues come up.

12 Terminology Issues

This chapter explains a couple of issues of terminology which are important for correcting two widespread and important misunderstandings about GNU.

12.1 Free Software and Open Source

The terms “free software” and “open source” are the slogans of two different movements which differ in their basic philosophy. The Free Software Movement is idealistic, and raises issues of freedom, ethics, principle and what makes for a good society. The Open Source Movement, founded in 1998, studiously avoids such questions. For more explanation, see <http://www.gnu.org/philosophy/open-source-misses-the-point.html>.

The GNU Project is aligned with the Free Software Movement. This doesn't mean that all GNU contributors and maintainers have to agree; your views on these issues are up to you, and you're entitled to express them when speaking for yourself.

However, due to the much greater publicity that the Open Source Movement receives, the GNU Project needs to overcome a widespread mistaken impression that GNU is *and always was* an activity of the Open Source Movement. For this reason, please use the term “free software”, not “open source”, in GNU software releases, GNU documentation, and announcements and articles that you publish in your role as the maintainer of a GNU package. A reference to the URL given above, to explain the difference, is a useful thing to include as well.

12.2 GNU and Linux

The GNU Project was formed to develop a free Unix-like operating system, GNU. The existence of this system is our major accomplishment. However, the widely used version of the GNU system, in which Linux is used as the kernel, is often called simply “Linux”. As a result, most users don't know about the GNU Project's major accomplishment—or more precisely, they know about it, but don't realize it is the GNU Project's accomplishment and reason for existence. Even people who believe they know the real history often believe that the goal of GNU was to develop “tools” or “utilities.”

To correct this confusion, we have made a years-long effort to distinguish between Linux, the kernel that Linus Torvalds wrote, and GNU/Linux, the operating system that is the combination of GNU and Linux. The resulting increased awareness of what the GNU Project has already done helps every activity of the GNU Project recruit more support and contributors.

Please make this distinction consistently in GNU software releases, GNU documentation, and announcements and articles that you publish in your role as the maintainer of a GNU package. If you want to explain the terminology and its reasons, you can refer to the URL <http://www.gnu.org/gnu/linux-and-gnu.html>.

To contrast the GNU system properly with respect to GNU/Linux, you can call it “GNU/Hurd” or “the GNU/Hurd system.” However, when that contrast is not specifically the focus, please call it just “GNU” or “the GNU system.”

When referring to the collection of servers that is the higher level of the GNU kernel, please call it “the Hurd” or “the GNU Hurd.” Note that this uses a space, not a slash.

13 Hosting

We recommend using savannah.gnu.org for the source code repository for your package, and, even more so, using ftp.gnu.org as the standard distribution site. Doing so makes it easier for developers and users to find the latest GNU releases.

However, it is ok to use other machines if you wish. If you use a company’s machine to hold the repository for your program, or as its ftp site, please put this statement in a prominent place on the site, so as to prevent people from getting the wrong idea about the relationship between the package and the company:

```
The programs <list of them> hosted here are free software packages
of the GNU Project, not products of <company name>. We call them
"free software" because you are free to copy and redistribute them,
following the rules stated in the license of each package. For more
information, see http://www.gnu.org/philosophy/free-sw.html.
```

```
If you are looking for service or support for GNU software, see
http://www.gnu.org/help/gethelp.html for suggestions of where to ask.
```

```
If you would like to contribute to the development of one of these
packages, contact the package maintainer or the bug-reporting address
of the package (which should be listed in the package itself), or look
on www.gnu.org for more information on how to contribute.
```

14 Free Software Directory

The Free Software Directory aims to be a complete list of free software packages, within certain criteria. Every GNU package should be listed there, so please contact bug-directory@gnu.org to ask for information on how to write an entry for your package.

15 Using the Proofreaders List

If you want help finding errors in documentation, or help improving the quality of writing, or if you are not a native speaker of English and want help producing good English documentation, you can use the GNU proofreaders mailing list: proofreaders@gnu.org.

But be careful when you use the list, because there are over 200 people on it. If you simply ask everyone on the list to read your work, there will probably be tremendous duplication of effort by the proofreaders, and you will probably get the same errors reported 100 times. This must be avoided.

