pyconfigure

for version 0.2.1, 21 August 2013

bug-pyconfigure@gnu.org

This manual is for pyconfigure (version 0.2.1, updated 21 August 2013).

Copyright © 2012, 2013 Brandon Invergo

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

Table of Contents

1 Introduction **1**
1.1 Configuring Python packages 1

2 Installation **2**

3 Invoking pyconf **3**
3.1 PKG-INFO metadata 3

4 Existing projects **5**

5 Customization **6**
5.1 configure.ac 6
5.1.1 Required macros 6
5.1.2 Verifying the Python version 7
5.1.3 Checking for a module or function 7
5.1.4 Writing test programs 8
5.1.5 Using Sphinxbuild to build documentation 8
5.2 Makefile.in 8
5.2.1 Makefile.in (distutils) 9
5.2.2 Makefile.in (Make) 9
5.3 setup.py.in 10

6 Appendix **11**
6.1 Autoconf macros 11

Appendix A GNU Free Documentation License **13**

1 Introduction

Python packages typically are configured and installed through the use of the `distutils` module or one of its derivatives. The user performs necessary actions via a Python script called `setup.py`. For simple programs, this is straight-forward. However, for more complex software packages, especially for those which also include code in other languages such as C or Fortran, the limitations of the `distutils` method quickly become apparent.

The configuration and installation of GNU software and many other programs, on the other hand, is done according to the use of standard `configure` scripts and Make recipes. This method has the advantage of being language-agnostic, very flexible, and time-proven. pyconfigure consists of all the files necessary to begin using the standard GNU build process to configure and install a Python package.

1.1 Configuring Python packages

Configuring and installing Python packages which use pyconfigure follows the familiar steps of all standard GNU software:

```
$ ./configure
$ make
$ make install
```

As usual, the user may pass arguments to `configure` in order to specify how she wants the software to be installed. By default, the generated `configure` script takes the following useful arguments, among others:

Argument	**Description**
`--prefix`	Set the root directory in which to install files (default=/usr/local)
`--with-virtualenv`	Install to a virtualenv at `$prefix`
`PYTHON`	Path to the Python interpreter to use
`PYTHONPATH`	The PYTHONPATH to use during the installation

However, as the developer is expected to customize these files, the final `configure` script may take many more arguments. The developer is expected to provide proper documentation in this case.

2 Installation

Pyconfigure includes the template files that you will use in your projects, the `pyconf` script to copy those files into a project's directory, and this documentation. In order for their usage to be convenient, it is recommended to install them. Installation of pyconfigure follows the standard GNU installation procedure. Upon unpacking the source, navigate into its directory and run the following command sequence:

```
$ ./configure --prefix=/usr/local
$ make install
```

If you wish the files to be installed to a different location, specify it using the `--prefix` option.

3 Invoking pyconf

Before invoking the `pyconf` script, you first must decide whether you would prefer to have your installation logic written in Python or in Make. If you choose the former, the generated Makefile will be a wrapper around the Python installation script (i.e. `setup.py`), while if you choose the latter, the Python installation script will be a wrapper around the Makefile.

Next, you must create a `PKG-INFO` file containing standard metadata about your project (see Section 3.1 [PKG-INFO metadata], page 3). Finally, in the most basic case, you would navigate to your project's directory and simply invoke `pyconf` on your project's `PKG-INFO` file:

```
$ pyconf PKG-INFO
```

This will generate a `configure.ac` Autoconf file, a `configure` script generated from that Autoconf file, a `setup.py.in` installation file (to be configured by the user upon the invocation of `configure`) and a `Makefile.in` file which wraps the functionality of `setup.py`. If any of these files already exist, `pyconf` will not overwrite them unless the `--overwrite` option is passed.

If you wish the files to be copied into a different directory, you may add the `--output` option (or its short form `-o`) to specify the directory into which you would prefer the files to be copied.

```
$ pyconf -output=$HOME/Projects/pyproject PKG-INFO
```

If you would prefer to write your installation logic using Make, pass the `--prefer-make` (`-m`) option:

```
$ pyconf --prefer-make PKG-INFO
```

Now, the `setup.py.in` script that is generated will instead be a wrapper around the `Makefile.in` file. You would then extend the installation process in the latter file.

If you would prefer a pure-Python approach, pyconf may optionally not generate any Makefile by passing the `--no-make` option. Finally, if you only need pyconfigure's Autoconf macros, you may pass the `--macros-only` option, which causes `pyconf` to exit immediately after copying the macros into your package directory.

