# GNUnited Nations

**by Yavor Doganov <yavor@gnu.org>**

This manual (updated 27 April 2024) is for GNUnited Nations (version 1.5), a suite for maintaining translations of www.gnu.org essays and other articles.

# Table of Contents

# 1 Introduction to GNUnited Nations

GNUnited Nations (abbreviated GNUN) is a collection of scripts that are supposed to make the life of `https://www.gnu.org` translators easier. Although it is specifically developed for the GNU Project's website, it could be customized, at least in theory, to fit the needs of other internationalized sites. GNUN is in an early stage of development, but if it proves useful, and if there is sufficient interest (and time), it is possible to develop a robust configuration interface that would be appropriate for general usage.

It is vitally important to understand that GNUN is *not* a silver bullet that solves all problems. To be honest, deploying GNUN in fact even does create some Section 1.4 [Disadvantages], page 4.

GNUnited Nations is free software, available under the GNU General Public License.

This manual is organized in a way that is suitable both for translators and GNU Web Translation Managers (plus eventually interested GNU Webmasters, if any). It may also serve as an introductory material and reference for new GNUN developers and contributors. Hopefully, it might be useful to people who customize and adopt the software for a third party site or for their own needs. Feel free to skip sections or entire chapters if they are irrelevant for your intended usage.

This manual is free documentation, and you can modify and redistribute it under the terms of the GNU Free Documentation License. See Appendix A [GNU Free Documentation License], page 37.

## 1.1 Why GNUN Is Being Developed

The GNU Project's website (`https://www.gnu.org`), has become considerably large over the years. Maintaining it requires significant effort, and sometimes a new web standard is developed faster than the time required to migrate all articles to the next widely adopted one.

When it comes to internationalization, the problems are so many that it is hard to enumerate them. It has become apparent that maintaining translations up-to-date is a major undertaking, involving tedious skimming through commit logs, reviewing diffs and other medieval techniques to catch up. Some translation teams have developed their own sets of scripts, but so far there has been no universal solution.

This unpleasant situation, combined with rapid and incompatible design changes, have led some teams to neglect the important work of keeping their translations in line with the changing original articles. As a consequence, the GNU Project is facing the problem of maintaining them in suboptimal ways, in order to keep the information updated.

The reasons for developing GNUnited Nations are very similar to those that led to the inception of GNU gettext, or GNOME Documentation Utilities (`gnome-doc-utils`) some years later.

## 1.2 What GNUnited Nations Is and Should Be

The basic concept behind GNUN is that localization of HTML articles is similar to localization of computer programs[1]. In articles, like in programs, not every string is considered translatable, so translatable strings must be identified first, and then collected in a file (called "PO template") for translation. Articles, like programs, tend to change over time, but not every change in the sources calls for a translation update. Sometimes the change does not affect the translatable strings, but sometimes it does. So, translators must have means to identify those changes and apply the appropriate updates to the translation.

---

[1] Actually, it is a lot closer to localization of software documentation, where typically strings (also known as "messages" in gettext's context) are longer than strings in programs. Nevertheless, all points raised still apply.

The GNU `gettext` package already provides the needed infrastructure for maintaining translations using PO files. See Section "Introduction" in *GNU gettext tools*, for a basic overview. GNUnited Nations fills the gaps to apply this infrastructure to articles in the `https://www.gnu.org` web site.[2]

The following diagram summarizes the relation between the files handled by GNUN. It is followed by somewhat detailed explanations, which you should read while keeping an eye on the diagram. Having a clear understanding of these interrelations will surely help translators and web maintainers.

```
    .---<--- * Original ARTICLE.html
    |
    |   .---> ARTICLE.pot ---> * ARTICLE.LANG.po --->---.
    '---+                                               |
        '--->---.    .------<---------------------------'
                |    |
                |    '---.
                |        +---> Translated ARTICLE.LANG.html
                '-------'
```

The indication '`*`' appears in two places in this picture, and means that the corresponding file is intended to be edited by humans. The author or web maintainer edits the original `article.html`, and translators edit `article.lang.po`. All other files are regenerated by GNUN and any manual changes on them will be lost on the next run.

Arrows denote dependency relation between files, where a change in one file will affect the other. Those automatic changes will be applied by running '`make -C server/gnun`'. This is the primary way to invoke GNUN, since it is implemented as a set of recipes for GNU `make`.

First, GNUN extracts all translatable strings from the original English article `article.html` into `article.pot`[3]. The resulting file is suitable for manipulation with the various GNU '`gettext`' utilities. It contains all original article strings and all translations are set to empty. The letter `t` in `.pot` marks this as a Template PO file, not yet oriented towards any particular language.

On the first run, though, there is no `article.lang.po` yet, so a translator must create `article.lang.po` from `article.pot`, where *lang* represents the target language. See Section 4.1 [New Translation], page 18, for details.

Then comes the initial translation of messages in `article.lang.po`. Translation in itself is a whole matter, whose complexity far overwhelms the level of this manual. Nevertheless, a few hints are given in another chapter of this manual.

It is possible to make GNUN get translations for common strings from dedicated PO files, the so-called compendia. See Section 2.6 [Compendia], page 11, for more information.

You may use any compatible PO editor to add translated messages into the PO file. See Section "Editing" in *GNU gettext tools*, for more information.

When the PO file actually exists (hopefully populated with initial translations), GNUN generates `article.lang.html` file. It takes its structure from the original `article.html`, but all translatable strings are replaced with the translations specified in `article.lang.po`.

Original articles sometimes change. A new paragraph is being added or a tiny change in the wording is introduced. Also, some articles are dynamic in nature, like the ones containing news entries or a list of other articles. If the original article changes, GNUN will automatically rebuild

---

[2] The process of converting HTML to PO and the other way around is performed using po4a ("PO for anything"), see PO4A website (`https://po4a.org`).

[3] A special group of templates, so called optional templates, have names like `article.pot.opt`. The PO file for them is still `article.lang.po`. See [optional-templates], page 10.

`article`.pot, and will merge the changes into `article.lang`.po. Any outdated translations will be marked as "fuzzy," any new strings will be added with empty translations, waiting to be translated. In the same run `article.lang`.html will be rebuilt, and consequently the relevant strings in the translation will be substituted with the original English text until the translation teams update them in `article.lang`.po.

Those changes in the original article that do not affect the translatable strings (or just delete whole strings) will not lead to new untranslated or "fuzzy" strings in `article.lang`.po. Thus, no actions from translators will be needed. `article.lang`.html will be automatically regenerated to reflect the changes.

The POT for every article under GNUN's control is kept in the 'www' repository under a special sub-directory `po/` of the relevant directory in the 'www' tree. Thus, for 'https://www.gnu.org/philosophy/free-sw.html' this sub-directory is `philosophy/po/`. In addition to `free-sw.pot`, this directory holds the canonical source of every translation, such as `free-sw.bg.po`, `free-sw.ca.po`, etc. For more details, see Section 6.1 [Files and Directories], page 27.

Several additional features are implemented, like automatic update of the list of the available translations. For example, if a new `free-sw.ja.po` translation is added, the list of translations included in `free-sw.html` and all translated `free-sw.lang`.html is updated. This saves a lot of tedious, repetitive work and eliminates a source of mistakes. There is a basic infrastructure to "inject" general information about a translation team—like a note on how to contact the team, or how to report a bug/suggestion for improvement. Translators' credits are also handled, as well as translators' notes, if any.

## 1.3 Major Advantages of GNUN

Here is a simple list of situations where we hope this suite would prove useful.

- Automatic rebuild of all translations when the original article changes. This is the most important feature, as it prevents accumulation of seriously outdated translations.
- Global update of the whole site. Apply the previous point to the web server templates (under `server/` in the 'www' repository). A single change to such a file will affect literally *all* articles, translated or not.
- Urgent notices. Sometimes an "urgent" notice is added by the webmasters, which should appear on all pages. Typically this is about an event where urgent action is needed, although often it is only relevant to a single country or even a particular city. Such a notice will propagate to all pages, and translators may choose whether to translate it or not. For example, the Urdu translation team may conclude that there are only a few Urdu speakers in Massachusetts, to participate in an event that will happen in Boston, so translating the "urgent" notice may not be very "urgent" for Urdu. However, such notice will appear in all translated pages and people who usually read gnu.org pages in their native language will see it, so they can take action as necessary. When the notice is removed, often within a week or two, it will disappear without translators' intervention, whether they translated it or not.
- HTML validation. As a preliminary step, GNUN validates the English pages before updating the POT files, and the regenerated translations before committing them. It often detects typos and other errors in the markup.
- Simplification of the translation process—lots of errors and typos come from the fact that translators basically have to duplicate the whole HTML markup of the original. The PO files eliminate most of the basic markup, which is where most of the validation errors come from.
- Markup consistency site-wide—it will be substantially easier to update the site to a future standard, because translations will naturally follow the changes in the original articles.

This also means that translation teams do not have to go through the boring process of converting their articles to the new SSI-based layout; this will be done automatically.

- Easy updates by translators. Modified paragraphs, links, etc. will appear as "fuzzy" strings in the PO files, newly added ones will appear as "untranslated", and deleted will appear as "obsolete". It is substantially easier to update a PO file, where a keystroke takes you to the part that needs updating, whatever it may be.

- Reporting and statistics. Since the basis is standard PO files, which are the canonical source of the translations, it is easy to manipulate them and extract useful information.

## 1.4 Known Bugs and Limitations

As it happens in real life, we don't wear pink glasses and are aware of certain limitations and annoyances of this semi-automatic system.