Also, the people on the list do not want to get a large amount of mail from it. So do not ever ask people on the list to send mail to the list!

Here are a few methods that seem reasonable to use:

- For something small, mail it to the list, and ask people to pick a random number from 1 to 20, and read it if the number comes out as 10. This way, assuming 50% response, some 5 people will read the piece.
- For a larger work, divide your work into around 20 equal-sized parts, tell people where to get it, and ask each person to pick randomly which part to read.

Be sure to specify the random choice procedure; otherwise people will probably use a mental procedure that is not really random, such as “pick a part near the middle”, and you will not get even coverage.

You can either divide up the work physically, into 20 separate files, or describe a virtual division, such as by sections (if your work has approximately 20 sections). If you do the latter, be sure to be precise about it—for example, do you want the material before the first section heading to count as a section, or not?

- For a job needing special skills, send an explanation of it, and ask people to send you mail if they volunteer for the job. When you get enough volunteers, send another message to the list saying “I have enough volunteers, no more please.”

Appendix A GNU Free Documentation License

Version 1.2, November 2002

Copyright © 2000,2001,2002 Free Software Foundation, Inc.
51 Franklin St, Fifth Floor, Boston, MA 02110-1301, USA

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. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

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

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

1. APPLICABILITY AND DEFINITIONS

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

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

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

The “Invariant Sections” are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

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

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

format is not Transparent if used for any substantial amount of text. A copy that is not “Transparent” is called “Opaque”.

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

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

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

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

2. VERBATIM COPYING

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

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

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document’s license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition.

Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

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

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

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

4. MODIFICATIONS

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

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.

- I. Preserve the section Entitled “History”, Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled “History” in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the “History” section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. For any section Entitled “Acknowledgements” or “Dedications”, Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section Entitled “Endorsements”. Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section to be Entitled “Endorsements” or to conflict in title with any Invariant Section.
- O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version’s license notice. These titles must be distinct from any other section titles.

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

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

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

5. COMBINING DOCUMENTS

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

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

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

6. COLLECTIONS OF DOCUMENTS

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

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

7. AGGREGATION WITH INDEPENDENT WORKS

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

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

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the

license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

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

9. TERMINATION

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

10. FUTURE REVISIONS OF THIS LICENSE

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

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License “or any later version” applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

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

```
Copyright (C)  year  your name.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.2
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover
Texts. A copy of the license is included in the section entitled ‘‘GNU
Free Documentation License’’.
```

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

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

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

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

Index

\$

\$ keywords in web pages..... 21

/

‘/gd/gnuorg’ directory..... 2

A

alpha.gnu.org, ftp site for test releases..... 15

‘AUTHORS’ file..... 6

automake..... 14

B

beta releases..... 15

bug reports..... 12

C

contributions, accepting..... 10

copyright notices in program files..... 7

copyright papers..... 2

cvs keywords in web pages..... 21

CVS repository..... 23

D

data base of GNU copyright assignments..... 2

diff..... 14

Directory, Free Software..... 23

distribution, tar files..... 13

E

email, for receiving bug reports..... 12

ethics..... 21

F

FDL, GNU Free Documentation License..... 24

free software..... 22

Free Software Directory..... 23

FTP site..... 23

ftp uploads, automated..... 15

ftp.gnu.org, the GNU ftp site..... 15

G

gendocs.sh..... 20

generating documentation output..... 20

GNU ftp site..... 15

GNU/Linux..... 22

H

hosting 23

L

legal matters 2
legal papers for changes in manuals 3
license notices in program files 8
Linux 22

M

mailing list for bug reports 12
movements, Free Software and Open Source 22

O

open source 22

P

patch 14
patches, against previous releases 14
philosophy 21
pretest releases 15
proofreading 24

Q

quality of changes suggested by others 10

R

rcs keywords in web pages 21
recording contributors 5
registration 16
repository 23
responding to bug reports 12

S

source repository 23

T

terminology 22
test releases 15
time stamp in diffs 14

U

uploads 16
uploads, registration for 16

V

version control 13
version control system 23

W

web pages 19
web pages, and cvs keywords 21