3.1 PKG-INFO metadata

As a base, the `pyconf` script requires a `PKG-INFO` file containing metadata about the project. This file should fit the requirements of the `PKG-INFO` metadata file format as outlined in the PEP 345 document. The file consists of several `Key: value` pairs. Some keys may be specified more than once, meaning that the package has several such values, while others may appear only once. Refer to PEP 345 for the authoritative specification.

For the purposes of pyconfigure, only four keys are required. The first, "Metadata-Version" must have a value of 1.2 or higher; earlier metadata specification versions are not supported. "Name", which may only appear once, contains the package's name. Similarly, "Version" contains the package version number. Finally, "Author-email" contains the principal email address for the project. Other keys are required to fully meet the PEP 345 specification; refer to that document for more information.

Here is a minimal example required to get started:

```
Metadata-Version: 1.2
Name: foo
Version: 1.5
Author-email: bug-foo@gnu.org
```

4 Existing projects

Using pyconfigure with existing projects is easy. For example, if your project already has a `setup.py` script, there is no need to replace it with pyconfigure. In this case, the best way to proceed would be to run `pyconf` to copy all of the files into your project's directory. Next, you simply need to copy the contents of your `setup.py` script into `setup.py.in`. Be sure not to just overwrite the file directly! Inside `setup.py.in` you will see several strings like `@PACKAGE_NAME@`. These are strings that will be replaced by the configure script and they should remain as they are. Most of the contents of the standard `setup` function should have already been filled in through the information in the `PKG-INFO` file but if not, they can be filled in manually. The default `setup.py.in` script is otherwise very simple, meaning any extensions to it that you have written in your `setup.py` script can simply be copied in.

If your project does not yet have a `setup.py` script but it already has a `Makefile`, the process is even easier. Simply call `pyconf` with `--prefer-make` and the `setup.py.in` file that is generated in your project's directory will simply wrap your `Makefile` (just be sure not to pass the `--overwrite` option!).

5 Customization

Once `pyconf` has generated the files in your project's directory, you should customize them to meet your project's needs.

In particular, you will want to customize `configure.ac` and `Makefile.in` or `setup.py.in`. `configure.ac` contains a series of macros which are used by Autoconf to build a portable `configure` shell script. This script either guesses important system settings or is provided them by the user. When the user invokes `configure`, it uses `Makefile.in` and `setup.py.in` as templates to create the Make recipe `Makefile` and the Python setup script `setup.py`.

5.1 configure.ac

There are some minimum modifications that should be made in `configure.ac`. The file contains a significant amount of information in the form of comments, so it is possible to discern your needs while editing. For more advanced usage, it is recommended to refer to the See Info file `autoconf`, node '`Autoconf`'.

In this file you will see a macro called `AC_INIT`. This is a standard Autoconf macro. The arguments to this are automatically generated from the `PKG-INFO` file that you used. These three values are used extensively in the files modified by the configure script, so it is important that they be correct.

Further down, you will also find a macro called `PC_INIT`. This is the core macro of pyconfigure. This will build the code necessary to find a suitable Python interpreter on the user's computer. To that end, you can pass arguments to this macro which specify the minimum and/or maximum supported Python versions.

While the default `configure.ac` script will likely be sufficient for a basic Python-based project, it may be made to be much more powerful for packages with more complex needs. To that end, several Autoconf macros are provided in the file `m4/python.m4` to allow the developer to write robust tests See Section 6.1 [Autoconf macros], page 11. Note that when you distribute your software, you must include this directory and file with your distribution if you also distribute your `configure.ac` file.

Once you modify your `configure.ac` to your liking, you must regenerate your `configure` script with the `bootstrap.sh` script that is generated by pyconfigure.

```
$ ./bootstrap.sh
```

A full explanation of the general use of Autoconf macros is beyond the scope of this document, however it is worth presenting some examples.

5.1.1 Required macros

Several macros are required in `configure.ac` to use pyconfigure. These are:

```
AC_INIT([project_name], [project_version], [project-email])
```

This initializes Autoconf and also substitutes your project name and version in any output that it generates. The initial argument values are automatically generated by pyconfigure when you first run the `pyconf` script. Note that the arguments are surrounded by braces in all cases. This is to prevent M4 from trying to expand the arguments using whatever macros it knows.

```
AC_CONFIG_MACRO_DIR([m4])
```

This macro imports all of the Python Autoconf macros. If you choose to write your own macros for other purposes, you should include them in the `m4` directory as well.

```
PC_INIT([2.5], [3.3.1])
```

