Information for Maintainers of GNU Software

Richard Stallman
last updated February 26, 2022

Information for maintainers of GNU software, last updated February 26, 2022.

Copyright © 1992, 1993, 1994, 1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2019, 2022 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 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Table of Contents

1 About This Document 1

2 Getting Help 1

3 GNU Accounts and Resources 2

4 Stepping Down 2

5 Recruiting Developers 2

6 Legal Matters 3

6.1 Copyright Papers 3
6.2 Legally Significant Changes 6
6.3 Recording Contributors 7
6.4 Copying from Other Packages 8
6.4.1 Non-FSF-Copyrighted Package 8
6.4.2 FSF-Copyrighted Package 8
6.5 Copyright Notices 8
6.6 License Notices 10
6.6.1 Licensing of GNU Packages 10
6.6.2 Canonical License Sources 11
6.6.3 License Notices for Code 11
6.6.4 License Notices for Documentation 12
6.6.5 License Notices for Code Examples 13
6.6.6 License Notices for Other Files 13
6.7 External Libraries 13
6.8 Crediting Authors 14

7 Cleaning Up Changes 14

8 Platforms to Support 15

9 Patches Not to Accept 16

9.1 Don't Install a Feature Till It Works on GNU 16
9.2 Interoperation with Nonfree Applications 18
9.3 Uninstalled Code in Repo 18

10 Dealing With Mail 19
10.1 Standard Mailing Lists 19
10.2 Creating Mailing Lists 19
10.3 Replying to Mail 20

11 Recording Old Versions 21

12 Distributions 21
12.1 Distribution tar Files 21
12.2 Distribution Patches 22
12.3 Binary Distribution for Nonfree Platforms 22
12.4 Distribution on `ftp.gnu.org` 23
12.5 Test Releases 23
12.6 Automated FTP Uploads 24
12.6.1 Automated Upload Registration 24
12.6.2 Automated Upload Procedure 25
12.6.3 FTP Upload Release File Triplet 25
12.6.4 FTP Upload Directive File 26
12.6.5 FTP Upload Directory Trees 27
12.6.6 FTP Upload File Replacement 27
12.6.7 FTP Upload Standalone Directives 27
12.6.8 FTP Upload Directive File - v1.1 29
12.6.9 FTP Upload Directive File - v1.0 29
12.7 Announcing Releases 29

13 Web Pages 30
13.1 Hosting for Web Pages 30
13.2 Freedom for Web Pages 31
13.3 Manuals on Web Pages 31
13.3.1 Invoking `gendocs.sh` 32
13.4 CVS Keywords in Web Pages 33

14 Ethical and Philosophical Consideration 33

15 Humor and GNU 34

16 Other Politics 35

17 Terminology Issues 35
17.1 Free Software and Open Source 35
17.2 GNU and Linux 35

18 Interviews and Speeches 36

19 Hosting **37**

20 Donations **38**

21 Free Software Directory **38**

22 Using the Proofreaders List **39**

Appendix A Suggested Responses **39**

Appendix B GNU Free Documentation License .. **40**

Index **48**

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 are or would like to be a GNU maintainer, then it is essential to follow these guidelines.

In addition to this document, please read and follow the GNU Coding Standards (see Section "Contents" in *GNU Coding Standards*). You may also usefully check the "Tips for new GNU maintainers" (`https://www.gnu.org/software/maintainer-tips`), a list of the most important things you will need to do as a new maintainer.

Please send corrections or suggestions for this document to `bug-standards@gnu.org`. If you make a suggestion, please include suggested new wording if you can. We prefer a context diff to the Texinfo source, but if that's difficult for you, you can make a diff for some other version of this document, or propose it in any way that makes it clear. The source repository for this document can be found at `https://savannah.gnu.org/projects/gnustandards`.

If you want to receive diffs for every change to these GNU documents, join the mailing list `gnustandards-commit@gnu.org`, for instance via the web interface at `https://lists.gnu.org/mailman/listinfo/gnustandards-commit`. Archives are also available there.

This document uses the gender-neutral third-person pronouns "person" (which can be shortened to "perse"), "per", "pers" and "perself." These pronouns (aside from "perse") 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 regardless of gender. For example, "Person placed per new program under the GNU GPL, to maintain freedom for all users of per work, and this way perse knows perse has done the right thing."

This release of the GNU Maintainer Information was last updated February 26, 2022.

2 Getting Help

If you have any general questions or encounter a situation where it isn't clear how to get something done or who to ask, you (as a GNU contributor) can always write to `mentors@gnu.org`, which is a list of a few experienced GNU folks who have volunteered to answer questions. Any GNU-related question is fair game for the `mentors` list.

The GNU Advisory Committee helps to coordinate activities in the GNU project on behalf of RMS (Richard Stallman, the Chief GNUisance). If you have any organizational questions or concerns you can contact the committee at `gnu-advisory@gnu.org`. See `https://www.gnu.org/contact/gnu-advisory.html` for the current committee members. Additional information is in `/gd/gnuorg/advisory`.

If you find that any GNU computer systems (`fencepost.gnu.org`, `ftp.gnu.org`, `www.gnu.org`, `savannah.gnu.org`, ...) seem to be down, you can check the current status at `https://hostux.social/@fsfstatus`. Most likely the problem, if it can be alleviated at the FSF end, is already being worked on.

The FSF system administrators maintain the GNU network and server hardware. You can email them at `sysadmin@fsf.org`. Please report any failures in GNU servers to them without delay. Aside from that, please try not to burden them unnecessarily.

3 GNU Accounts and Resources

The directory `/gd/gnuorg` mentioned throughout this document is available on the general GNU server, currently `fencepost.gnu.org`. If you are the maintainer of a GNU package, you should have an account there. If you don't have one already, see `https://www.gnu.org/software/README.accounts.html`. Such GNU login accounts include email (see `https://www.fsf.org/about/systems/sending-mail-via-fencepost`).

You can request for accounts for people who significantly help you in working on the package; we will do this in special cases when there is a good reason.

Other resources available to GNU maintainers are described at `https://www.gnu.org/software/devel.html`, as well as throughout this document. In brief:

- Login accounts (see above).
- Version control (see Chapter 11 [Old Versions], page 21).
- Mailing lists (see Chapter 10 [Mail], page 19).
- Web pages (see Chapter 13 [Web Pages], page 30).
- Mirrored release areas (see Chapter 12 [Distributions], page 21).
- Pre-release portability testing, both automated (via Hydra) and on request (via volunteers).

4 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 or update the package under `savannah.gnu.org` (see Chapter 11 [Old Versions], page 21). 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.

5 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`.

We're happy to acknowledge all major contributors to GNU packages on the `https://www.gnu.org/people/people.html` web page. Please send an entry for yourself to `webmasters@gnu.org`, and feel free to suggest it to other significant developers on your package.

6 Legal Matters

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

6.1 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.

GNU packages need not be FSF-copyrighted; this is up to the author(s), generally at the time the package is dubbed GNU. When copyright is assigned to the FSF, the FSF can act to stop GPL violations about the package. Otherwise, legal actions are up to the author(s). The rest of this section is about the case when a package is FSF-copyrighted.

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, which confirms that the work was not part of the person's job and the employer makes no claim on it. However, a copy of the person's employment contract, showing that the employer can't claim any rights to this work, is often sufficient.

If the employer does claim the work was part of the person's job, and there is no clear basis to say that claim is invalid, then we have to consider it valid. Then the person cannot assign copyright, but the employer can. Many companies have done this. Please ask the appropriate managers to contact `assign@gnu.org`.

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` mentioned throughout this document is available on the general GNU server, currently `fencepost.gnu.org`. If you are the maintainer of a GNU package, you should have an account there. If you don't have one already, see `https://www.gnu.org/software/README.accounts.html`. Such GNU login accounts include email (see `https://www.fsf.org/about/systems/sending-mail-via-fencepost`).