- There is no easy way to preview the translations. The official build is invoked twice an hour, because doing it much more often is not feasible with current build server hardware. Additionally, any errors interrupt the build so they have to be fixed before the next articles are processed.

- In very rare cases the structure of the translation should diverge from the structure of the original document. So far, our system has developed no means of taking such issues into account (apart from alphabetically sorted lists).

# 2 General Usage

> If anything may go wrong, it will definitely go wrong.
> —Murphy's Law

> Murphy is an optimist.
> —O'Reilly's Law

GNUN currently consists of a few makefiles, scripts and template files. They are designed to reside in the `server/gnun` directory, but this may change. In all examples in this manual, "invoking" means executing on the command line `make -C server/gnun [target] [variable=value ...]` while the working directory is the root in the 'www' web repository. For the purpose of brevity, we will refer to the above command as simply `make`, which is equivalent to `cd server/gnun; make`. It is desirable never to invoke `make` with the `-k` (`--keep-going`) option, because an eventual error in only one make recipe might create a mess in many articles, both original and translated. Do this with caution, and generally only when debugging in a safe environment.

The build process is intended to be invoked by a cron job, although manual intervention is possible to a certain degree.

## 2.1 Invoking GNUN

The central part of GNUnited Nations is a makefile; actually a `GNUmakefile` since it heavily relies on features and extensions available in GNU Make. Thus, invoking a build consists of typing `make` on the command line, or within cron. If you are deploying the software on a non-GNU machine, probably GNU Make is installed and available as `gmake` or `gnumake`. If not, you should consider installing it, since the build will fail otherwise. See GNU Make homepage (`https://www.gnu.org/software/make`) for information on how to download and install GNU Make.

If you don't specify a target, `make` by default builds the target `all`, which in this case is to rebuild all translations that are not up-to-date. However, there are special targets that do not depend on the standard `all` target, which can be built by `make target`. Some of the variables in the next section apply to them, and some do not.

Note that GNUN expects `GNUmakefile`, `config.mk` and `gnun.mk` to be *present* under `server/gnun` of the 'www' web repository, otherwise `make` has no way to know what to build, and how. Another file, `priorities.mk`, is expected to be present under `server/gnun` in order to define the priorities when reporting about the outdated translations. If absent, the `report` target will not sort the translations by priority. Since the location of the repository working copy is strictly user-specific and cannot be determined in any way, `GNUmakefile` and `config.mk` must be copied there manually after the package installation. For convenience, these files are installed in '`$(pkgdatadir)`' (`/usr/local/share/gnun` with the default '`prefix`') so you can just create symlinks pointing to them, e.g.:

```
ln -s /usr/local/share/gnun/config.mk /path/to/www/server/gnun/
ln -s /usr/local/share/gnun/GNUmakefile /path/to/www/server/gnun/
```

If next GNUN releases are installed with the same `--prefix`, you will always use the latest versions without the need for any manual intervention.

If you are playing with a non-gnu.org setup, you also have to take care of `gnun.mk` and put a modified version under `server/gnun` of your hypothetical tree. (For gnu.org, that is not necessary since a proper `gnun.mk` is maintained in 'www'.)

## 2.2 Variables to Control the Build Process

The build process has several modes of operation, and they all relate to the handling of files that are to be added to the repository or performing certain sanity checks at build time. The variables are specified on the command line, after `make`, in the form `VARIABLE=value`, e.g. `make VCS=yes`. In the future, additional features will be implemented in a similar fashion.

‘`VCS=no`’

‘`...`’

Do not add any files to the repository. This is the default. You may as well omit to define `VCS` entirely; there is no special code that expects assigning the value ‘no’.

‘`VCS=yes`’    Automatically add any new files in the repository (CVS, Subversion or GNU Bazaar—the repository type is auto-determined at build time, `bzr` being a fallback).[1] These are any POT files, if they are generated for the first time, and the translated articles (`.lang.html`) in HTML format. Finally, any missing PO and their HTML counterparts of the server templates will be added, computed on the basis of the `extra-templates` and `optional-templates` variables.

‘`VALIDATE=no`’

‘`...`’       Skips validation of the HTML articles and generated translations.

‘`VALIDATE=yes`’

Validates all original articles before generating the POTs, to ensure that the ultimate source is valid (X)HMTL. Also, validates all generated translations in HTML format and all PO files. It is highly recommended to run the build this way, even if it is a bit tedious to fix the errors that are reported as a result of enforcing validation.

This is the default, and not defining this variable has the same effect.

‘`NOTIFY=no`’

‘`...`’       Do not send email notifications about errors. This is the default.

‘`NOTIFY=yes`’

If an error occurs, send a mail with a meaningful subject and the error message as body to the concerned party. The variables `devel-addr`, `web-addr` and `transl-addr` control the recipients; normally they should be set to the GNUN maintainers, webmasters and translators accordingly. These variables are defined in the configuration file `gnun.conf` and by default are set to `bug-gnun@gnu.org`.

‘`ANNOUNCE=yes`’

If defined, automatic announcements for new translations will be sent to the address defined in the `ann-addr` variable (in `gnun.conf`). The email messages contain the translated article title as Subject, and the URL of the translation as its body. For the official GNUN build, they are delivered to the `trans-coord-news@gnu.org` mailing list and each language has its own Mailman *topic*. It is possible to subscribe to the list and receive only traffic related to a specific language. See Section “Mailing Lists” in *GNU Web Translators Manual*.

The default behavior is not to send such announcements.

‘`VERBOSE=yes`’

If defined, the value of the variables `templates-translated`, `ALL_POTS`, and `articles-translated` will be printed to the standard output. This is off by default, but recommended in general since it will show a bug in the computation of the basic variables.

---

[1]  When GNU Bzr is used, files are added locally only; you need to take care to use `bzr push` manually (or via cron) to take care of effectively adding them to the public repository. See Section 2.3.4 [triggers], page 8, for a short explanation.

'GRACE'        If defined, ordinary articles that have "fuzzy" strings will not be regenerated. This
               functionality is implemented specifically to prevent gratuitous replacement of trans-
               lated strings with the English text when there are only minor formatting changes
               in the original. The translator should review the changes, update the translation
               and clear the "fuzzy" mark from the strings, while keeping the online translation
               intact[2]. Note that this variable has no effect on the server templates and all articles
               defined in the variable `no-grace-articles`.

'OUTDATED-GRACE=*days*'
               Grace period for the out-of-date notice. When the variable `GRACE` is defined,
               `OUTDATED-GRACE` defaults to 60 days. The out-of-date notice is a special text
               (`server/outdated.html` in the 'www' repository) that is inserted into every out-
               dated translation when the period defined in this variable is over; its purpose is to
               inform the reader that the translation may not correspond to the original English
               article.

'TEAM=*lang*'
               The translation team whose articles need to be checked for completeness. This
               variable is applicable only for the `report` target, and is mandatory for it. See
               Section 2.3.3 [report], page 8.

When validation is enabled, the original English articles are validated first, before any com-
mands that generate the other files, and `make` exits with an error on the first encountered article.
This is done on purpose, to prevent the propagation of an eventual error in the markup of the
original article to all translations.

Validation of the translated `.lang.html` is performed after it is preliminarily generated as
a temporary file. When no errors are found, the translation is updated; otherwise the real file
is not changed (and it is not added if absent)—the build will fail and further processing of the
remaining articles will not be performed. The translator has time until the next run to fix the
error—usually by modifying the corresponding `.lang.po` file.

If notification is enabled (`NOTIFY=yes`), and the build system encounters errors (mostly when
validating articles), email messages will be sent to the party that is expected to fix the error.
The subject of the messages always includes the problematic article, for example:

        Subject: [GNUN Error] gnu/gnu.fa.html is not valid HTML

## 2.3 Targets Specified on the Command Line

Some targets are not built by default, because they are only useful under certain circumstances.
Think of them like semi-automated commands or canned command sequences that are more
complicated, and more importantly, whose arguments are variables computed at the time `make`
reads the makefiles—the filesets they affect are specific and already defined, one way or another.

### 2.3.1 The `no-grace-items` Target

The `no-grace-items` target regenerates a limited set of articles that are not affected by the
`GRACE` variable, namely, the server templates, and all articles defined in the variable `no-grace-
articles` (see [grace flag], page 6).

This target can be rebuilt more often than `all`; however, currently we just build all with the
same period.

---

[2] This variable used to define the "grace period" in days to let the translator update the strings before the
    English text propagates to the translated page, but in practice that period has always been unlimited

### 2.3.2 The `update-localized-URLs` Target

The `update-localized-URLs` target invokes a script that generates the `localized-urls.mk` file. This file includes the list of URLs that should be localized, it is extracted from special comments in the HTML source (see Section 5.3 [Localized URLs], page 24).

This target is meant to be rebuilt nightly.

### 2.3.3 The `report` Target

This target exists solely for convenience to translators, enabling them to check which articles are not 100% translated and have to be updated. The way to check this is by running `make report TEAM=`*lang*, where *lang* is the language code, as usual. Thus, to check all French translations, one would run

```
make report TEAM=fr
```

This target checks only the PO files; the old translations that haven't been converted to PO files are reported, but there is no reasonable way to check if they are up-to-date. In fact, this is one of the main reasons GNUN is being developed, if you recall.

When present, `priorities.mk` defines four classes of articles by priority: *priority-articles* for the most important translations, *important-articles* for the second priority level, *important-directories* for the directories with important articles; all other translations are reported as less important.

There is also a script, `gnun-report`, to generate HTML reports. See Section 6.2.8 [gnun-report], page 33. GNUmakefile.team provides another `report` target, which produces a more detailed report. See [report in GNUmakefile.team], page 15.