This is the key macro. It finds a Python interpreter available on the system that meets optional version requirements specified in its arguments and saves its path in the `PYTHON` variable. Generally speaking, the highest-version Python interpreter found within the given version range (inclusive) will be used. Note, however, that minor version differences may cause discrepancies. For example, the user may have Python 3.3.1 installed but a slight difference in its release may cause the interpreter to internally report a slightly higher version, causing this interpreter to not pass the version check. To be safe, set the maximum version one bugfix release higher (i.e. "3.3.2" in this case).

```
PC_PYTHON_SITE_PACKAGE_DIR
PC_PYTHON_EXEC_PACKAGE_DIR
```

These two macros figure out where Python expects packages to be installed (i.e. `/usr/lib/python2.7/site-packages/`) and saves them in the variables `pkgpythondir` and `pkgpyexecdir`, respectively, for use in `Makefile.in`. These macros are only required if you will be writing your installation logic in Make.

5.1.2 Verifying the Python version

As described in the previous section, `PC_INIT` finds the Python interpreter with the highest version that meets the provided requirements. You may wish to perform other tests on the version number yourself. There is a macro available to simplify this, `PC_PYTHON_VERIFY_VERSION` (indeed, `PC_INIT` uses this macro internally).

```
m4_define(python_min_ver, 2.6.1)
PC_PYTHON_VERIFY_VERSION([>=], python_min_ver, ,
                         [AC_MSG_ERROR(Python interpreter too old)])
```

In this example, we set the minimum version to 2.6.1 through the use of an M4 macro. We then check if the interpreter stored in the `PYTHON` variable (either set by the user or found by `PC_INIT`) is at least of that version. If it is not, the resulting `configure` script will exit with an appropriate error message. You may use any mathematical comparison operator that Python recognizes for the first argument ("==", "<=", ">", etc.).

5.1.3 Checking for a module or function

It's reasonable to assume that many Python packages will have dependencies on other, external modules. With the provided pyconfigure macros, it is simple to check for the presence of dependencies on the system. All you have to do is use the `PC_PYTHON_CHECK_MODULE` macro as follows:

```
PC_PYTHON_CHECK_MODULE([foo])
```

In this example, we checked for the presence of a module "foo."

If the module is a hard requirement, you may provide actions to do if it is not present:

```
PC_PYTHON_CHECK_MODULE([foo], , AC_MSG_ERROR([Module foo is not installed]))
```

If you need more fine-grained control, you can also test for a specific function, for example `foo.bar(arg1, arg2)`:

```
PC_PYTHON_CHECK_FUNC([foo], [bar], [arg1, arg2])
```

Remember that you may omit arguments to Autoconf macros: in the previous example, the final two arguments, which correspond to the action to take if the test is successful and if it fails simply are not present in the argument list. Similarly, if you do not need to pass arguments to the test function, you can entirely omit the third argument to the macro:

```
PC_PYTHON_CHECK_FUNC([foo], [bar])
```

5.1.4 Writing test programs

One great benefit of Autoconf is the ability to embed test programs inside `configure`. The pyconfigure macros allow for this by defining Python as a language within Autoconf. You then would proceed to write test programs as you would in any other language that Autoconf supports like C.

```
AC_LANG_PUSH(Python)[]
AC_RUN_IFELSE([AC_LANG_PROGRAM([dnl
# some code here
import foo
], [dnl
    # some more code here
    foo.bar()
])], [ACTION-IF-SUCCESSFUL], [ACTION-IF-FAILED])
AC_LANG_POP(Python)[]
```

The first argument to `AC_LANG_PROGRAM` is the so-called "prolog", and typically will contain your `import` statements or function definitions. The second argument contains the main body of the program, which will be in the scope of an `if __name__=="__main__":` block. So, you must be sure to indent the code appropriately.

5.1.5 Using Sphinxbuild to build documentation

Using pyconfigure and Autoconf to test for other tools is quite easy. For example, many Python packages use Sphinxbuild to build their documentation. If this is the case for your project, you might do something like the following:

```
AC_CHECK_PROGS([SPHINXBUILD], [sphinx-build sphinx-build3 sphinx-build2], [no])
AS_IF([test "x$SPHINXBUILD" = xno],
    AC_MSG_WARN(sphinx-build is required to build documentation))
```

We simply use Autoconf's `AC_CHECK_PROGS` macro to check for a series of possible Sphinxbuild binaries and save the result to the SPHINXBUILD variable, which may then be used in `Makefile.in`:

```
docs/build/index.html: $(wildcard $(srcdir)/docs/source/*)
ifneq ($(SPHINXBUILD),no)
  $(SPHINXBUILD) -b html docs/source/ docs/build/
endif
```

5.2 Makefile.in

How you will customize the file `Makefile.in` and, indeed, what you will find in the file when it is first generated both depend on whether you specified if you prefer to write your installation logic in Make See Chapter 3 [Invoking pyconf], page 3.