You can request for accounts for people who significantly help you in working on the package; we will do this in special cases when there is a good reason.

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 *package*?

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 *package*?

If the contributor then wants more information, you can send per the file `/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 `https://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 an electronic (usually PDF) copy of the assignment. This, or whatever response is required, should happen within five business days of the initial request. If no reply from the FSF comes

after that time, please send a reminder. If there is still no response after an additional week, please write to `maintainers@gnu.org` about it.

After receiving the necessary form, the contributor needs to sign it. Contributors residing in the USA or Italy may use GPG in order to sign their assignment. Contributors located anywhere else can print, sign, and then email (or fax) a scanned copy back to the FSF. (Specific instructions for both cases are sent with the assignment form.) They may use postal mail, if they prefer. To emphasize, the necessary distinction is between residents and non-residents of these countries; citizenship does not matter.

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 (`https://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.

When a copyright holder has signed an assignment for all future changes to the package, and contributes a change made up of new files which require no change to any of the old files, we want to avoid any uncertainty about whether these files are intended as a change to the package and thus covered by that assignment. The way to do this is to ask the contributor to say so in a message to you — for instance, "My modules 'frog' and 'kangaroo' are intended as changes to the program Hoppers." Forward the message to `assign@gnu.org`, who will

save it permanently. A variation on this procedure: the contributor who wrote the new files can send copies of the new files which contain such a message.

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.

6.2 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)
```

6.3 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 6.2 [Legally Significant], page 6) 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.

It is ok to include other email addresses, names, and program information in `AUTHORS`, such as bug-reporting information. See Section 10.1 [Standard Mailing Lists], page 19.

6.4 Copying from Other Packages

This section explains legal considerations when merging code from other packages into your package. Using an entire module as a whole, and maintaining its separate identity, is a different issue; see Section 6.7 [External Libraries], page 13.

6.4.1 Non-FSF-Copyrighted Package

Here we suppose that your package is not FSF-copyrighted.

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.

If the code is supposed to be in the public domain, make sure that is really true: 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 not in the public domain, but rather released under a GPL-compatible free license, may require preserving copyright notices or other steps. Of course, you should follow the requirements stated.

6.4.2 FSF-Copyrighted Package

If you are maintaining an FSF-copyrighted package, please don't copy in any code without verifying first that we have suitable legal papers for that code.

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 6.7 [External Libraries], page 13).

6.5 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 their format permits. 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 word '`Copyright`' must always be in English, by international convention.

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. Alternatively, the '`(C)`' or C-in-a-circle can be omitted entirely; the word '`Copyright`' suffices.

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.

You can use a range ('`2008-2010`') instead of listing individual years ('`2008, 2009, 2010`') if and only if: 1) every year in the range, inclusive, really is a "copyrightable" year that would be listed individually; *and* 2) you make an explicit statement in a `README` file about this usage.

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.

6.6 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 nonfree.)

The package itself should contain a full copy of GPL in plain text (conventionally in a file named `COPYING`) and the GNU Free Documentation License (included within your documentation, so there is no need for a separate plain text version). If the package contains any files distributed under the Lesser GPL, it should contain a full copy of its plain text version also (conventionally in a file named `COPYING.LESSER`).

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

6.6.1 Licensing of GNU Packages

Normally, GNU packages should use the latest version of the GNU GPL, with the "or any later version" formulation. See Section 6.6.3 [License Notices for Code], page 11, for the exact wording of the license notice.

Occasionally, a GNU library may provide functionality which is already widely available to proprietary programs through alternative implementations; for example, the GNU C Library. In such cases, the Lesser GPL should be used (again, for the notice wording, see Section 6.6.3 [License Notices for Code], page 11). If a GNU library provides unique functionality, however, the GNU GPL should be used. `https://www.gnu.org/licenses/why-not-lgpl.html` discusses this strategic choice.

Some of these libraries need to work with programs released under GPLv2-only; that is, which allow the GNU GPL version 2 but not later versions. In this case, the GNU package should be released under a dual license: GNU GPL version 2 (or any later version) and the GNU Lesser GPL version 3 (or any later version). Here is the notice for that case:

```
This file is part of GNU package.

GNU package is free software: you can redistribute it and/or
modify it under the terms of either:

  * the GNU Lesser General Public License as published by the Free
    Software Foundation; either version 3 of the License, or (at your
```

```
    option) any later version.

or

  * the GNU General Public License as published by the Free
    Software Foundation; either version 2 of the License, or (at your
    option) any later version.

or both in parallel, as here.

GNU package 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 copies of the GNU General Public License and
the GNU Lesser General Public License along with this program.  If
not, see https://www.gnu.org/licenses/.
```

For small packages, you can use "This program" instead of "GNU *package*".

6.6.2 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.

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

6.6.3 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 *package*.
>
> GNU *package* 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 *package* 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 `https://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 `https://www.gnu.org/licenses/`.

In either case, for those few packages which use the Lesser GPL (see Section 6.6.1 [Licensing of GNU Packages], page 10), insert the word "Lesser" before "General" in *all three* places. `https://www.gnu.org/licenses/gpl-howto.html` discusses application the GPL in more detail.

6.6.4 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 1.3 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. However, if it is copyright FSF, always ask the FSF what to do.

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

If you write a manual that people might want to buy on paper, please write to `maintainers@gnu.org` to tell the FSF about it. We might want to publish it.

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 the GNU GPL, as in the notice above. Please also 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.

6.6.5 License Notices for Code Examples

When a code example in documentation is more than two or three lines, and specific enough that people might want to copy and adapt it, we suggest putting a copy of the example in a file of code and releasing that under some free software license. That means it will be released under two different licenses: in the manual under the GFDL, and in the code example file under a software license.

If the example is important and nontrivial, and 40 lines or more, we suggest releasing the code copy under the same license as the program it pertains to. Otherwise, we recommend releasing it under the X11 license.

6.6.6 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.  This file is offered as-is,
without any warranty.
```

Older versions of this license did not have the second sentence with the express warranty disclaimer. There is no urgent need to update existing files, but new files should use the new text.

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.

These kinds of files can also be put in the public domain. If publishing in the US, it is enough to insert a notice saying so. Otherwise, use Creative Commons's CC0—See `https://creativecommons.org/choose/zero/`.

6.7 External Libraries

As maintainer of an FSF-copyrighted GNU package, how do you use separately-published general-purpose free modules? (We also call them "libraries" because we are using them as libraries; it doesn't matter whether they are packaged as libraries or not.)

It would be unreasonable to ask their authors to assign copyright to the FSF. They didn't write those modules as contributions to GNU. We just happen to want to use them, as any developer might. It would be rude to ask developers, out of the blue, to give the FSF their copyright. Please don't ask for that in cases like these.

The proper way to use those modules is to link your package with them and say they are *not* part of your package. See below for the mechanics of this.

To avoid present or future legal trouble, you must make sure the license of the module is compatible with current *and future* GPL versions. "GNU GPL version 3 or later" is good, and so is anything which includes permission for use under those GPL versions (including "GNU GPL version 2 or later", "LGPL version *n* or later", "LGPL version 2.1", "GNU Affero GPL version 3 or later"). Lax permissive licenses are ok too, since they are compatible with all GPL versions.

"GPL version 2 only" is obviously unacceptable because it is incompatible with GPL version 3. "GPL version 3 only" and "GPL version 2 or 3 only" have a subtler problem:

they would be incompatible with GPL version 4, if we ever make one, so the module would become an obstacle to upgrading your package's license to "GPL version 4 or later". Don't use such modules.

One library you need to avoid is `goffice`, since it allows only GPL versions 2 and 3.

So, here are the mechanics of how to arrange your package to use the unrelated free module.

1. Assume the module is already installed on the system, and link with 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's distribution, 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.