### 2.3.4 The `triggers` Target

This is a special target intended to be run by the automatic build after the main build and *after* `cvs|svn|bzr commit`.

The `triggers` target currently executes the files named *article.lang*`.html.hook` in the `server/gnun` directory—these files are created during the main build and each of them contains the command to update the timestamp of the prerequisite based on the timestamp of the target that must be rebuilt. Finally, it deletes all those `*.hook` files.

This is the rule that takes care of actually sending the announcements if `ANNOUNCE=yes`. Since it is a completely valid scenario to have a new translation which is initially invalid HTML, the canned command sequence for announcements is recorded in `*.hook-ann` files, and they are treated by `triggers` in a different way. A newly added `.`*lang*`.po` file may be invalid, in which case the HTML file is not added, and it is not appropriate to announce it as a new translation as it is completely useless for the general public. The `triggers` rule detects this case, and postpones the announcement to the next build attempt, when the generated HTML translation is supposed to be human-readable.

To summarize, for effective operation GNUN should be invoked automatically as `make; cvs|svn|bzr commit -m ...; make triggers`. To illustrate this, here is a concrete example showing the official job once running at fencepost.gnu.org[3]:

```
25 4,12,20 * * *  cd $HOME/projects/www; cvs -q update &>/dev/null; \
                  make -j1 -C server/gnun VCS=yes VALIDATE=yes; \
                  NOTIFY=yes VERBOSE=yes GRACE=5000; ANNOUNCE=yes; \
                  cvs commit -m \
                  "Automatic update by GNUnited Nations."; \
                  make -C server/gnun triggers
```

---

[3] These days we run a more complicated script instead of this cron job in order to address some exceptional situations like concurrent builds.

The above example is for CVS; if the underlying repository is Subversion, they need to be amended accordingly. If it is GNU Bzr, remember to add a `bzr push` after commit (in the usual scenario), otherwise changes will be committed only locally. Since a distributed Version Control System can be used in multiple (sometimes radically different) ways, this step cannot be anticipated and therefore cannot be automated. Adding the `push` command in the makefile rules would not work if a so called "bound branch" is used, for instance.

In the future, this target may be extended further to do other useful things that should be "triggered" after the main build.

### 2.3.5 The `validate-all` Target

The `validate-all` target validates all HTML pages under GNUN's control. It is needed because GNUN doesn't track dependencies on the included files, so errors in those included files could pass unnoticed.

This target is invoked once a day by the official GNUN cron job.

## 2.4 Defining Articles to Be Built

The file `gnun.mk` contains variable definitions, based on which almost all other important variables are computed. In other words, the variables defined in that file directly affect the overall behavior of the build process.

There are two types of variables, which are specifically separated in order to make translators' life easier: variables that translators are free to modify and variables that are modified by the web-translators[4], ideally after performing some local tests. A translation team leader should update only `FUZZY_DIFF_LINGUAS` and `TEMPLATE_LINGUAS`; everything else is supposed to be built automagically, without manual intervention. If not, that is a bug that should be reported and fixed.

You can also request building files separately this way:

```
make -C server/gnun ../../philosophy/not-ipr.bg.html
```

'TEMPLATE_LINGUAS'

> A space-separated list with languages. Add here your language code *if and only if* you have all the SSI templates translated, and have already committed all template files listed in the `extra-templates` and `localized-ssis` variables in `server/gnun/gnun.mk`.

'FUZZY_DIFF_LINGUAS'

> Add your language code here if you want GNUN to add differences to "previous" `msgid`s in your PO files. See Section 6.2.1 [gnun-add-fuzzy-diff], page 30, for more information.

'localized-ssis'

> This variable lists the templates that are not under GNUN's control and are translated manually, like
>
> − `server/header.lang.html`
> − `server/head-include-1.lang.html`
> − `server/banner.lang.html`
> − `server/footer.lang.html`.
>
> They contain HTML code and SSI directives; PO4A doesn't extract them to any messages in PO files.

---

[4] Only because presumably, they are more familiar with GNUnited Nations' internals. From a purely technical point of view, there is no difference.

'extra-templates'
>    The extra-templates variable lists templates under GNUN control; they are rebuilt
>    from corresponding *template.lang*.po files; when the *template.lang*.po file is absent,
>    GNUN initializes and commits a file with empty msgstrs.

'optional-templates'
>    The optional-templates variable defines optional templates under GNUN control.
>    Those are the templates of low priority items, like news lines included in some
>    pages. They are managed like the additional templates listed in the *extra-templates*
>    variable, except
>
>    − *template.lang*.po is not added when absent, and
>
>    − their POTs end in .pot.opt rather than .pot.
>
>    This way, the scripts reporting outdated translations and translations that haven't
>    been converted to PO files won't complain about them unless the team decides to
>    actually commit *template.lang*.po.

'sitemap'    The sitemap variable declares the pages that are treated like sitemaps, that is,
>    an additional externally generated compendium is used when updating them. See
>    Section 2.7 [Sitemap], page 12, for more information.

'ROOT'    Add here articles that are in the server root, like home.html and keepingup.html.
>    Always write only the basename of the article, i.e. if you add these two articles,
>    the value of ROOT should be home keepingup. This is true for all the variables that
>    expect values in the form of article names.

'ALL_DIRS'
>    The list of directories containing articles, like philosophy, gnu, licenses, etc.

'gnu'
'philosophy'
'...directory...'
>    A space-separated list of basenames for articles residing in *directory*, for which
>    POTs will be generated and updated when the original article changes. If an article
>    is missing here, GNUN doesn't maintain its translations.

## 2.5 The languages.txt File

The file server/gnun/languages.txt is used when generating lists of translations; those lists
are subsequently included in all translations of the article. Every line in the file is either a
comment (when it begins with "#") or a definition for a language. A language is defined by
three TAB-separated fields. The first field defines the language suffix; it is used in file names and
in texts of the links. The second field is the name of the language in English. The third field
defines the name of the language in that language itself; it is used in texts of the links, and it
should be in UTF-8. For example:

```
de German Deutsch
```

The generated list of translations may look like this:

```
<!-- begin translist file -->
<div id="translations">
<p>
<span class="original"><a href="/distros/screenshot.en.html">English</a> [en]</spa
<span><a href="/distros/screenshot.de.html">Deutsch</a> [de]</span>  
<span><a href="/distros/screenshot.pl.html">polski</a> [pl]</span>  
</p>
```

```
        </div> <!-- id="translations" -->
        <!-- end translist file -->
```

## 2.6 Using Compendia

Compendium is a PO file including translations for common strings; it is used to fill other PO files. See Section "Using Translation Compendia" in *GNU gettext tools*. One example of such common strings is the footer text about reporting bugs and sending inquiries: when a webmaster updates footer texts in an article, GNUN will use compendia to automatically fill the translations for the new version of the strings.

GNUN has two types of compendia:

1. Site-Wide Compendia This type is used for all articles, and only contains common strings.

2. Per-Directory Compedia These files accumulate all strings from all articles in the directory, even if these strings are not repeated in different articles.

### 2.6.1 Site-Wide Compendia

Site-wide compendia are located in the `server/gnun/compendia` directory of the 'www' web repository. There are two kinds of compendia: `master.lang.po` and `compendium.lang.po`.

The first kind, `master.lang.po`, can be used to simultaneously update all occurrences of the translations of a given string. Translations from this file will override the translations from `article.lang.po`. When `master.lang.po` is updated, the translations will be rebuilt. GNUN doesn't modify this kind of compendia.

The second kind, `compendium.lang.po`, is updated automatically. GNUN finds strings that repeat many times in POTs of articles and collects them in `compendium.pot`. Then it checks all available `article.lang.po` files for translations of those strings and generates `compendium.lang.po`. This file is also used to fill missing translations, but it doesn't override the translations from `article.lang.po`, and the strings coming from `compendium.lang.po` are always marked as "fuzzy" to prevent propagation of translations that may be wrong in a different context.

When updating `compendium.pot`, some strings should be excluded even though they repeat in the POT files many times—for instance, GNUN slots for translators' notes. See Section 4.3.1 [Notes Slot], page 20. They are not real translations, this is why they are likely to be different for different articles. In order to avoid including them in compendia, GNUN checks a specific file, `exclude.pot`, and when that file contains the string, it won't be added to `compendium.pot`.

### 2.6.2 Per-Directory Templates

When GNUN detects a file named `pot` in some `po` directory, it writes to it all strings from all articles in that directory (the subdirectories are not affected); it also merges translations of those strings to files named `lang.po`, for all languages available in that directory. In its turn, files `lang.po` are used to update translations in articles from that directory.

This helps updating translations in sets of articles like `proprietary/` section, where an item may belong to many files at once. The translator can only update `lang.po`, or any article where the outdated item is listed. Then GNUN will copy the translation to all other places.

Like with `article.lang.po`, GNUN doesn't override the up-to-date translations already present in articles, and vice versa, it only adds translations to `lang.po` when they are absent. Thus, the translators can provide specific translations for a string in every article if needed—for example, for the special GNUN messages (see Section 4.3 [GNUN Slots], page 19).

## 2.7  Building Sitemap

Sitemaps differ from regular pages in two respects:

1. They are typically regenerated automatically from the texts of the articles.

2. They mostly consist of links, therefore a significant part of their text should come to translations without any changes.

Also, sitemaps contain many translated messages, so it is desirable to eliminate manual work where possible.

In order to take this into account, GNUN provides the `sitemap` variable in `gnun.mk`. All articles listed in `sitemap` are rebuilt using an additional compendium (when present), `server/gnun/compendia/sitemap-compendium.`*`lang`*`.po`.