5.2.1 Makefile.in (distutils)

If you did not specify `--prefer-make`, `Makefile.in` will be a wrapper around the functionality of the Python `setup.py` script. For a basic program, no great amount of customization of this file will be necessary. The file contains many comments, which introduce its various sections.

By default, the file supports installing to a Virtualenv, depending on whether the user has specified to do so when running `configure`. You will likely not have to change the "install" recipe. If you have other files to install, it is recommended to create new targets to install them, and to add those targets as prerequisites to the "install" target. For example, if you have extra data files to install, you might create a "install-data" target and corresponding recipe, and then add "install-data" as a prerequisite to "install":

```
install: installdirs install-data
```

If you do install more files, be sure that they are properly removed when the user runs `make uninstall` by modifying the recipe for the "uninstall" target. Note that, at this time, Python's `distutils` does not have its own "uninstall" target, so this must be done manually.

If you intend to produce source distributions via the `Makefile`, which is more flexible than doing so via `setup.py`, it is important to modify the `DIST_FILES` variable located near the top of `Makefile.in`. Any file or directory you list there will be included in your source distribution.

Finally, you may write recipes to build your package's documentation, which may not be covered by your `setup.py` script. How you accomplish this is highly dependent upon how you have organized your documentation sources. One example of how you might do it is included in the `Makefile.in`, commented-out at the end.

5.2.2 Makefile.in (Make)

If you passed the option `--prefer-make` to `pyconf`, `Makefile.in` will contain all of the installation logic for your package. It is highly recommended that you be familiar with basic Make usage. See the See Info file `make`, node '`Make`'.

By default, `Makefile.in` will contain the logic necessary to install a basic Python package consisting of one or more modules. The primary customization may be performed via the variables found at the beginning of the file: `PYPACKAGES`, `PYPACKAGE_ROOT`, `SCRIPTS`, `PKG_DATA`, `DATA`, and `DATA_ROOT`.

`PYPACKAGES` should contain a space-separated list of all of the Python modules in your package (i.e. top-level directories containing a `__init__.py` file). When your package is installed to the user's computer, these modules will be stored in the Python package directory (generally `$prefix/lib/python$version/{site,dist}-packages/`). If the modules are contained in a sub-directory, say `src`, of your source directory, you may set the `PYPACKAGE_ROOT` variable to that directory.

```
PYPACKAGES = foo bar
PYPACKAGE_ROOT = src
```

In this example, there are two modules to install: "foo" and "bar". The modules are to be found under the `src` directory; thus, for example, module "foo" is to be found at `src/foo`.

The directories listed under `PYPACKAGES` will only have their Python files installed. If your modules depend on other, non-Python data files, you may list these under the `PKG_DATA` variable. Data files should be listed relative to their parent module. Thus, if module "foo" contains a file called `bar.dat`, set `PKG_DATA = foo/bar.dat`.

Other data files, which are not specific to any of the Python modules, may be specified under the `DATA` variable. As before, if your data files are all stored under a particular sub-directory, you may specify it in `DATA_ROOT`. Files listed under `DATA` are installed to the package's data directory, which is typically `/usr/local/share/$package`).

Finally, if your package has any scripts to install, list them under the `SCRIPTS` variable. They should be listed as files relative to the directory containing `Makefile.in`. Thus, if your script `baz` is located in the sub-directory `bin`, you would set `SCRIPTS = bin/baz`.

One particular advantage of writing the installation logic in Make is the ease with which you may work with non-Python code in your project, such as extensions written in C. How these recipes are to be written is dependent upon the build requirements of this code, and you are thus referred to the See Info file `make`, node '`Make`'. Any installation recipes should be given their own targets and made as prerequisites of the "install" target.

5.3 setup.py.in

`pyconf` will automatically generate a `setup.py.in` file, to be configured by the `configure` script to produce the Python `setup.py` script. If the `--prefer-make` option was specified, this file will merely contain Python code which calls Make on the generated `Makefile`, and needs not to be modified. Otherwise, the file will contain basic Python code to use `distutils` for package installation. The reader is referred to the Python documentation for more information on how to customize this file.