   With this method, it is not necessary to treat the module as a library and make a '`.a`' file from it. You can link directly with the '`.o`' files 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 use these methods only for general-purpose modules that were *not* written for your package. For anything that was written as a contribution to your package, please get papers signed.

6.8 Crediting Authors

Strictly speaking, this is not a legal issue, but it seems to belong with copyright notices.

In any FSF-copyrighted GNU package, the authors of a file are not named in the copyright notice. Therefore, it is nice to include a comment line '`Authors: authors of this file`' at the top near the copyright notice, to give them credit in close association with their contribution.

7 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 Lisp authors to read it (see Section "Tips and Conventions" in *The GNU Emacs Lisp Reference Manual*).

8 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 the free variants of the GNU operating system, regardless of which kernel it uses.

The GNU Project's practical work is developing the GNU operating system; a GNU package should make the whole GNU system more powerful and encourage people to switch to that system.

Please keep those goals in mind in your work. For instance, every new feature you add should work on GNU. If a new feature runs only on GNU (for instance, on GNU/Linux), it is acceptable. However, a feature that runs only on other systems, and not on GNU, would undermine the goal.

Therefore, please say no when asked to implement such a feature, citing these reasons, and ask the contributors to implement the feature for the GNU system as well. See Chapter 9 [Patches Not to Accept], page 16.

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 handle them?

Don't worry about trying to get access to all of these platforms. Even if you did have access to all of them, 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 some free variants of GNU, 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, especially 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, as a GNU maintainer you can get access to the general GNU login machine; see `https://www.gnu.org/software/README.accounts.html`.

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 nonfree platforms such as Windows.

9 Patches Not to Accept

Maintaining a program includes receiving suggested patches from users and deciding which of them to install. For the most part that is a matter of technical benefits and drawbacks, which you as maintainer will weigh and judge.

However, there are certain patches which have fundamental moral problems, so you should not accept them unless/until those problems are fixed.

9.1 Don't Install a Feature Till It Works on GNU

Please *don't* write or install code for features that would have worse or less functionality (or none) on the GNU system than on some non-GNU systems.

The primary purpose of any GNU program is to enhance the capability of the GNU system to give users freedom, so every feature of a GNU package should be usable and useful on free distributions of the GNU operating system (`https://www.gnu.org/distros/`). For this purpose, a "feature" is an operation which does something substantially useful for the user and not the technical details of an implementation. We explain this point further below.

A feature that functions only on, or functions better on, some non-GNU operating system would undermine that primary purpose; worse, it would promote that non-GNU system at the expense of GNU. Such a situation would work directly against the goal of liberating users from those systems, even though installing features that create such a situation would be seen as desirable in terms of the "open source" philosophy.

Since freedom in use of computing is the overall purpose, we need to aim clearly for that freedom. We need to show with our practical decisions—and with our explanations of them—that we're working for something deeper and more important than "better software" and "more convenience." That will give users a chance to reflect about our reasons for taking a path different from what most programmers would do. See `https://www.gnu.org/philosophy/open-source-misses-the-point.html`.

Therefore, when you as a GNU maintainer receive contributions of features that do not work on the GNU system, please explain this rule and why it is necessary. Explain that we need all features in GNU packages to work properly on the GNU system, so that they potentiate each other and make the GNU system better. Thus, the change should not be installed in its present form.

That doesn't mean the change can't be installed *ever*. It could be installed later, if and when equally good support on the GNU system for the same feature can be installed at the same time. Explaining this is a good opportunity to ask people to help write the code to support the feature on GNU. Also inform the contributor about resources to learn about how to support this feature on GNU, so perse could consider doing that job—or recruiting others to help.

If parts of the code are system-independent, and will do part of the job of supporting the feature on the GNU system, you can install them right away. Or you can put them in the package repo without actually installing them, in a '`wip-`*`name`*' branch. Having them in the repository may help and encourage people to finish implementing the feature on GNU.

If you think it is very important, you can implement the support for that feature on the GNU system yourself. If you think it is highly desirable to have that feature on GNU

someday, you can make special arrangements to put the non-GNU system-specific code in the package repo but not install it—see Section 9.3 [Uninstalled Code in Repo], page 18.

It is ok to implement or support a feature for non-GNU systems if the feature works at least as well on GNU, even if it is implemented differently on different systems, uses different system facilities in its implementation, or looks different to the user in some details. It is ok to implement those little details, on each system, in the way that is convenient or conventional for making the features work. The point is that the program and the feature should "run best on GNU."

If system facilities on some other system provide, without any special application code, a feature not available on GNU, there is no need to do work to prevent it from functioning. In that case, we should work to implement that feature in GNU.

We don't consider the little details of interfaces to control or configure specific operations, or details of implementing operations, as "features." Likewise, a system facility (including libraries that come with the system) is a means for implementing features but use of the facility is not in itself a feature.

For instance, a programming platform often offers an interface to control the computer or the operating system at a low level. It is OK to support the feature of low-level control on a non-GNU system provided the package supports the same capabilities on the GNU system also, even if the details of how to invoke the feature vary from system to system. But do undertake to make the invocation of this feature consistent across systems, for the specific actions that are supported on multiple systems.

Features mainly for communicating with other users' computers, or between computers not set up for tightly-coupled use as a group, are a different matter entirely. A communication feature is truly "the same feature" as on GNU only if it interoperates with a free distribution of the GNU system—as, for instance, TCP/IP does. Unportable, system-specific communication facilities for non-GNU systems are abuse of the community, so don't install support for them. This point also applies to file formats used for communication between programs, if there is ever an occasion to move those files between unrelated computers. (Exception: it is admirable to write code to extract the user's data from such a file, if you can do it.)

Finally, please be careful not to let installing or supporting system-specific code for non-GNU systems consume much of your own time. See Section "System Portability" in *GNU Coding Standards*.

Suppose people ask for support for feature F on some non-GNU system, and feature F does work on GNU. You can say yes if you wish, but you have no obligation to write or maintain that code. You can tell them that it's their responsibility to write it and maintain it. If they write good clean code for it, you can approve its installation, but that doesn't mean you or anyone else is obliged to support it. If someday the code suffers from bitrot, you can delete it if users don't fix it.

See Appendix A [Suggested Responses], page 39, for some text you can use or adapt, if you like, to say no to these patches. It aims to invite them to support the GNU system equally well in the new feature. If there is no hope of that, just "No thanks" is enough.

9.2 Interoperation with Nonfree Applications

It is quite usual to implement features in GNU programs to make them work conveniently with widely used nonfree tools and applications. But there are situations where you should not implement cooperation with a nonfree program, which we can refer to here as ShackleMe.

- If ShackleMe is not well-known, reject the idea. GNU packages should not even *mention* an obscure nonfree program (see Section "References" in *GNU Coding Standards*).
- If ShackleMe imposes something particularly nasty or dangerous, such as effective DRM or monopolistic file formats, you should refuse to give it any specific support. But don't cripple general features so that they refuse to work with ShackleMe; that would be excessive. It is ok to write code to extract the user's data from such files, if that is possible.
- If ShackleMe does not run on the GNU operating system, and there is no comparable free program that people could use on the GNU system to do the same job, special support for ShackleMe would be a feature that works on non-GNU systems only. Thus, you should refuse to support it. See Section 9.1 [Non-GNU-only Features], page 16.
- If ShackleMe runs on GNU systems also, you can include support for it if you wish, but you don't have an obligation to include that, let alone ever to *run* it. If you do include support for it, make sure the support for communicating with it works as well on the GNU system as it does on non-GNU systems.
- If there are free programs that can replace ShackleMe, or try to, make sure your program works with them as well as it is reported to work with ShackleMe, or better.
- You never have an obligation to write, install or maintain any sort of support for a nonfree program. If it is unmaintained and breaks, and nobody else wants to maintain it you can delete it. Don't feel trapped into working on it!