Those additional compendia are supposed to be compiled outside of GNUN by the same tool that generates the sitemap. It is the responsibility of that tool to provide the compendia with reasonable translations whose `msgid`s match those generated by PO4A when processing the English version of the sitemap.
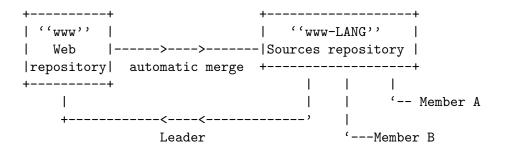
# 3 Maintaining Translations in Your Team's Repository

GNUN operates on the "official" Web repository of the Savannah project 'www', where normally only the co-ordinators of translation teams have write access. However, all translation teams have their own projects, so it is possible to take advantage of Savannah as a hosting facility to make the team work more comfortable.

The PO files provide an excellent and natural way to review each other's translations, because the translation appears right below the original message. Mutual reviews and proof-reading of translations is a crucial part of the process. Furthermore, team work is great for the community spirit; automating some of the operations also result in more time for all members to concentrate on the important tasks.

The file `GNUmakefile.team` in the 'gnun' package is a template, aimed for all translation teams who wish to use their own project's repository as a place to keep their draft translations, until they ripe and are ready to be installed officially.

The following diagram illustrates a typical workflow—it is applicable for small, medium and large teams:

```
    +----------+                   +------------------+
    | ``www''  |                   |   ``www-LANG''   |
    |   Web    |------>---->-------|Sources repository |
    |repository|  automatic merge  +------------------+
    +----------+                     |    |    |
         |                           |    |    '-- Member A
      +-----------<----<------------'    |
             Leader                     '---Member B
```

All members and the team leader commit in their project's repository—when a translation is ready, the leader checks it in the official 'www' repository. If an original article changes, a build can be invoked to synchronize (i.e. merge) the changes and optionally automatically commit them so that the draft PO files are updated; at the same time, automatic notifications are sent to the persons interested in particular translations[1]. A translator will then normally update the PO file, and commit it again in the project's Sources repository, from where the co-ordinator will pick it up and install it in 'www'.

To take advantage of this semi-automation, rename this template `GNUmakefile.team` as `GNUmakefile` and install it in the root of your project's Sources repository. Then create directories and sub-directories exactly as they are in 'www'. Do not create the `/po` sub-directories; they are redundant here. Instead, install the PO files in the normal locations where the corresponding `.lang.html` resides in 'www', for example:

---

[1] The following sections describe in more details how to setup these automatic actions; however, if you have any difficulties with that, please ask the GNU Web Translation Managers to set them up for your team on our server.

```
    Root
      |
      |--GNUmakefile
      |--home.lang.po
      |--...
      |--gnu
      |    |
      |    |
      |    +--linux-and-gnu.lang.po
      |    +--manifesto.lang.po
      |    +--...
      |
      |
      +--philosophy
      |        |
      |        |
      |        +--free-sw.lang.po
      |        +--not-ipr.lang.po
      |        +--open-source-misses-the-point.lang.po
      |        +--...
      |
      +--...
```

The next sections explain how to adapt the makefile for your team and how to use it.

## 3.1 Adapting `GNUmakefile.team` for a Specific Team

To adjust the makefile for your team, you need to edit the next variables.

'TEAM'      Set this to the language code, like `bg` or `pt-br`. It is autodetected if not defined.

'wwwdir'    The relative or absolute path to the working copy of the master 'www' repository. So
            if you have checked out your project's Sources repository at `~/projects/www-lang`
            and the 'www' Web repository at `~/projects/www`, the value of `wwwdir` should be
            `../www/` or `/home/user/projects/www/`. Note the slash at the end, it is important.

'TEAM_URL_PREFIX'
'TEAM_URL_POSTFIX'
            These variables are used to transform the path to a file in the team's working copy
            into a URL for this file; they are only needed for automatic email notifications. It
            is expected that typically the URL will point to some sort of Web interface for the
            repository, for example,

                TEAM_URL_PREFIX := http://cvs.savannah.gnu.org/viewvc/*checkout*/www-bg/
                TEAM_URL_POSTFIX := ?root=www-bg

If two variants of one language share the same project and repository (such as `zh-cn` and
`zh-tw`), they should maintain two directories with two `GNUmakefile`s and each directory having
its own tree.

Some variables are specified on the command line, and alter the behavior of the build process.

'ADD_FUZZY_DIFF'
            The program to manage differences to "previous" `msgid`s added to "fuzzy" transla-
            tions when merging PO files. When undefined, the makefile tries to find `gnun-add-`
            `fuzzy-diff` and assigns the result to this variable. See Section 6.2.1 [gnun-add-
            fuzzy-diff], page 30, for more information.

            Define to empty string to disable the feature.

'`NOTIFICATION_PERIOD=`*`days`*'
> Default period of reminders sent by the `notify` target.

'`NOTIFY=yes`'
> Make the `notify` target actually send notifications.

'`NOTIFY=no`'
> Do not send email notifications about errors. This is the default.

'`VERBOSE=yes`'
> Print more information from `cvs`, `svn` and `msgmerge`; off by default. Note that `VERBOSE` can be defined to any string, it will have the same effect.

'`VCS=yes`'  Update both '`www`' and '`www-`*`lang`*' repositories, then commit the merged PO files in the latter repository. By default, there is no VCS interaction. The VCS of the translation project repository is determined automatically; currently only CVS, Subversion, GNU Bzr, Git, Mercurial (Hg) and GNU Arch repositories are supported.

> **Caution:** The makefile rule will commit all local changes, not only those that resulted from running `msgmerge`. Thus, it is better to use a separate working copy dedicated solely for this purpose.

## 3.2 Targets in `GNUmakefile.team`

`update-team`
> Updates the working copy of team's repository.

`update-www`
> Updates the working copy of *www*.

`update`  A shortcut for making `update-team` and `update-www`.

`sync`  Merges all available PO files from the corresponding POT in *www*. If a POT is missing in the master repository (usually, because it was deleted when the original article was either split, renamed or deleted), a warning is printed for the corresponding file and no merging occurs for it.

`report`  Verifies which translations are complete, and prints a list (with statistics) of those that need to be updated. Also, it reports the translations that are not consistent with the revisions from *www* repository.

> More fine-grained notifications are implemented as the `notify` target.

`notify`  Invokes the `report` target, filters its output for different translators and sends them notifications.

> As an addition to the output of the `report` target, URLs for relevant files are written; also, when *www* and team revisions are not consistent, their differences generated with `gnun-diff-po` are optionally attached (see Section 6.2.2 [gnun-diff-po], page 30, for more details).

> The list of files to report against are configured with `nottab` file. The lines of this file that begin with '#' are ignored; other lines should contain two fields separated by ':'.

> The first field is an extended grep regular expression; the files are selected for the report if their names match this expression. See Section "Regular Expressions" in *GNU Grep manual*.

> The second field is a space-separated list of single-word translators' identifiers for which the line applies. The file may look like this:

```
^gnu/gnu-history: alice bob carol
```

```
^distros/(common-distros|distros|screenshot): dan alice
^(licenses|philosophy)/: eve bob frank
^licenses/: dan
```

Email addresses of the translators are kept in a separate file, `email-aliases`. Like in `nottab`, the lines beginning with '#' are ignored; other lines contain fields separated by ':'.

The first field is the translator's identifier, the second is space-separated list of translator's email addresses. These two fields are mandatory.

The third field is the personal reminder period in days. After that period the message is sent again even though its contents don't change. Otherwise, the `notify` target only sends messages when something changes. Naturally, no message is sent when no action is needed in the requested set of translations (unless the `force` option is used).

The team-wide default period is defined in its GNUmakefile.team using the `NOTIFICATION_PERIOD` variable.

The fourth field provides comma-separated options for the report. Currently recognized options are:

force      Forces sending messages even when there are no files to work on.

no-diffs   Disables sending *www* vs. *www-lang* differences as attachments.

www        Report cases that suggest actions by the team leader, that is, when the translation in the team repository is complete whereas in *www* it is incomplete or absent.

This is an example of `email-aliases`:
```
alice:alice@example.com abc@test.net
# Disable 'dan' temporarily.
#dan:miller@test.by:5:no-diffs,www
bob:brh@invalid.tr:3
# Note that the third field is empty; the default value for period
# will be used.
carol:carol@localhost.br translator@example.bg::no-diffs
eve:eve@test.gr:2:www
frank:gnun-notifications@example.gr:1:www,no-diffs
```

The `email-aliases` file should be kept privately because it may contain personal email addresses of people. How to do this is out of scope of GNUN and this manual.

format     A convenience rule to re-wrap all files up to the standard length. Many PO editors leave the `msgstr` as a single long line after it has been edited, but GNUN will automatically re-wrap the file to the standard line length when it is processed. Wrapping long lines in PO files is a good practice as it avoids unnecessary revisions.

This rule filters all translations with `msgcat`; basically, it reformats the lines that are longer than the default page width of `msgcat`.

publish    The `publish` rule's task is to copy all modified files to the official *www* repository. It depends on the `format` target to ensure that all files are re-wrapped to the standard line length limit, but deliberately does not depend on `sync VCS=yes`. Usually, one would run `make publish` when one or a bunch of PO files are in a satisfactory condition to be published, and this rule is just a convenience to avoid multiple manual `cp` invocations. As a rule of thumb, before running this rule it is sane to run `sync` and correct any "fuzzy" messages and other problems, if necessary.

