Embench™ User Guide

Authors: Embench™ Task Group Issue: 1.0

Table of Contents

  - Collections of documents
  - Aggregation with independent works
  - Translation
  - Termination
  - Future revisions of this license
  - Addendum: how to use this License for your documents

About Embench

Embench is a suite of free and open source C benchmarks, all of which are small real-world programs suitable for running on deeply embedded systems with at least 64kB of ROM and 64kB of RAM.

Embench is based on the following principles:

1. **Embench must be free:**

- if behind a paywall, it is not going to be as widely used; and
- academics publish frequently and can promote the use of benchmarks, so free and open accelerates their adoption and hence publicizes benchmarks to the rest of the community.

2. **Embench must be easy to port and run:**

- if very difficult or expensive to port and run, then it will not be as widely used; and
- Linpack, Dhrystone, and CoreMark don't have good reputations yet are widely reported presumably because they are free and easy to port and run.

3. **Embench must be a suite of real programs:**

- real programs are much likely to be representative than synthetic programs;
- a realistic workload is easier to represent as a suite of programs than as a single program;
- compilers and hardware change rapidly, so benchmarks need to evolve over time to deprecate pieces that are newly irrelevant and add missing features that speak to current challenge;
- a single program is hard to evolve; and
- a suite of around 20 real programs means a more realistic workload that is much easier to evolve.

4. **Embench must have a supporting organization that maintains its relevance over time:**

- need an organization to evolve the suite over time;
- goal of refreshes every two years; and
- set up inside an existing organization, the Free and Open Source Silicon Foundation (FOSSi), rather than creating a new organization.

5. **Embench must report a single summarizing performance score:**

- if no official single performance score, then it won't be as widely reported;
- even worse, others may supply unofficial summaries that are either misleading or conflicting;
- might include orthogonal measures, like code size or power – each would have a summarizing score.

6. **Embench should use geometric mean and standard deviation to summarize:**

- recommend first calculating the ratios of performance relative to a reference platform; and
- report geometric mean (GM) of ratios + geometric standard deviation (GSD).

7. **Embench must involve both academia and industry:**

- academia helps maintain fair benchmarks for the whole field;
- industry helps ensure that programs genuinely important in the real world; and
- we need both communities participating to succeed.

The Bristol/Embecosm Embedded Benchmark Suite (BEEBS)

The benchmarks are largely derived from BEEBS, which in turn draws its material from various earlier projects, notably:

- MiBench;
- the WCET benchmark collection;
- DSPstone;
- the Simple Generic Library; and
- the Nettle low-level cryptographic library.

For the reasoning behind the choice of benchmarks in BEEBS, see BEEBS: Open Benchmarks for Energy Measurements on Embedded Platforms by J Pallister, S Hollis and J Bennett . For an example of BEEBS in use, see Identifying Compiler Options to Minimise Energy Consumption for Embedded Platforms by J Pallister, S Hollis and J Bennett.

Future work

This is a work in progress. This first stable release 0.5 was published at the Embedded World Conference in February 2020. It is intended to allow the wider community to explore a stable version of the platform.

Based on feedback from this exercise, it is anticipated that the first full release will be produced by the end of 2020.

Feedback and how to contribute

Comments on this document should be made through the Embench mailing list. Proposed changes may be submitted as git pull requests.

You are encouraged to contribute to this repository by submitting pull requests and by commenting on pull requests submitted by other people.

Note. Don't forget to add your own name to the list of contributors in the document.

Contributors

This document has been created by the following people (in alphabetical order of surname).

- Jeremy Bennett.

Document history

Revision	*Date*	*Author(s)*	*Modification*
WIP	9 Aug 20	Roger Shepherd	Note re python module names
WIP	19 Jul 20	Jeremy Bennett	Add pyelftools to prerequisites.
0.5	27 Feb 20	David Patterson Jon Taylor Jeremy Bennett	Incorporate latest review comments to accompany release 0.5. Update document versioning for consistency with release numbering.
	21 Jan 20	Lyell Read	Correcting partial sentence and removing duplicate log file flag definition.
0.5 rc2	13 Nov 19	Jeremy Bennett	Migrated to markdown and updated to refer to Python scripts. Version previously known as draft 0.6.
0.5 rc1	6 Jun 19	Jeremy Bennett	First draft of this document, drawing heavily on the BEEBS documentation. Version previously known as draft 0.5.

Building and running Embench

The benchmarks have to be first compiled, then they can then be measured on the target.

Prerequisites

Embench expects the following version of tools. Update your system accordingly.