See Appendix A [Suggested Responses], page 39, for text you can use, if you wish, to state your refusal to support ShackleMe without equally good support for ShackleMe's free competitors. Its purpose is to invite the contributors to support those. You can modify it as needed to fit the situation.

9.3 Uninstalled Code in Repo

When you want to put system-dependent code for a non-GNU feature into the package repository, without actually installing it, you need to make special arrangements with the GNU Project.

To do that, you write to `maintainers@gnu.org` and explain the feature, its dependance on some other system, and the obstacle that has prevented supporting it on GNU. They will make sure you understand the situation and the arrangements, and get your commitment to make the branch fade away later, in the proper way, if the feature goes unfinished.

Practically speaking, these special arrangements mean you put the code in the package repository in a *discouraged branch* to show it is *not* installed, that you have no commitment to finish it, and that it might fade away. Name the branch '`ungnu-temp/`*`name`*'. (If that name doesn't fit with the version control system you use, we will work out a solution.)

Put in the branch a `README` file saying this:

```
This code partially implements the what is it feature.  We can't
```

```
install it now because it needs to be finished, so that it runs on the
GNU system.

We invite you to write the missing code to implement this feature on
GNU, so we can install the feature.  Until then, this branch must not
be merged into any branch that might ever be released.

See the section "Don't Install a Feature Until It Works on GNU", in the
GNU Maintainer's Guide, for explanation of the reasons for this.
```

The discouraged branch "fades away" because you don't merge in changes from the program's trunk of development. If the branch gets too obsolete to work at all, you simply delete it.

10 Dealing With Mail

This chapter describes setting up mailing lists for your package, and gives advice on how to handle bug reports and random requests once you have them.

10.1 Standard Mailing Lists

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-`*package*`@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 if you prefer.

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`.

See Section 10.3 [Replying to Mail], page 20, for more about handling and tracking bug reports.

Some GNU programs with many users have a help list, '`help-`*package*`@gnu.org`', for people to ask other users for help. If your program has many users, you should create such a list for it. For a fairly new program, which doesn't have a large user base yet, it is better not to bother with this.

If you wish, you can also have a mailing list '`info-`*package*`@gnu.org`' for announcements (see Section 12.7 [Announcements], page 29). Any other mailing lists you find useful can also be created.

The package distribution should state the name of all the package's mailing lists in a prominent place, and ask users to help us by reporting bugs appropriately. The top-level `README` file and/or `AUTHORS` file are good places. Mailing list information should also be included in the manual and the package web pages (see Chapter 13 [Web Pages], page 30).

10.2 Creating Mailing Lists

Using the web interface on `savannah.gnu.org` is by far the easiest way to create normal mailing lists, managed through Mailman on the GNU mail server. Once you register your package on Savannah, you can create (and remove) lists yourself through the 'Mailing Lists' menu, without needing to wait for intervention by anyone else. Furthermore, lists created

through Savannah will have a reasonable default configuration for antispam purposes (see below).

To create and maintain simple aliases and unmanaged lists, you can edit `/com/mailer/aliases` on the main GNU server. If you don't have an account there, please read `https://www.gnu.org/software/README.accounts.html` (see Chapter 3 [GNU Accounts and Resources], page 2).

But if you don't want to learn how to do those things, you can ask `new-mailing-list@gnu.org` to help you.

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.

Lists created through Savannah will have this setting, and a number of others, such that spam will be automatically deleted (after a short delay). The Savannah mailing list page describes all the details. You should still review the held messages in order to approve any that are real.

10.3 Replying to Mail

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. Copying and adapting that could be very useful for your package.

If you would like to use an email-based bug tracking system, see `https://bugs.gnu.org`; this can be connected with the regular bug-reporting address. Alternatively, if you

would like to use a web-based bug tracking system, Savannah supports this (see Chapter 11 [Old Versions], page 21), but please don't fail to accept bugs by regular email as well—we don't want to put up unnecessary barriers against users submitting reports.

11 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 Bazaar, RCS, CVS, Git, Subversion, . . .) if you like. An easy way to use many such systems is via the Version Control library in Emacs (see Section "Introduction to 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 packages can use for source control and other package needs: `savannah.gnu.org`. Savannah is managed by `savannah-hackers@gnu.org`. For more details on using and contributing to Savannah, see `https://savannah.gnu.org/maintenance`.

It's not an absolute requirement, but all GNU maintainers are strongly encouraged to take advantage of Savannah, as sharing such a central point can serve to foster a sense of community among GNU developers as well as help in keeping up with project management. Please don't mark Savannah projects for GNU packages as private; that defeats a large part of the purpose of using Savannah in the first place.

If you do use Savannah, please subscribe to the `savannah-announce@gnu.org` mailing list (`https://lists.gnu.org/mailman/listinfo/savannah-announce`). This is a very low-volume list to keep Savannah users informed of system upgrades, problems, and the like.

12 Distributions

Please follow the GNU conventions when making GNU software distributions.

12.1 Distribution tar Files

All packages should provide tar files for the distribution of their releases. 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.

12.2 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.

12.3 Binary Distribution for Nonfree Platforms

Some package maintainers release pre-compiled binaries for proprietary systems such as Microsoft Windows or MacOS. It’s entirely up to you whether to do that; we don’t ask you

to do it, but we don't object. Please do not let anyone make you feel you have an obligation to do this.

If you distribute them, please inform their users prominently that those nonfree platforms trample their freedom. It is useful to refer them to `https://www.gnu.org/philosophy/free-software-even-more-important.html`. You can say, "This program respects your freedom, but Windows does not. To have freedom, you need to stop using Windows and other software that denies your freedom."

12.4 Distribution on `ftp.gnu.org`

We strongly recommend using `ftp.gnu.org` to distribute official releases. If you want to also distribute the package from a site of your own, that is fine. To use some other site instead of `ftp.gnu.org` is acceptable, provided it allows connections from anyone anywhere.

See Section 12.6 [Automated FTP Uploads], page 24, for the procedural details of putting new versions on `ftp.gnu.org`.

12.5 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 server `alpha.gnu.org` for pretests and prerelease versions. See Section 12.6 [Automated FTP Uploads], page 24, for the 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 three 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.)

Another 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.

A third method, if the package uses Git, is to run the script `build-aux/git-version-gen` from Gnulib to generate test release version numbers. It generates version numbers in the form '*`version.commits-commithash`*', where *version* is the latest version tag, *commits* is the number of commits since that tag, and *commithash* is a hash code for the latest commit.

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.

12.6 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.

12.6.1 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 `https://savannah.gnu.org`, and register your package there, if you haven't already done that. By the way, this is also needed to maintain the web pages at `https://www.gnu.org` for your project (see Chapter 13 [Web Pages], page 30).
2. Compose a message with the following items in some *msgfile*. Then GPG-sign it by running `gpg --clearsign` *`msgfile`*, and finally email the resulting *`msgfile`*`.asc` as an attachment to `ftp-upload@gnu.org`.
    1. Name of package(s) that you are the maintainer for, your preferred email address, and your Savannah username.
    2. The ASCII armored copy of your GPG key, as an attachment.
    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 GPG keys for any individuals listed in (3).
3. Publish the concatenated ASCII armored copies of your GPG key with the GPG keys listed in the previous step in the '`GPG Keys Used for Releases`' area of the '`Public info`' of the Savannah group of your package.

    Optional but recommended: Send your keys to a GPG public key server: `gpg --keyserver keys.gnupg.net --send-keys` *`keyid...`*, where *keyid* is the eight hex digits reported by `gpg --list-public-keys` on the `pub` line before the date. For full information about GPG, see `https://www.gnu.org/software/gpg`.

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

The upload system will email receipts to the given email addresses when an upload is made, either successfully or unsuccessfully.

Should you later have to update your GPG key, you'll have to re-submit it to both Savannah and `ftp-upload@gnu.org`, as these systems are not connected.