The `publish` rule does not depend on `sync` explicitly, because that would be a nuisance for offline operations and basically unnecessary when the committer is fairly confident that there are no changes to (re-)merge. A hard dependency on `sync` would slow down the operation considerably.

As usual, when committing to the official repository, it is always a good practice to examine the output of `cvs diff`.

Invoking `make publish` prints warnings and does not actually copy the affected file if the sub-directory in "www" is non-existent or the corresponding `.pot` is missing.

Typically, after an editing session (whether it involves actual editing or just merging contributions from team members), one would do:

```
$ make sync VCS=yes
$ make publish
$ cd wwwdir
$ cvs up
(Add all new translations, if any.)
  $ cvs add file ...
$ cvs diff
$ cvs commit -m "Some descriptive message."
```

`clean`      Deletes all backup and auto-generated files.

`make VCS=yes` is the recommended command to be run periodically. To check the status of the translations, run `make report`.

Feel free to replace all strings with equivalents in your native language and of course—do not hesitate to extend this file and modify it as much as you like. If you come up with something interesting, it would be nice to send a message to `bug-gnun@gnu.org`, so that `GNUmakefile.team` gets updated for all teams' benefit.

## 3.3 Automatic Synchronization and Status Reports

It is convenient to invoke such synchronization automatically, for example once every day. If you have enabled commit notifications for the project's repository, any new changes will be visible for subscribers. Here is an example crontab entry:

```
# m h  dom mon dow   command
@daily              cd $HOME/projects/www-lang; make VCS=yes
```

The job doesn't have to run on the team leader's machine, since all team members have write access to their project repository.

If desired, you could set up another job to report the status of the translations weekly or fortnightly, for example:

```
# m h  dom mon dow   command
@weekly             cd $HOME/projects/www-lang; \
                    make report | mail -s "Weekly statistics" \
                    www-lang-list@gnu.org
```

**Caution:** Most cron implementations do not allow the character '\' as a line continuation character—the example shown is made that way for better readability.

# 4 Working with PO Files

This section provides technical details about working with PO files. For general information about PO editors, see Section "Editing PO Files" in *GNU Web Translators Manual*.

## 4.1 Starting a New Translation

To start a new translation, the most simple way is to copy the existing POT as `article.lang.po`, where *lang* is your language code. The name of the POT is `article.pot` or, when you are translating an optional template, `article.pot.opt` (see [optional-templates], page 10). All these files are situated in the `po/` subdirectory of the directory containing the HTML file of the article. See Section 6.1 [Files and Directories], page 27, for more info on GNUN directory layout. For example, to prepare for a new translation of the essay `https://www.gnu.org/philosophy/free-sw.html` in Manx, you can simply `cd philosophy/po; cp free-sw.pot free-sw.gv.po` and then edit the latter. If `free-sw.pot` does not exist it is because either the article is not yet "templated" (i.e. migrated to the new style), or the GNUN maintainers have not yet added it to the value of the appropriate variable in `server/gnun/gnun.mk`. In that case, just ask them to do what's needed for the POT to be generated.

You could also use the `msginit` utility that would populate the PO file header with the right information, provided your environment is set up correctly. See Section "msginit Invocation" in *GNU gettext tools*.

GNUN also provides a customized script to automatically fill more header fields. See Section 6.2.3 [gnun-init-po], page 31.

## 4.2 Filling the PO File Header

The PO file header as generated usually looks like this:

```
# SOME DESCRIPTIVE TITLE
# Copyright (C) YEAR Free Software Foundation, Inc.
# This file is distributed under the same license as the PACKAGE package.
# FIRST AUTHOR <EMAIL@ADDRESS>, YEAR.
#
#, fuzzy
msgid ""
msgstr ""
"Project-Id-Version: PACKAGE VERSION\n"
"POT-Creation-Date: 2008-02-06 16:25-0500\n"
"PO-Revision-Date: YEAR-MO-DA HO:MI+ZONE\n"
"Last-Translator: FULL NAME <EMAIL@ADDRESS>\n"
"Language-Team: LANGUAGE <LL@li.org>\n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=CHARSET\n"
"Content-Transfer-Encoding: ENCODING\n"
```

You have to edit the header to match the already established conventions, and the rules for gnu.org translations. For reference, here is a list with all fields explained:

'`Project-Id-Version`'
> Add here the filename of the original article, without the sub-directory, like "body-include-1.html" or "free-sw.html".

'`POT-Creation-Date`'
> Do not edit this field, it is already set when the POT is created.

'PO-Revision-Date'
> Likewise, do not edit. This field is automatically filled in when you save the file
> with any decent PO editor.

'Last-Translator'
> The name and email address of the last translator who has edited the translation.
> Pay attention that normally this is the name of a member of your team, it can be
> the translation team leader if he/she was the person who updated the translation.
> For example:
>
>> Elvis Parsley <king@grassland.com>

'Language-Team'
> This field should contain the mailing list on which the translation team can be
> reached—usually `www-LANG-...@gnu.org`. Example:
>
>> Czech <www-cs-general@gnu.org>

'MIME-Version'
> Leave it like it is.

'Content-Type'
> Usually this is `text/plain; charset=UTF-8`; change the charset accordingly.

'Content-Transfer-Encoding'
> Set this to `8bit`. Note that the PO file header ends with this field, and it should
> contain a newline ('`\n`'). Unfortunately, some PO editors remove the newline, which
> causes an unnecessary revision when the file is automatically modified by GNUN's
> rules.

Here is an example of a properly edited header:

```
# Bulgarian translation of https://www.gnu.org/philosophy/free-sw.html
# Copyright (C) 2008 Free Software Foundation, Inc.
# This file is distributed under the same license as the gnu.org article.
# Yavor Doganov <yavor@gnu.org>, 2008.
#
msgid ""
msgstr ""
"Project-Id-Version: free-sw.html\n"
"POT-Creation-Date: 2008-02-06 16:25-0500\n"
"PO-Revision-Date: 2008-02-09 15:23+0200\n"
"Last-Translator: Yavor Doganov <yavor@gnu.org>\n"
"Language-Team: Bulgarian <www-bg-list@gnu.org>\n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=UTF-8\n"
"Content-Transfer-Encoding: 8bit\n"
```

Notice the absence of the "fuzzy" marker; you should "unfuzzy" the header after entering
the necessary information (this is done by simply pressing `TAB` in PO mode).

It is recommended that you wrap all lines in the comments to be less than 80 charac-
ters; that looks better from a purely aesthetic point of view and improves the performance
of `GNUmakefile.team`'s `publish` rule (see [publish], page 16).

## 4.3 Special GNUN messages

There are some special messages that appear in the POT and PO:

'*GNUN-SLOT: TRANSLATOR'S NOTES*'

>    This is for translator's notes that are injected in the resulting translation. See
>    Section 4.3.1 [Notes Slot], page 20, for more information. If your translation does
>    not have notes, you *must* translate this as a space:

```
msgid "*GNUN-SLOT: TRANSLATOR'S NOTES*"
msgstr " "
```

'*GNUN-SLOT: TRANSLATOR'S CREDITS*'

>    This is again optional, and should contain the name (and address) of the person
>    who made the translation. "Translate" this string as a space if you do not want
>    your name to appear there. See Section 4.3.2 [Credits Slot], page 21.

## 4.3.1 The Special Slot for Translator's Notes

Sometimes it is necessary to complement the translation of an essay with translator's notes. The special message *GNUN-SLOT: TRANSLATOR'S NOTES* is designed to serve this purpose. If your translation doesn't have notes, you should "translate" the msgstr as a space (SPC)—otherwise the PO file will be considered incomplete, which is not what you want. Here is an example how to use translators' notes in a PO file:

```
# type: Content of: <p>
msgid ""
"To understand the concept, you should think of <q>free</q> "
"as in <q>free speech,</q> not as in <q>free beer.</q>"
msgstr ""
"Translated message, where you want to clarify beer<sup><a "
"href=\"#TransNote1\">1</a></sup>, presumably because the "
"expression in your language is different"
...
...
# type: Content of: <div>
#. TRANSLATORS: Use space (SPC) as msgstr if you don't have notes.
msgid "*GNUN-SLOT: TRANSLATOR'S NOTES*"
msgstr ""
"<h3>Translator's notes</h3>\n"
"<ol>\n"
"<li id=\"TransNote1\">Note clarifying the text.</li>\n"
"</ol>\n"
```

Certainly, everything in the msgstrs should be in your native language; we use English here, so that everyone understands the example. If you have more notes, each subsequent one should be with incremented number, i.e. 'TransNote2', 'TransNote3', etc. and you have to add them as more <li> elements accordingly.

Do not worry about the \n character—it is inserted automatically when you press RET. It is not compulsory that notes start on a new line, this is the recommended way simply because it is easier to edit them.

It is important to follow this specification, because notes will look consistent in all languages and will be clearly distinguishable from authors' footnotes, if any. Furthermore, it will be easier to define a special CSS class for them, and also to convert the translations in other formats such as Texinfo—when these conversions are implemented.

### 4.3.2 The Special Slot for Translator's Credits

Most of the translators usually put their name under the translation, in the "footer" area. This is entirely acceptable, since some readers prefer to send buggestions directly to the translator. Also, giving credit where credit is due is a natural thing.

Like the previous slot, you should "translate" it as a space if you don't want your name to appear there.

Here is an example of the recommended way to specify credits:

```
<b>Traduction</b>: Benjamin Drieu
<a href="mailto:foo@example.org">&lt;foo@example.org&gt;</a>,
2007, 2008.
```

It is highly desirable to use this form, but you may omit the email address or add a link to translator's noncommercial personal home page, provided that the translation team leader ensures that it constantly meets the linking criteria for gnu.org. Please follow the FSF HTML Style Sheet when adding URIs or other information.