6 Appendix

6.1 Autoconf macros

Macro Name & Arguments	Description	Variables exported
PC_INIT([MINIMUM-VERSION], [MAXIMUM-VERSION])	Initialize pyconfigure by finding the highest-version Python interpreter that meets the specified requirements. If no such interpreter is found, exit with an error. This is a convenience macro that includes PC_PROG_PYTHON and does the version checking via PC_PYTHON_VERIFY_VERSION.	PYTHON
PC_PROG_PYTHON([NAME-TO-CHECK], [MINIMUM-VERSION], [MAXIMUM-VERSION])	Find a Python interpreter with the highest version number between the given minimum and maximum versions. The version requirement is performed in a naive way, by simply appending the major and minor release numbers to the interpreter name (i.e. "python2.7").	PYTHON
PC_PROG_PYTHON_CONFIG([NAME-TO-CHECK])	Find a python-config program	PYTHON_CONFIG
PC_PYTHON_VERIFY_VERSION([OPERATOR], [VERSION], [ACTION-IF-TRUE], [ACTION-IF-NOT-TRUE])	Verify that the Python interpreter is of a sufficient version number according to some comparison operator ("==", "<=", etc.)	
PC_PYTHON_CHECK_VERSION	Get the version of the Python interpreter	PYTHON_VERSION
PC_PYTHON_CHECK_PREFIX	Check what Python thinks is the prefix	PYTHON_PREFIX
PC_PYTHON_CHECK_EXEC_PREFIX	Check what Python thinks is the exec_prefix	PYTHON_EXEC_PREFIX
PC_PYTHON_CHECK_INCLUDES	Check the include flags ('-I[header]...') for including the Python header files	PYTHON_INCLUDES

PC_PYTHON_CHECK_HEADERS	Check for the Python header files (i.e. Python.h)	HAVE_PYTHON_H
PC_PYTHON_CHECK_LIBS	Check for the proper LIBS flags to load the Python shared libraries	PYTHON_LIBS
PC_PYTHON_TEST_LIBS	Test for the presence of the Python shared libraries	HAVE_LIBPYTHON
PC_PYTHON_CHECK_CFLAGS	Find the CFLAGS that Python expects	PYTHON_CFLAGS
PC_PYTHON_CHECK_LDFLAGS	Find the LDFLAGS that Python expects	PYTHON_LDFLAGS
PC_PYTHON_CHECK_EXTENSION_SUFFIX	Check the extension suffix given to Python extension modules (Python 3 only)	PYTHON_EXTENSION_SUFFIX
PC_PYTHON_CHECK_ABI_FLAGS	Check the ABI flags used by Python (Python 3 only)	PC_PYTHON_ABI_FLAGS
PC_PYTHON_CHECK_PLATFORM	Check what platform Python thinks this is PYTHON_PLATFORM	
PC_PYTHON_CHECK_SITE_DIR	Check the appropriate place to install Python packages (i.e. $(prefix)/lib/python2.7/site-packages)	pythondir
PC_PYTHON_SITE_PACKAGE_DIR	A convenience macro; adds the package's name to pythondir	pkgpythondir
PC_PYTHON_CHECK_EXEC_DIR	Check directory for installing Python extension modules	pyexecdir
PC_PYTHON_EXEC_PACKAGE_DIR	A convenience macro; adds the package's name to pyexecdir	pkgpyexecdir
PC_PYTHON_CHECK_MODULE	Test if a given Python module can be successfully loaded	
PC_PYTHON_CHECK_FUNC	Test if a given Python function can be called successfully.	

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

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

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

1. APPLICABILITY AND DEFINITIONS

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

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

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

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

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

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

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

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

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

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

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

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

2. VERBATIM COPYING

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

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

3. COPYING IN QUANTITY

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

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

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

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

4. MODIFICATIONS

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

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

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

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

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

D. Preserve all the copyright notices of the Document.

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

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

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

H. Include an unaltered copy of this License.

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

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

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

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

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

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

O. Preserve any Warranty Disclaimers.

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

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

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

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

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

5. COMBINING DOCUMENTS

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

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

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

6. COLLECTIONS OF DOCUMENTS

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

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

7. AGGREGATION WITH INDEPENDENT WORKS

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

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

8. TRANSLATION

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

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

9. TERMINATION

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

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

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

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

10. FUTURE REVISIONS OF THIS LICENSE

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

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

11. RELICENSING

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

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

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

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

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

ADDENDUM: How to use this License for your documents

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

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

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

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

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

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