12.6.2 Automated Upload Procedure

Once you have registered your information as described in the previous section, you can and should do ftp uploads for your package. There are two basic kinds of uploads (details in the following sections):

1. Three related files (a "triplet") to upload a file destined for `ftp.gnu.org` or `alpha.gnu.org`: see Section 12.6.3 [FTP Upload Release File Triplet], page 25.
2. A single (signed) standalone "directive file" to perform operations on the server: see Section 12.6.7 [FTP Upload Standalone Directives], page 27.

In either case, you upload the file(s) via anonymous ftp to the host `ftp-upload.gnu.org`. If the upload is destined for `ftp.gnu.org`, place the file(s) in the directory `/incoming/ftp`. If the upload is destined for `alpha.gnu.org`, place the file(s) in the directory `/incoming/alpha`.

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. Spurious and stale uploaded files are deleted automatically after 24 hours.

Your designated upload email addresses (see Section 12.6.1 [Automated Upload Registration], page 24) are sent a message if there are problems processing an upload for your package. You also receive a message when an upload has been successfully processed.

One programmatic way to create and transfer the necessary files is to use the `gnupload` script, which is available from the `build-aux/` directory of the `gnulib` project at `https://savannah.gnu.org/projects/gnulib`. Run `gnupload --help` for a description and examples. (With `gnupload`, you specify a destination such as '`ftp.gnu.org:`'*pkgname* rather than using the '`ftp-upload`' hostname.)

`gnupload` invokes the program `ncftpput` to do the actual transfers; if you don't happen to have the `ncftp` package installed, the `ncftpput-ftp` script in the `build-aux/` directory of `gnulib` can serve as a replacement. It uses the plain command line `ftp` program.

If you have difficulties with an upload, email `ftp-upload@gnu.org`. You can check the archive of uploads processed at `https://lists.gnu.org/archive/html/ftp-upload-report`.

12.6.3 FTP Upload Release File Triplet

Ordinarily, the goal is to upload a new release of your package, let's say, the source archive `foo-1.0.tar.gz`. To do this, you simultaneously upload three files:

1. The file to be distributed; in our example, `foo-1.0.tar.gz`.
2. Detached GPG binary signature file for (1); for example, `foo-1.0.tar.gz.sig`. Make this with '`gpg -b foo-1.0.tar.gz`'.
3. A clearsigned *directive file*; for example, `foo-1.0.tar.gz.directive.asc`, created with '`gpg --clearsign foo-1.0.tar.gz.directive`'. Its contents are described in the next section.

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.*

12.6.4 FTP Upload Directive File

To repeat, a (signed) *directive file* must be part of every upload. The unsigned original is just a plain text file you can create with any text editor. Its name must be, e.g., `foo-1.0.tar.gz.directive` for accompanying an upload of `foo-1.0.tar.gz`.

After creating the file, run '`gpg --clearsign foo-1.0.tar.gz.directive`', which will create `foo-1.0.tar.gz.directive.asc`; this is the file to be uploaded.

When part of a triplet for uploading a release file, the directive file must always contain the directives `version`, `filename` and `directory`. In addition, a `comment` directive is optional. These directives can be given in any order.

Continuing our example of uploading `foo-1.0.tar.gz` for a package named `foo` to `ftp.gnu.org`, the values would be as follows:

`version` must be the value '`1.2`' (the current version, as of May 2012):
`version: 1.2`

`filename` must be the name of the file to be distributed:
`filename: foo-1.0.tar.gz`

`directory`
specifies the final destination directory where the uploaded file and its `.sig` companion are to be placed. Here we will put our file in the top level directory of the package, as is the most common practice:
`directory: foo`

`comment` is optional, and ignored if present:
`comment: let's hope this works!`

Putting all of the above together, the complete contents of the directive file `foo-1.0.tar.gz.directive` for our example would be:

```
version: 1.2
directory: foo
filename: foo-1.0.tar.gz
comment: let's hope this works!
```

Then you '`gpg --clearsign`' the file as given above, and upload (using anonymous ftp) the three files:

```
foo-1.0.tar.gz
foo-1.0.tar.gz.sig
foo-1.0.tar.gz.directive.asc
```

to the host `ftp-upload.gnu.org`, directory `/incoming/ftp` (for official releases), or the directory `/incoming/alpha` (for test releases).

After the system authenticates the signatures, the files `foo-1.0.tar.gz` and `foo-1.0.tar.gz.sig` are placed in the directory `gnu/foo/` on `ftp.gnu.org`. That is, we'll have made our release available at '`https://ftp.gnu.org/gnu/foo/foo-1.0.tar.gz`' (and then from our many mirrors via '`https://ftpmirror.gnu.org/foo/foo-1.0.tar.gz`'). Whew.

A common reason for the upload not succeeding is your GPG signature not being registered with the upload system. There is nothing that makes this happen automatically. You

must email the system administrators as described above (see Section 12.6.1 [Automated Upload Registration], page 24).

12.6.5 FTP Upload Directory Trees

You can make any directory hierarchy you like under your package directory. The system automatically creates any intermediate directories you specify in the `directory` directive.

Slightly modifying the example above, the following directive file:

```
version: 1.2
directory: foo/foo-1.0
filename: foo-1.0.tar.gz
comment: creates per-version subdirectory as needed
```

would put the tar file in the `foo-1.0/` subdirectory of the package `foo`, thus ending up at '`ftp.gnu.org:gnu/foo/foo-1.0/foo-1.0.tar.gz`'.

However, to keep things simpler for users, we recommend not using subdirectories, unless perhaps each release of your package consists of many separate files.

12.6.6 FTP Upload File Replacement

You can replace existing files that have already been uploaded by including a directive line `replace: true`. For example, you might like to provide a README file in the release directory and update it from time to time. The full directive file for that would look like this:

```
replace: true
version: 1.2
directory: foo
filename: README
comment: replaces an existing README
```

It is ok if the file to be replaced doesn't already exist; then the new file is simply added, i.e., the `replace` directive has no effect.

When an existing file is replaced, the original is archived to a private location. There is no automated or public access to such archived files; if you want to retrieve or view them, please email `sysadmin@fsf.org`.

We very strongly discourage replacing an actual software release file, such as `foo-1.0.tar.gz`. Releases should be unique, and forever. If you need to make fixes, make another release. If you have an exigent reason for a particular release file to no longer be available, it can be explicitly archived, as described in the next section.

If you want to make the current release available under a generic name, such as `foo-latest.tar.gz`, that is better done with symlinks, also as described in the next section.

12.6.7 FTP Upload Standalone Directives

The previous sections describe how to upload a file to be publicly released. It's also possible to upload a directive file by itself to perform a few operations on the upload directory. The supported directives are:

`symlink` creates a symlink.

`rmsymlink`
: removes a symlink.

`archive`
: takes a file or directory offline.

As for the directives described above, the `directory` and `version` directives are still required, the `comment` directive remains optional, and the `filename` directive is not allowed.

The `.sig` file should not be explicitly mentioned in a directive. When you specify a directive to operate on a file, its corresponding `.sig` file will be handled automatically.

When uploaded by itself, the name of the directive file is not important. But it must be still be signed, using '`gpg --clearsign`'; the resulting `.asc` file is what should be uploaded.

Here's an example of the full directive file to create a `foo-latest.tar.gz` symlink:

```
version: 1.2
directory: foo
symlink: foo-1.1.tar.gz foo-latest.tar.gz
comment: create a symlink
```

If you include more than one directive in a standalone upload, the directives are executed in the sequence they are specified in. If a directive results in an error, further execution of the upload is aborted.

Removing a symbolic link (with `rmsymlink`) which does not exist results in an error. On the other hand, attempting to create a symbolic link that already exists (with `symlink`) is not an error. In this case `symlink` behaves like the command `ln -s -f`: any existing symlink is removed before creating the link. (But an existing regular file or directory is not replaced.)