## 4.4 Wrapping Long Lines

Most of the PO editors do not wrap long lines that inevitably appear in `msgstr`s. If that happens, long lines make reading subsequent diffs harder, and are generally annoying for most people. If this issue bothers you, you can "normalize" the already finished PO translation by executing on the command line `msgcat -o file.po file.po`, before installing it in the repository. Either way, the build system will treat it as a valid PO file.

For those lucky Emacs users, here is a code snippet that you can put in your `.emacs`; doing `M-x po-wrap` while in PO mode will wrap all long lines:

```
(defun po-wrap ()
  "Filter current po-mode buffer through 'msgcat' tool to wrap all lines."
  (interactive)
  (if (eq major-mode 'po-mode)
      (let ((tmp-file (make-temp-file "po-wrap."))
    (tmp-buf (generate-new-buffer "*temp*")))
(unwind-protect
    (progn
      (write-region (point-min) (point-max) tmp-file nil 1)
      (if (zerop
   (call-process
    "msgcat" nil tmp-buf t (shell-quote-argument tmp-file)))
  (let ((saved (point))
(inhibit-read-only t))
    (delete-region (point-min) (point-max))
    (insert-buffer tmp-buf)
    (goto-char (min saved (point-max))))
(with-current-buffer tmp-buf
  (error (buffer-string)))))
  (kill-buffer tmp-buf)
  (delete-file tmp-file)))))
```

It is highly desirable that you check if the PO file you finished translating (or editing) is valid, before committing it. This is done by running `msgfmt -cv -o /dev/null file` or by simply pressing `V` in PO mode. The build system automatically verifies each PO file when invoked with `VALIDATE=yes`, but you won't get a warm and fuzzy feeling if a stupid typo you made halts the whole update of all translations. Such things happen to everyone, so it is a good practice to check before you actually commit.

## 4.5 Useful Hints for Editing PO Files

This section contains additional explanations, some in the form of advice and recommendations; not all of them are strictly related to PO files editing.

- When you install a new translation of an article (that is different from a server template), all you need to do is to add your PO file in the appropriate `/po` sub-directory.

  In the next build, your *article.lang*`.html` will be built and the link to it will be added to the list of translations and propagate to all translations, provided that they are under GNUN's control.

- If you don't feel comfortable editing `gnun.mk`, do not worry. Someone from the GNUN maintainers will notice and will amend `TEMPLATE_LINGUAS` for you, as appropriate.

- Dealing with obsolete strings. Elements which are removed from the original articles appear in the PO files as "obsolete" strings—the translation is not lost, but they are marked in a special way at the end of the PO file. You don't have to update a PO file if it contains obsolete strings—do this only if it has "fuzzy" or "untranslated", and of course when you want to improve the existing translated ones. Sometimes these obsolete strings are useful, and they can save time. For example, if you anticipate that the deleted text may reappear some time in the future, you can preserve the string and hopefully it would be marked as "fuzzy" when this happens. Failing that, you can still copy it and yank it at the appropriate place.

- You can add comments to every message in a PO file—for example if you want to remember that you have to do something, or to remind you why this particular message is translated in a special way. These comments do not appear in the generated HTML source.

- Sometimes, especially when the original message contains many links, it is easier to copy it to `msgstr` and edit the latter by translating the English text. In PO mode of Emacs, this is done by `C-j`. This is useful also for large chunks of text in `<pre>` elements, which normally you would want to preserve verbatim.

- Do not use HTML entities for non-ASCII letters as in the English original. They are harder to type and read. So, if there is '`&uuml;`' and this is a character from the alphabet of your language, just write it as 'ü' directly.

- Wrapping of `msgstr` using `M-q` in Emacs (or other means) is considered harmful. It is best to leave GNUN (or more precisely, Po4a) do the wrapping—that way all generated HTML translations will have predictable results. This will help tremendously for the conversion to other formats, like Texinfo. Also, note that not all elements are wrapped by default, so deliberately wrapping the text inside the `msgstr` could lead to an invalid page or a page that is valid, but is rendered incorrectly by the web browser.

## 4.6 Transforming Existing Translation in PO Format

Migrating an existing translation to a PO file format is basically editing the header as described in the previous section, and populating each of the messages by copying the already translated text and markup from the existing translation in HTML format in the relevant message.

Typically, you will visit `po/foo.`*lang*`.po` (in PO mode) and `foo.`*lang*`.html` (in HTML mode) in another buffer. Then you can copy a paragraph or an element from the latter and yank it in the relevant message in the former. Be extra careful, since this is the time to check *precisely* that the translation corresponds to the original. Further changes will be reflected, but if your "initial" PO file is not a 100% match, that would not necessarily mean that it is an improvement. Since it is very easy to do this kind of check, because the relevant `msgid` and `msgstr` appear one above the other in the same buffer (or the similar concept in other PO editors), please *do* perform this initial sanity check even if you are confident that the translation you have been yanking strings from is a completely up-to-date translation.

There is also a semi-automatic way to produce an initial PO file. You checkout the revision of the English page, `foo.html`, that corresponds to the latest revision of the translation, `foo.lang.html`. Then you run `gnun-preconvert` (see Section 6.2.6 [gnun-preconvert], page 32):

```
gnun-preconvert foo.lang.html foo.html
```

If some passages in `foo.lang.html` don't match the structure of `foo.html`, error messages will be displayed. Check them, adjust the files and try again. When you succeed, the result will be written to foo.lang.po. After that, run `gnun-merge-preconverted` (see Section 6.2.7 [gnun-merge-preconverted], page 32):

```
gnun-merge-preconverted -C compendium.lang.po foo.lang.po foo.pot
```

If you have no compendium, just omit the "`-C compendium.lang.po`" part.

You get `foo.lang.po` where all messages are marked as "fuzzy" (unless you use a compendium); you still should make sure that the translations correspond to the original and remove those "fuzzy" marks. The script adds differences against "previous" `msgids` to facilitate checking.

There is no need to delete the existing HTML translation, GNUN will automatically overwrite it. The only thing the translator should do is commit the PO file to the repository.

When an essay has been translated by several people through the years, it is important that this information is recorded and reflected in the PO file. In the future, special targets may be added to enable the FSF to check who translated a particular article, and when.

A recommended way to do this is as follows:

```
# French translation of https://www.gnu.org/philosophy/bsd.html
# Copyright (C) 2006, 2007, 2008 Free Software Foundation, Inc.
# This file is distributed under the same license as the gnu.org article.
# Cédric Corazza <cedric.corazza@wanadoo.fr>, 2006, 2008.
# Jerôme Dominguez <taz@gnu.org>, 2007.
```

In this example, it is clear that Cédric made the initial translation, Jerôme made some changes in 2007, and the original translator returned in 2008 and continued maintaining it.

# 5 Tips and Hints for Webmasters

This section contains some tips and general recommendations for webmasters in no particular order—it is not mandatory to follow them, but doing so will make translators' lives substantially easier.

First and foremost, respect translators' work—it is ungrateful and hard, undoubtedly much harder than translation of programs. It is important to have as many translations as possible, and make them as good as possible, and you don't have to make titanic efforts to help.

If you plan to edit a certain page extensively, please do so within a reasonable time frame, for example a day or preferably, several hours. That way, the translators who are quick to update it immediately won't be disappointed if the POT changes again in the next GNUN run.

## 5.1 Modifying Templates

Any significant structural diversion from `boilerplate.html` in a specific article may result in errors from GNUN. Any untested intrusive updates to the server templates (such as changing the entire look & feel of the site) will probably break GNUN build process. Of course, this does not mean that no changes should happen—only that they must be applied in a sandbox first, to ensure a smooth transition.

The script `gnun-validate-html` is useful for webmasters who want to verify if their (potentially intrusive) changes result in a valid markup. Before committing your changes, you can check whether the file is valid by running

```
gnun-validate-html --root . philosophy/not-ipr.html
```

See Section 6.2.9 [gnun-validate-html], page 33, for more information.

## 5.2 Comments for Translators

If you want a comment to be visible for translators, place it *inside* the element, for example:

```
<p>
<!--TRANSLATORS: Note that foo is bar in this context.-->
The fooish bar mumbles bazzling.
</p>
```

This will result in:

```
# type: Content of: <p>
#. TRANSLATORS: Note that foo is bar in this context.
msgid "The fooish bar mumbles bazzling."
msgstr ""
```

As per the established convention, start the comment with `TRANSLATORS:` to catch their attention, and do not add a space after the beginning of the HTML comment (`<!--`), since this will unnecessarily indent the comment in the POT.

## 5.3 Localized URLs

Some articles and templates may contain diagrams or other images with English texts that call for translations. In order to make the translated versions appear in the respective translations, add a HTML comment like this to the English page:

```
<!-- GNUN: localize URL /philosophy/category.png,
  /licenses/template-diagram.png and /graphics/jesus-cartoon.jpg -->
```

The URLs are separated with spaces. One trailing comma at the end of every word is removed if present. Words without a dot, such as 'and', do not count as URLs; they are ignored.

Such comments will be extracted nightly and compiled into per-article lists of URLs in `localized-urls.mk`.

At the end of every build GNUN will check if the respective `philosophy/category.`*`lang`*`.png` and other files are present in the working copy; then GNUN will substitute the references in the HTML file of the translation.

GNUN relies on URLs being absolute, starting from the root directory as suggested in the FSF HTML Style Sheet Guidelines (`https://www.gnu.org/server/fsf-html-style-sheet.html#FilenameAndURLGuidelines`).