Here's an example of removing a symlink, e.g., if you decide not to maintain a `foo-latest` link any more:

```
version: 1.2
directory: foo
rmsymlink: foo-latest.tar.gz
comment: remove a symlink
```

And here's an example of archiving a file, e.g., an unintended upload:

```
version: 1.2
directory: foo
archive: foo-1.1x.tar.gz
comment: archive an old file; it will not be
comment: publicly available any more.
```

The `archive` directive causes the specified items to become inaccessible. This should only be used when it is actively bad for them to be available, e.g., you uploaded something by mistake.

If all you want to do is reduce how much stuff is in your release directory, an alternative is to email `sysadmin@fsf.org` and ask them to move old items to the `https://ftp.gnu.org/old-gnu/` directory; then they will still be available. In general, however, we recommend leaving all official releases in the main release directory.

12.6.8 FTP Upload Directive File - v1.1

The v1.1 protocol for uploads lacked the `replace` directive; instead, file replacements were done automatically and silently (clearly undesirable). This is the only difference between v1.2 and v1.1.

12.6.9 FTP Upload Directive File - v1.0

Support for v1.0 uploads was discontinued in May 2012; please upgrade to v1.2.

In v1.0, the directive file contained 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-1.0.tar.gz.directive.asc` file might contain the single line:

```
directory: bar/v1
```

This directory line indicates that `foo-1.0.tar.gz` and `foo-1.0.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-1.0.tar.gz` and `foo-1.0.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`).

12.7 Announcing Releases

When you have a new release, please make an announcement. For official new releases, including those made just to fix bugs, we strongly 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. On the other hand, please do not announce test releases on `info-gnu` unless it's a highly unusual situation.

Please also post release announcements in the news section of your Savannah project site. Here, it is fine to also write news entries for test releases and any other newsworthy events. The news feeds from all GNU projects at savannah are aggregated at `https://planet.gnu.org` (GNU Planet), unless the text of the entry contains the string '`::noplanet::`'. You can also post items directly, or arrange for feeds from other locations; see information on the GNU Planet web page.

You can maintain your own mailing list (typically '`info-`*package*`@gnu.org`') for announcements as well if you like. For your own list, of course you decide as you see fit what events are worth announcing. (See Chapter 10 [Mail], page 19, for setting this up, and more suggestions on handling mail for your package.)

When writing an announcement, please include the following:

- A very brief description (a few sentences at most) of the general purpose of your package.
- Your package's web page (normally '`https://www.gnu.org/software/`*package*`/`').
- Your package's download location (normally '`https://ftp.gnu.org/gnu/`*package*`/`'). It is also useful to mention the mirror list at `https://www.gnu.org/order/ftp.html`, and that '`https://ftpmirror.gnu.org/`*package*`/`' will automatically redirect to a nearby mirror.

- The `NEWS` (see Section "NEWS File" in *GNU Coding Standards*) for the present release.

You may find the `announce-gen` script useful for creating announcements, which is available from the `build-aux/` directory of the `gnulib` project at `https://savannah.gnu.org/projects/gnulib`.

13 Web Pages

When we dub a program a GNU package, we list its GNU home page, named *package* in '`https://www.gnu.org/software/`', on `https://www.gnu.org/software/software.html` and `https://www.gnu.org/manual/manual.html`. To avoid broken links, the webmasters create a temporary home page as follows:

- If there is a Savannah project for this package (see Section 13.1 [Hosting for Web Pages], page 30), the temporary home page redirects to the project's main page, '`https://savannah.gnu.org/projects/`*package*', where a short description should be available.
- Otherwise, the webmasters make a simple home page containing the short description provided with the original submission of the package to GNU.

This temporary home page ought to be replaced with the real one as soon as possible.

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 archives 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.

Your web pages should follow our usual standards (see `https://www.gnu.org/server/fsf-html-style-sheet.html`). The overall goals are to support a wide variety of browsers, to focus on information rather than visual adornments, and to keep gnu.org/software/ consistent on certain important points.

We encourage you to use the standard `www.gnu.org` template as the basis for your pages: `https://www.gnu.org/server/standards/boilerplate-source.html`. This template changes slightly from time to time for various reasons. If a change affects existing web pages, the webmasters will inform you; then you can make the change or they can.

Please follow the best practices of accessibility in your web pages (see `https://www.gnu.org/accessibility/accessibility.html`).

13.1 Hosting for Web Pages

The best 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, using the separate "web pages repository" available on Savannah, which corresponds to '`https://www.gnu.org/software/`*package*`/`'. You can keep your source files there too (using any of a variety of version control systems), but you can use `savannah.gnu.org` only for your gnu.org web pages if you wish; 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 Savannah if you can.

Please note that the GNU webmasters may fix technical details in your web pages (HTML, CSS, obvious typos, broken links in the footer, etc.) and inform you of the change afterwards.

If you use Savannah, you can use a special file named `.symlinks` in order to create symbolic links, which are not supported in CVS. For details, see `https://www.gnu.org/server/standards/README.webmastering.html#symlinks`.

13.2 Freedom for Web Pages

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 nonfree software, the public will see this, and it will have the effect of granting legitimacy to the nonfree 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.

Please make sure it is possible to use the web site fully using the Lynx browser, and with the IceCat browser with LibreJS enabled. It should work both with Tor and without Tor. Of course, it is desirable for the site to support as many other browsers as possible.

Historically, web pages for GNU packages did not include GIF images, because of patent problems (see Chapter 14 [Ethical and Philosophical Consideration], page 33). Although the GIF patents expired in 2006, using GIF images is still not recommended, as the PNG and JPEG formats are generally superior. See `https://www.gnu.org/philosophy/gif.html`.

Please make sure that any Javascript code in your web pages is covered by a free license, and has its license indicated in a way LibreJS can recognize. See `https://gnu.org/philosophy/javascript-trap.html`. If the Javascript in the page is minified, or for any other reason is not the source code, it must point to its source code as described there.

13.3 Manuals on Web Pages

The web pages for the package should include its manuals, in HTML, DVI, Info, PDF, plain ASCII, and the source Texinfo. All of these can be generated automatically from Texinfo using Makeinfo and other programs. If the Texinfo itself is generated from some other source format, include that too.

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 list all GNU manuals on the page `https://www.gnu.org/manual`, so if yours isn't there, please send mail to `webmasters@gnu.org`, asking them to add yours, and they will do so based on the contents of your `manual` directory.

13.3.1 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 Gnulib development:

`https://git.savannah.gnu.org/cgit/gnulib.git/tree/build-aux/gendocs.sh`
`https://git.savannah.gnu.org/cgit/gnulib.git/tree/doc/gendocs_template`

There is also a minimalistic template, available from:

`https://git.savannah.gnu.org/cgit/gnulib.git/tree/doc/gendocs_template_min`

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

```
gendocs.sh --email yourbuglist yourmanual "GNU yourmanual manual"
```

where *yourmanual* is the short name for your package and *yourbuglist* is the email address for bug reports (which should be `bug-`*package*`@gnu.org`). The script processes the file *yourmanual*`.texinfo` (or `.texi` or `.txi`). For example:

```
cd .../texinfo/doc
# download gendocs.sh and gendocs_template
gendocs.sh --email bug-texinfo@gnu.org texinfo "GNU Texinfo 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 --email bug-texinfo@gnu.org -o texinfo texinfo "GNU Texinfo manual"
gendocs.sh --email bug-texinfo@gnu.org -o info info "GNU Info manual"
gendocs.sh --email bug-texinfo@gnu.org -o info-stnd info-stnd "GNU info-stnd 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`, etc., to control the programs that get executed, and `GENDOCS_TEMPLATE_DIR` to control where the `gendocs_template` file is found.

As usual, run '`gendocs.sh --help`' for a description of all the options, environment variables, and more information.

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

13.4 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 the "Keyword Substitution" section in the CVS manual, available from `https://cvs.nongnu.org`.

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.

14 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 `licensing@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. That is why we rejected MP3 audio format in favor of the unpatented Ogg Vorbis format. These patents have reportedly expired, but we still prefer Ogg formats to MP3 formats. Please support this campaign by making Ogg Vorbis the preferred format for audio distribution in GNU packages and their web sites.

We will consider using Ogg Opus at some point in the future. It is fine to distribute Ogg Opus files *also*, but please continue distributing Ogg Vorbis, so as not to hurry users to change the software with which they listen to audio.

We are unable to find a modern compressed video format that is truly safe from patents, so we use the Ogg Theora and WebM formats for which no licensing consortium has been set up. GNU programs and their web sites should not distribute video in MPEG-2 or MPEG 4 formats.

A GNU package should not recommend use of any nonfree program, nor should it require a nonfree program (such as a nonfree 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 nonfree software to build or function. The GNU Coding Standards say a lot more about this issue.

Similarly, a GNU package should not require the use of nonfree software, including JavaScript, for the coordination of its development. For example, please don't use Transifex for translation of your software because it requires your translators to use nonfree, JavaScript-based editing tools. Instead, a service without any ethical concerns should be used, such as The Translation Project (`https://translationproject.org`).

A GNU package should not refer the user to any nonfree 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.

Please don't host discussions about your package in a service that requires nonfree software. For instance, Google+ "communities" require running a nonfree JavaScript program to post a message, so they can't be used in the Free World. Google Groups has the same problem. To host discussions there would be excluding people who live by free software principles.

Of course, you can't order people not to use such services to talk with each other. What you can do is not legitimize them, and use your influence to lead people away from them. For instance, where you say where to have discussions related to the program, don't list such a place.

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.

15 Humor and GNU

In GNU, we appreciate humor in our work.

GNU is a project of hackers, and hacking means playful cleverness. Even the name "GNU" is an example of playful cleverness—it is a recursive acronym for "GNU's Not Unix."

In that spirit, we prize occasional doses of humor in GNU packages. Humor is not mandatory in a GNU package; we do not tell maintainers, "Make users smile, or else!" But when maintainers do that, we too smile.

Nowadays, our humor-positive approach occasionally encounters direct, blanket opposition. Some people advocate, and even demand, removal of jokes from software packages simply because they are jokes. We shrug off that point of view.

Jokes are subject to the same sorts of issues and criticism as other writing. Sometimes there is a valid objection to text which is humorous, so we do not defend every joke obtusely to the bitter end. But *the fact that it is a joke* is not a valid objection.

There are people who frown on anything that is slightly risqué or controversial, including jokes. It would be a terrible shame for that attitude to prevail, so our policy is that the occasional risqué joke is ok. GNU is a 21st century project, not a 19th.

16 Other Politics

The GNU Project supports the cause of software freedom, that the users of computing should have control of their computing activities. This requires that they have control of their software that does those activities, which in turn requires that they do these activities with free software and have the possibility of replacing any shared copies with their own copies.

It also supports basic human rights in computing including use of the internet; opposing censorship, for instance.

A GNU package should not seriously advocate any other political causes. Not that the GNU Project opposes those other causes. Rather, it is neutral on them, and GNU packages should be neutral too. For example, if you are (say) a pacifist, you must not advocate pacifism in the GNU package you develop. Contrariwise, if you want to launch a war, the GNU package you develop shouldn't advocate that either.

17 Terminology Issues

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

17.1 Free Software and Open Source

The terms "free software" and "open source", while describing almost the same category of software, stand for views based on fundamentally different values. The free software movement is idealistic, and raises issues of freedom, ethics, principle and what makes for a good society. The term open source, initiated in 1998, is associated with a philosophy which studiously avoids such questions. For a detailed explanation, see `https://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 term "open source" receives, the GNU Project needs to overcome a widespread mistaken impression that GNU is *and always was* an "open source" activity. For this reason, please use the term "free software", not "open source" or "FOSS", 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.

17.2 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 `https://www.gnu.org/gnu/linux-and-gnu.html`.

To make it clear that Linux is a kernel, not an operating system, please take care to avoid using the term "Linux system" in those materials. If you want to have occasion to make a statement about systems in which the kernel is Linux, write "systems in which the kernel is Linux" or "systems with Linux as the kernel." That explicitly contrasts the system and the kernel, and will help readers understand the difference between the two. Please avoid simplified forms such as "Linux-based systems" because those fail to highlight the difference between the kernel and the system, and could encourage readers to overlook the distinction.

To contrast the GNU system proper with 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.

For more about this point, see `https://www.gnu.org/gnu/gnu-linux-faq.html`.

18 Interviews and Speeches

Interviews and speeches about your package are an important channel for informing the public about the GNU system and the ideas of the free software movement. Please avoid saying "open source" and avoid calling the GNU system "Linux", just as you would in the package itself (see Chapter 17 [Terminology], page 35). Likewise, avoid promoting nonfree programs (see Section "References" in *GNU Coding Standards*) as you would in the package itself.

Many GNU users have erroneous ideas about GNU. Outside of our community, most people think it is Linux. Please use your opportunity to set them straight. Start the presentation with the answers to these basic questions:

- What GNU is (an operating system developed to be Unix-like and totally free software). It is good to mention `https://www.gnu.org`.
- What free software is (the users control it, so it doesn't control them). It is good to state the four freedoms and/or refer to `https://www.gnu.org/philosophy/free-sw.html`.
- What GNU/Linux is (Linux filled the last gap in GNU). It is useful to refer to `https://www.gnu.org/gnu/linux-and-gnu.html`.

- What the GNU Project is (the project to develop GNU).
- How your package fits in (it's part of GNU, and the work is part of the GNU Project).

If you feel a social pressure not to say these things, you may be coming in contact with some who would prefer that these things not be said. That's precisely when we need your support most.

Please don't include advertisements or plugs for any company, product or service. Even if the product would meet the standards for the FSF to endorse it, an ad for it is out of place in a presentation about a GNU package. Likewise, please don't include company slogans. Mention a company only when called for by the subject matter.

A few GNU packages are actually business activities of a particular company. In that case, it is ok to say so at the start. Otherwise, please show that this is a project of the GNU Project, and avoid suggesting it is any company's project.

If you are paid by a company to work on the GNU package, it is appropriate to thank the company in a discreet way, but please don't go beyond that.

Before you do a speech or interview, please contact the GNU Project leadership. We can give you advice on how to deal with various eventualities.

When your interviews and speech recordings or transcript are posted, please tell us about them. Then we can publicize them.

Please post them in formats that are friendly to free software: not in Doc or Docx format, not with Flash, not with QuickTime, not with MP3, MPEG2 or MPEG4. Plain text, HTML and PDF are good.

19 Hosting

We recommend using `savannah.gnu.org` for the source code repository for your package, but that's not required. See Chapter 11 [Old Versions], page 21, for more information about Savannah.

We strongly urge you to use `ftp.gnu.org` as the standard distribution site for releases. Doing so makes it easier for developers and users to find the latest GNU releases. However, it is ok to use another server if you wish, provided it allows access from the general public without limitation (for instance, without excluding any country).

If you use a company's machine to hold the repository for your program, or as its release distribution 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 https://www.gnu.org/philosophy/free-sw.html.

If you are looking for service or support for GNU software, see
https://www.gnu.org/gethelp/ 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.
```

20 Donations

As a maintainer, you might want to accept donations for your work, especially if you pay for any of your own hosting/development infrastructure. Following is some text you can adapt to your own situation, and use on your package's web site, `README`, or in wherever way you find it useful:

```
We appreciate contributions of any size -- donations enable us to spend
more time working on the project, and help cover our infrastructure
expenses.

If you'd like to make a small donation, please visit url1 and do
it through payment-service.  Since our project isn't a
tax-exempt organization, we can't offer you a tax deduction, but for
all donations over amount1, we'd be happy to recognize your
contribution on url2.

We are also happy to consider making particular improvements or
changes, or giving specific technical assistance, in return for a
substantial donation over amount2.  If you would like to discuss
this possibility, write to us at address.

Another possibility is to pay a software maintenance fee.  Again,
write to us about this at address to discuss how much you want
to pay and how much maintenance we can offer in return.  If you pay
more than amount1, we can give you a document for your records.

Thanks for your support!
```

We don't recommend any specific payment service. However, GNU developers should not use a service that requires them to sign a proprietary software license, such as Google's payment service. Please also avoid sites that requires users to run nonfree software in order to donate. (This includes JavaScript software, so try it with LibreJS or with JavaScript disabled.)