Please be sure to also commit the images in their source form (typically, in SVG format). It's important both for the English image so that the translators could derive their localized versions from it, and for the translations so that people could maintain them.

## 5.4 Splitting Long Passages

GNUN splits typical text by paragraphs, and translators write the translations for the resulting parts of the text one by one. When a part is too long, it is hard to translate and proofread the translation; the likelihood of skipping some phrases is relatively high.

In order to make our translators' life easier, it is desirable to keep the paragraphs short (no more than 350–700 characters). If rearranging the paragraphs is not an option, you can use a conventional separator, `<span class="gnun-split"></span>`:

```
<p>GNUN splits typical text by paragraphs, and translators write
the translations for the resulting parts of the text
one by one. <span class="gnun-split"></span> When a part
is too long, it is hard to translate and proofread the translation;
the likelihood of skipping some phrases is relatively high.</p>
```

It is important that the separator be inserted between complete sentences, because different languages may require different orders of parts of the sentence, and the translator has no control over the sequence of the strings in the translation.

## 5.5 Sorting Lists

Sometimes a list is sorted alphabetically to make search easier for the visitors, and this property should be preserved in translations. GNUN has specific marks for such lists.

```
<!-- GNUN-SORT-START -->
```

This HTML comment marks the begin of the list.

```
<!-- GNUN-SORT-STOP -->
```

This comment marks the end of the list.

```
<!-- GNUN-SORT-NEXT-ITEM -->
```

This mark separates list items.

The next two marks are optional, they delimit the key to sort items.

```
<!-- GNUN-SORT-BEGIN-KEY -->
```

```
<!-- GNUN-SORT-END-KEY -->
```

When extracting the keys, all HTML tags are removed, leading and trailing whitespace is trimmed.

Since the items are reordered in translations, webmasters should carefully choose places for the separators: reordering shouldn't break the page.

The translators can modify the sorting order using HTML comments if needed. For example, let the English text say

```
<!-- GNUN-SORT-BEGIN-KEY -->
```

```
<h3>Afrikaans</h3>
<!-- GNUN-SORT-END-KEY -->
```

In the PO file, this transforms to an `msgstr` like

```
Afrikaans
```

The translator inserts a comment in the translation:

```
<!-- boer --> afrikaans
```

Now the item sorts like '`boer afrikaans`'.

Additional care should be taken when the items have a separator like

```
<!-- GNUN-SORT-START -->
A |
<!-- GNUN-SORT-NEXT-ITEM -->
B |
<!-- GNUN-SORT-NEXT-ITEM -->
...
<!-- GNUN-SORT-NEXT-ITEM -->
Y |
<!-- GNUN-SORT-NEXT-ITEM -->
Z
<!-- GNUN-SORT-STOP -->
```

Note that the last item has no separator ('`|`'). When translating, make sure that the item to be last *in your translation* has no separator, and all other items have it.

# 6 Unexciting Information for GNUN's Operation

This chapter might be of interest probably only to people who would have special interest in the software, plan to enhance it or develop a front-end, except the section about scripts that includes descriptions of programs that may also be useful for the translators.

## 6.1 Files and Directories

This is a brief diagram of *www* working copy; note that in this section we don't describe the files out of GNUN control, including the old HTML-only translations.

```
Legend:
*    The file is maintained locally, without committing it to CVS.
#    The file is edited manually.
@    The file is regenerated automatically.
%    The file is edited both manually and automatically.
#/@ The file is generated automatically, but outside of GNUN.
```

```
Root/
  |
  +--proprietary/
  |   |
  |   +--proprietary.html #
  |   +--proprietary.lang.html @
  |   +--malware-cars.html #
  |   +--malware-cars.lang.html @
  |   +--...
  |   |
  |   +-----------------------po/
  |                            |
  +--home.html #               +--malware-cars.lang.po %
  +--home.lang.html @          +--malware-cars.pot @
  +--planetfeeds.html #/@      +--malware-cars.proto * @
  +--planetfeeds.lang.html @   +--malware-cars.translist @
  +--po/                       |
  |   |                        +--malware-cars.lang-en.html @
  |   +--home.pot @            +--malware-cars.lang-diff.html @
  |   +--home.proto * @        +--proprietary.lang.po %
  |   +--home.translist @      +--pot @
  |   +--home.lang.po %        +--lang.po %
  |   +--home.lang-en.html @   +--...
  |   +--home.lang-diff.html @
  |   +--planetfeeds.proto * @
  |   +--planetfeeds.pot.opt @
  |   +--planetfeeds.lang.po %
  |   +--...
  |
  +--server/
  |   |
  |   +-----------------------gnun/
  |   |                        |
  |   +--sitemap.html #/@      +--GNUmakefile * #
  |   +--sitemap.lang.html @   +--config.mk * #
  |   +--po/                   +--gnun.mk #
  |   |   |                    +--priorities.mk #
  |   |   +--sitemap.pot @     +--languages.txt #
  |   |   +--sitemap.proto * @ +--compendia/
  |   |   +--sitemap.lang.po % |
  |   |   |                    +--master.lang.po #
  |   |   +--body-include-2.pot @       +--compendium.lang.po @
  |   |   +--body-include-2.proto * @   +--compendium.pot @
  |   |   +--body-include-2.lang.po %   +--excluded.pot #
  |   |   +--...
  |   |
  |   +--body-include-2.html #
  |   +--body-include-2.lang.html @
  |   +--...
  +--...
```

A description of some files follows.

`.../file.html`
> The English files.

`.../file.lang.html`
> The *lang* translation of `.../file.html`.

`.../po/file.proto`
> Intermediate files generated from `file.html`; they are used only internally. Essentially, GNUN inserts its special slots. See Section 4.3 [GNUN Slots], page 19, for more details.

`.../po/file.translist`
> The list of translations for `.../file.html`. It is included in `.../file.html` and all its translations.

`.../po/file.pot`
> PO templates for articles and included files.

`.../po/file.pot.opt`
> POTs of optional templates. See [optional-templates], page 10, for more details.

`.../po/file.lang.po`
> Translations of `file.pot` or `file.pot.opt` committed by the translators and merged by GNUN.

`.../po/file.lang-en.html`
> The latest revision of the English file for which a full translation into *lang* language was provided.

`.../po/file.lang-diff.html`
> The highlighted differences between the current version of `...file.html` and `.../po/file.lang-en.html`.

`.../po/pot`
> Per-directory template. It contains all strings from all articles in this directory. See Section 2.6.2 [Per-Directory Templates], page 11.

`.../po/lang.po`
> Per-directory compendium. It is only used and regenerated when `.../po/pot` exists. See Section 2.6.2 [Per-Directory Templates], page 11.

`planetfeeds.html`
> A list of news; it is automatically generated outside of GNUN. For GNUN, it is an optional template.

`server/` Most template SSI files are located in this directory.

`server/sitemap.html`
> The sitemap; it is generated automatically by a script that is not included in GNUN. See Section 2.7 [Sitemap], page 12.

`server/gnun/`
> The directory for GNUN files.

`server/gnun/GNUmakefile`
`server/gnun/config.mk`
> These files are maintained locally. They are used to rebuild the translations. See Section 2.1 [Invoking GNUN], page 5.

`server/gnun/gnun.mk`
> The file that defines the list of files to build. See Section 2.4 [Main Variables], page 9.

`server/gnun/priorities.mk`
>       This file defines the list of translation priorities. See [priorities.mk], page 8.

`server/gnun/languages.txt`
>       Canonical names for languages. See Section 2.5 [languages.txt], page 10.

`server/gnun/compendia`
>       The directory for compendia. See Section 2.6 [Compendia], page 11.

## 6.2 GNUN Scripts

For the time being there are several helper scripts. Some of them are used internally as commands with certain arguments in the makefile rules; other scripts were written specifically to facilitate some mechanical actions, like initially filling the headers in the PO files. They all can be invoked separately, as stand-alone programs.

### 6.2.1 The `gnun-add-fuzzy-diff` Script

This script adds comments with differences of current `msgid`s against "previous" ones to "fuzzy" translations in a PO file. To produce the differences `wdiff` is used. (See Section "The word difference finder" in *GNU wdiff*, for more information.) This may be useful to figure out what has changed. In fact, it wraps around a `sed` script used in GNUN internally.

>       `gnun-add-fuzzy-diff [OPTION...] [FILE]`

`-i`
`--in-place`
>       Edit the file in place.

`-v`
`--version`
>       Display copyright and version information and exit.

`-h`
`--help`      Display usage information and exit.

The script is invoked automatically when requested through the `FUZZY_DIFF_LINGUAS` variable in `GNUmakefile` (see [FUZZY_DIFF_LINGUAS], page 9) and through the `ADD_FUZZY_DIFF` variable in `GNUmakefile.team` (see [ADD_FUZZY_DIFF], page 14).

### 6.2.2 The `gnun-diff-po` Script

This script compares two versions of a PO file. It produces a HTML page with a table that contains original strings with highlighted differences of their translations.

If the sets of original strings in the input PO files differ, one of them will be merged with the other in order to eliminate the differences in the original strings.

The results are written to standard output. Example:

>       `gnun-diff-po [OPTION...] PO_FILE1 PO_FILE2 > diff.html`

`-1`
`--merge-against-first`
>       Adjust the original strings of PO_FILE2 against PO_FILE1 when their sets differ. This is the default.

`-2`
`--merge-against-second`
>       Adjust the original strings of PO_FILE1 against PO_FILE2 when their sets differ.

`--no-common`
>       Suppress strings that don't differ.

`-t`
`--title=`*`title`*

> Specify the title of the output HTML page.

`-v`
`--version`

> Display copyright and version information and exit.

`-h`
`--help`     Display usage information and exit.

### 6.2.3 The `gnun-init-po` Script

This script initializes a PO file using the POT generated with GNUN, and fills some fields in the header. It also optionally uses a compendium (or compendia) to fill translations.

     gnun-init-po [OPTION...] POT

`-C`
`--compendium=`*`comp`*

> Specify a compendium to use. You can issue this option many times to use multiple compendia simultaneously. The suffix of compendium is used when the language suffix is not specified with the `--language` option.

`-d`
`--disable-diffs`

> Don't add diffs to "previous" messages.

`-g`
`--team="`*`team <list>`*`"`

> Specify team's name and mailing list.

`-l`
`--language=`*`lang`*

> Specify language suffix, e.g "bg". The suffix also defines the name of the language which is used in some fields of PO file header.

`-t`
`--translator="`*`full name <email>`*`"`

> Specify translator.

`-v`
`--version`

> Display copyright and version information and exit.

`-h`
`--help`     Display usage information and exit.

The PO file name is guessed from the name of POT and language suffix; the file is created in the current working directory.

### 6.2.4 The `gnun-link-diff` Script

This script finds mismatched links and anchors and fills some fields in the header. Usually this is a typo, though in *very* rare cases changes in links are justified.

The script takes into account possible translator's notes and changes in links to Creative Commons licenses. See Section "Distribution Terms" in *GNU Web Translators Manual*.

The output is a HTML page with highlighted differences. In the beginning, numbers of mismatched links and anchors are written. Links starting with "mailto:" are not counted, but highlighted.

The exit status of the script is 0 when the counts of mismatched links and anchors are zero, 1 when their sum is not zero, 2 when an error occurred.

```
gnun-link-diff [OPTION...] FILE
```

`-l`
`--language=`*`lang`*

> Specify language suffix, e.g "bg". The suffix is used to adjust links to Creative Commons licenses. When unspecified, it is filled from the file name.

`-t`
`--title="`*`page title`*`"`

> Specify title. When unspecified, the file name is used.

`-v`
`--version`

> Display copyright and version information and exit.

`-h`
`--help`    Display usage information and exit.

### 6.2.5 The `gnun-po4a-translate` Script

In newer PO4A releases, `po4a-translate` emits a deprecation warning. The `gnun-po4a-translate` script makes a drop-in replacement for the deprecated script using `po4a`, as the warning recommends. When the `configure` script is invoked with the `--enable-gnun-po4a-translate` option, GNUN is configured to use the replacement instead of `po4a-translate`.

### 6.2.6 The `gnun-preconvert` Script

This script uses `po4a-gettextize` to convert a translation from HTML to PO format (see `https://po4a.org/man/man1/po4a-gettextize.1.php`). If the conversion is successful, you can merge the result with the new POT using `gnun-merge-preconverted`.

```
gnun-preconvert [OPTION...] TRANSLATION MASTER
```

`-e`
`--encoding`

> Specify the encoding of TRANSLATION (if other than UTF-8).

`-v`
`--version`

> Display copyright and version information and exit.

`-h`
`--help`    Display usage information and exit.

### 6.2.7 The `gnun-merge-preconverted` Script

This script takes `po4a-gettextize` output, adds current `msgids` as "previous" values, merges the file with the new POT, and adds differences against "previous" values like `gnun-add-fuzzy-diff` does.

```
gnun-merge-preconverted [OPTION...] PO POT
```

`-C`
`--compendium`

> Specify the compendium (if any). This option can be used more than once to specify multiple compendia.

`-d`
`--disable-diffs`

> Don't add diffs to previous messages in the format of gnun-add-fuzzy-diff.

`-v`
`--version`

Display copyright and version information and exit.

`-h`
`--help`        Display usage information and exit.

### 6.2.8 The `gnun-report` Script

This script generates HTML reports about translations of a given team. The HTML contains a set of tables sorted by priority and translation status (existing translations that need maintenance, untranslated files, complete translations).

The script depends on the presence of the `priorities.mk` file in the `server/gnun` subdirectory of the working copy of 'www' repository.

The results are written to standard output. Example:

```
gnun-report --root=../www -t bg > report-bg.html
```

`-t`
`--team=lang`

Specify the language code of the team, for example, 'ml'.

`--root`        Specify the top directory of the working copy; the default is the current directory.

`-l`
`--language=language`

Specify the language name, for example, 'Malayalam'. When this option is missing, `gnun-report` tries to figure out the language name based on the language code provided with the "-t" option.

`-v`
`--version`

Display copyright and version information and exit.

`-h`
`--help`        Display usage information and exit.

A cron job commits updated reports for all active teams to GNUN project web repository, typically twice an hour. The links to those reports are provided on the GNUN Reports (`https://www.gnu.org/software/gnun/reports/reports.html`).

There is also a target in `GNUmakefile` to generate text reports intended for monthly messages sent to the teams. See Section 2.3.3 [report], page 8.

### 6.2.9 The `gnun-validate-html` Script

This is a Bash script whose purpose is to "validate" both the original and translated articles to make sure that they conform to the respective W3C standard. Sometimes webmasters make mistakes, and translators too, so this tool is useful to catch errors of that kind.

GNUN enforces HTML validation at build time by default.

The script expects only one *file* as the last argument and will exit with an error if it is not specified (which might be the case when an automatic variable is not expanded properly due to a bug in the makefile). Example:

```
gnun-validate-html --root . philosophy/free-sw.html
```

`--root=directory`

Specify the top *directory* of the working copy; the default value is `../..`.

`--expand-to=file`

Save the expanded HTML as *file*.

`--apache-vars="`*var1=val1;var2=val2...*`"`
> Specify additional (or override assumed) emulated predefined Apache variables.

`-V`
`--verbose`
> Produce more detailed output intended for automatic email reports; essentially, it adds the expanded HTML to facilitate finding errors by people who receive the report.

`-v`
`--version`
> Display copyright and version information and exit.

`-h`
`--help`      Display usage information and exit.

## 6.2.10 The `mailfail` Script

This is a helper script that runs a command, and mails the output of that command in case it exits with a non-zero exit status. `mailfail` depends on GNU Mailutils, or a compatible implementation, such as BSD's mailx.

Usage:

```
mailfail [--dry-run] RCPT SUBJECT CMD [ARG ...]
```

The `mailfail` script accepts the following options:

`--dry-run`
> Does not send the email message.

`RCPT`       The recipient of the message in a valid format, like `someone@somehost.org`.

`SUBJECT`    The subject of the message; if it is longer than a word you should guard it with quotes.

`CMD`        The command you want to run and send a mail in case it fails.

`ARG...`     The arguments of `CMD`, if any.

Here is a typical example, similar to the way it is used in GNUN:

```
mailfail translators@example.org "Bad PO" msgfmt -cv -o /dev/null bg.po
```

This will check the validity of `bg.po` with the `msgfmt` program and in case there are errors, a message will be sent to the specified address with 'Bad PO' as subject and the error output from `msgfmt` as body.

`mailfail` inherits the exit status of the command being run. If an argument is missing, the usage information is printed to the standard output and the exit code is 1.

## 6.2.11 The `validate-html-notify` Script

This script is a wrapper around `gnun-validate-html` (see Section 6.2.9 [gnun-validate-html], page 33); it is necessary because it is hard to capture the output of the program from a program that itself captures the output of another program that it runs.

Usage:

```
validate-html-notify [--dry-run] RCPT FILE
```

`--dry-run`
> Does not actually send the message, just like `mailfail`. In this case it runs `gnun-validate-html` without `--verbose` because it is expected that the expanded file will be available locally.

RCPT        The recipient of the message.

FILE        The HTML file that has to be validated for compliance with the W3C standard.

    The subject of the message is hardcoded in the script, since this wrapper has a specific task and cannot be used to invoke an arbitrary command—use `mailfail` for that. See Section 6.2.10 [mailfail], page 34.

# 7 Reporting Bugs

GNUnited Nations, like any other software, is not bug free. There are some known bugs and annoyances, which are listed in the `TODO` file, but it is absolutely certain that there are more which we know nothing about.

If you encounter a bug, or if you have suggestions of any kind, please do not hesitate to report them at `bug-gnun@gnu.org` or GNUN Bug Tracker (`https://savannah.gnu.org/bugs/?group=gnun`).

# Appendix A  GNU Free Documentation License

Version 1.3, 3 November 2008

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

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

0. PREAMBLE

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

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

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

1. APPLICABILITY AND DEFINITIONS

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

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

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

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

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

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

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

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

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

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

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

2. VERBATIM COPYING

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

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

3. COPYING IN QUANTITY

   If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both

covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

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

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

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

4. MODIFICATIONS

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

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

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

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

D. Preserve all the copyright notices of the Document.

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

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

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

H. Include an unaltered copy of this License.

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

Title Page, then add an item describing the Modified Version as stated in the previous sentence.

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

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

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

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

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

O. Preserve any Warranty Disclaimers.

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

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

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

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

5. COMBINING DOCUMENTS

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

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

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

6. COLLECTIONS OF DOCUMENTS

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

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

7. AGGREGATION WITH INDEPENDENT WORKS

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

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

8. TRANSLATION

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

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

9. TERMINATION

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

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

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

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

10. FUTURE REVISIONS OF THIS LICENSE

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

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

11. RELICENSING

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

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

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

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

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

# ADDENDUM: How to use this License for your documents

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

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

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

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

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

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

# Index

# P

# R

# S

# T

# V

# W