In the text you post on the site, please pay attention to the terminological issues we care about (see Chapter 17 [Terminology], page 35).

We have no objections to using Bitcoin to receive donations.

The FSF can collect donations for a limited number of projects; if you want to propose that for your project, write to `maintainers@gnu.org`. The FSF is required to supervise the spending of these funds.

Of course, it is also good to encourage people to join the FSF (`https://www.fsf.org`) or make a general donation, either instead of or as well as package-specific donations.

21 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 see `https://www.gnu.org/help/directory.html#adding-entries` for information on how to write an entry

for your package. Contact `bug-directory@gnu.org` with any questions or suggestions for the Free Software Directory.

22 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 Suggested Responses

Here are some responses you can use when appropriate, if you want to.

Here's a way to say no to installing code to make your package work on a proprietary operating system, ShackleOS.

> You've asked us to install support for doing XYZ on ShackleOS. We can't do that until we have support for XYZ on the GNU system. GNU Project policy is not to add special support for a nonfree operating system until we have equivalent support for the GNU system.
>
> A nonfree system subjugates users. You may not notice this if you have become accustomed to such subjugation, but we do. The Free Software Movement aims

> to liberate those users by replacing nonfree systems with free software such as the GNU system.
>
> This program does not aim to replace ShackleOS, but the GNU system does. We must support the effort to supplant ShackleOS, not weaken it. If we were to implement more or better support for ShackleOS than for GNU, we would score an own goal.
>
> So please make this feature work on GNU, and then we can install it.

Here's a way to say no to installing code to make your package work with a proprietary program, ShackleMe.

> You've asked us to install a feature specifically to work with ShackleMe, but that program is nonfree. GNU Project policy is not to add special support for interoperation with a nonfree program until we support interoperation with comparable free programs equally well or better.
>
> A nonfree program subjugates users. You may not notice this if you have become accustomed to such subjugation, but we do. The mission of the GNU Project is to liberate those users by replacing the nonfree programs with free programs.
>
> This program does not aim to replace ShackleMe, but other free programs do or should. We must support their effort to supplant ShackleMe. If we were to implement interoperability with ShackleMe more than with them, this program would become an additional obstacle to their success. We would score an own goal.
>
> So please make this feature work well with those free replacements first. Once we support them, we can support ShackleMe too.

Appendix B GNU Free Documentation License

Version 1.3, 3 November 2008

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

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

0. PREAMBLE

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

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

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

1. APPLICABILITY AND DEFINITIONS

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

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

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

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

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

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

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

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

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

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

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

2. VERBATIM COPYING

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

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

3. COPYING IN QUANTITY

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

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

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

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

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

4. MODIFICATIONS

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

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

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

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

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

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

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

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

O. Preserve any Warranty Disclaimers.

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

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

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

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

5. COMBINING DOCUMENTS

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

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

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

6. COLLECTIONS OF DOCUMENTS

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

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

7. AGGREGATION WITH INDEPENDENT WORKS

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

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

8. TRANSLATION

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

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

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

9. TERMINATION

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

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

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

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

10. FUTURE REVISIONS OF THIS LICENSE

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

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

11. RELICENSING

    "Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of

such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

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

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

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

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

ADDENDUM: How to use this License for your documents

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

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

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

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

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

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

Index

$

$ keywords in web pages 33

/

/gd/gnuorg directory 3

A

advisory committee 1
alpha.gnu.org, test release site 23
announcement mailing list, project-specific 29
announcements 29
announcements, mailing list for 19
assignments, copyright 3
AUTHORS file 7
automake 22

B

beta releases 23
bug reports, email tracker for 20
bug reports, handling 20
bug reports, web tracker for 20
bug-gnu-utils@gnu.org 19
bug-standards@gnu.org email address 1

C

contents of announcements 29
contributions, accepting 14
copyright notices in program files 8
copyright papers 3
creating mailing lists 19
crediting authors 14
CVS keywords in web pages 33
CVS repository 37

D

data base of GNU copyright assignments 3
development method, open source 35
development resources 2
diff 22
directive file, for FTP uploads 26
directives for ftp uploads, standalone 27
directory trees, in ftp uploads 27
Directory, Free Software 38
disclaimers 3
distribution, tar files 21
documentation output, generating 32
Donations, for packages 38
down, when GNU machines are 1

E

email ... 19
ethics ... 33

F

FDL, GNU Free Documentation License ... 40
`fencepost.gnu.org` GNU login host ... 2
formats for documentation, desired ... 31
Free Software Directory ... 38
free software movement ... 35
FSF system administrators ... 1
ftp uploads, automated ... 24
`ftp.gnu.org`, the GNU release site ... 23
FTP site ... 37
FTP uploads, of release files ... 25

G

`gendocs.sh` ... 32
generating documentation output ... 32
GNU ftp site ... 23
GNU system administrators ... 1
GNU/Linux ... 35
`gnustandards` project repository ... 1
`gnustandards-commit@gnu.org` mailing list ... 1

H

help for users, mailing list for ... 19
help requests, handling ... 20
help, getting ... 1
hierarchy, under ftp upload directory ... 27
hosting ... 37
`https://bugs.gnu.org` ... 20
`https://hostux.social/@fsfstatus` ... 1
`https://planet.gnu.org` ... 29
Hydra ... 2

I

`info-gnu` mailing list ... 29

L

legal matters ... 3
legal papers for changes in manuals ... 5
license notices in program files ... 10
Linux ... 35

M

mailing list for bug reports ... 19
mailing lists, creating ... 19
mailing lists, standard names of ... 19
`maintainers@gnu.org` ... 2
`mentors@gnu.org` mailing list ... 1
Money, donated to packages ... 38
movement, free software ... 35

O

open source ... 35
outage, of GNU machines ... 1

P

`patch` ... 22
patches, against previous releases ... 22
philosophy ... 33
Piercy, Marge ... 1
`platform-testers` mailing list ... 2
pretest releases ... 23
proofreading ... 39

Q

quality of changes suggested by others ... 14

R

RCS keywords in web pages ... 33
recording contributors ... 7
registration for uploads ... 24
release site ... 37
replacing uploaded files ... 27
repository ... 37
resigning as maintainer ... 2
resources for GNU developers ... 2
responding to bug reports ... 20

S

Savannah repository for `gnustandards` ... 1
Savannah, news area ... 29
`savannah-announce@gnu.org` mailing list ... 21
`savannah-hackers@gnu.org` ... 21
shell account, on fencepost ... 2
source repository ... 37
spam prevention ... 20
standalone directives, for ftp uploads ... 27
standard mailing lists ... 19
stepping down as maintainer ... 2
sysadmin, FSF ... 1

T

terminology 35
test releases 23
time stamp in diffs 22

U

uploads 25
uploads, directory trees in 27
uploads, registration for 24
uploads, replacing 27

V

version control 21
version control system 37

W

web pages 30
web pages, and CVS keywords 33
web pages, freedom for 31
web pages, hosting for 30
web pages, including manuals on 31