Components	*Version*
python	3.6 or later

The following non-standard Python packages are needed.

Package	*Comments*
pyelftools	

Preparation

Unpack the software. Either extract from the supplied *tar* file:

```
tar xf embench-VERSION.tar.bz2
```

or clone this git repository:

```
git clone https://github.com/embench/embench-iot.git
```

Configuring the benchmarks

The benchmarks are configured in several ways:

- default values in the build script
- an architecture specific configuration file
  - found in `config/<architecture>/arch.cfg`
  - for example `config/riscv32/arch.cfg`;
- a chip specific configuration file
  - found in `config/<architecture>/chips/<chip>/chip.cfg`
  - for example `config/riscv32/chips/speed-test/chip.cfg`;
- a board specific configuration file
  - found in `config/<architecture>/boards/<board>/board.cfg`
  - for example `config/riscv32/boards/ri5cyverilator/board.cfg`; and
- on the command line to the build script.

Any number of these may be specified, they may duplicate specification, in which case the configuration latest in the list takes precedence, or in the case of flags, the flags are appended to the list, so any later flags take precedence.

In addition there may be chip and board specific heads and code:

- a chip specific header in `config/<architecture>/chips/<chip>/chipsupport.h`
  - for example `config/riscv32/chips/speed-test/chipsupport.h`;
- chip specific code in `config/<architecture>/chips/<chip>/chipsupport.c`
  - for example `config/riscv32/chips/speed-test/chipsupport.c`;
- a board specific header in `config/<architecture>/boards/<board>/boardsupport.h`
  - for example `config/riscv32/boards/ri5cyverilator/boardsupport.h`; and
- board specific code in `config/<architecture>/boards/<board>/boardsupport.c`
  - for example `config/riscv32/boards/ri5cyverilator/boardsupport.c`.

The configuration files (architecture, chip, board) take the form of assignments to python variables. The following parameters may be set.

- `cc`: The C compiler to use. Default value `cc`.
- `ld`: The linker to use. Default value is be the same value as was set for `cc`, which means with most modern compilers there is no need to set `ld`, since the compiler can act as a linker driver.
- `cflags`: A Python list of additional compiler flags, which are appended to any other compiler flags. Default is the empty list.
- `ldflags`: A Python list of additional linker flags, which are appended to any other linker flags. Default is the empty list.
- `cc-define1-pattern`: A Python formatted string pattern with positional arguments to be used when defining a constant on the compiler command line. Default value `-D{0}`.
- `cc-define2-pattern`: A Python formatted string pattern with positional arguments to be used when defining a constant to a specific value on the compiler command line. Default value `-D{0}={1}`.
- `cc-incdir-pattern`: A Python formatted string pattern with positional arguments to be used when specifying an include directory on the compiler command line. Default value `-I{0}`.
- `cc-input-pattern`: A Python formatted string pattern with positional arguments to be used when specifying the input file on the compiler command line. Default value `{0}`.
- `cc-output-pattern`: A Python formatted string pattern with positional arguments to be used when specifying the output file on the compiler command line. Default value `-o {0}`.
- `ld-input-pattern`: A Python formatted string pattern with positional arguments to be used when specifying the input file on the linker command line. Default value `{0}`.
- `ld-output-pattern`: A Python formatted string pattern with positional arguments to be used when specifying the output file on the linker command line. Default value `-o {0}`.

- `user-libs`: A list of libraries to be appended to the linker command line. The libraries may be absolute file names or arguments to the linker. In the latter case corresponding arguments in `ldflags` may be needed. For example with GCC or Clang/LLVM if `-l` flags are used in `user_libs`, then `-L` flags may be needed in `ldflags`. Default value is the empty list.
- `dummy-libs`: A list of dummy libraries to be used (for example if system libraries have been disabled through options in `ldflags`). Dummy libraries have their source in the `support` subdirectory. Thus if `crt0` is specified, there should be a source file `dummy-crt0.c` in the `support` directory. Default value is the empty list.
- `cpu-mhz`: The clock rate of the target in MHz. Default value 1.
- `warmup-heat`: How many times the benchmark code should be run to warm up the caches. Default value 1.
- `timeout`: The maximum time (in seconds) allowed for the compiler or the linker to run for each invocation. Default value 5.

Any other variables are silently ignored. There is no need to set an unused parameter, and any configuration file may be empty or missing if no flags need to be set.

The board and chip specific files are used only to provide code essential to the functionality. Chip support files are usually empty. The board support header (`boardsupport.h`) is typically used to define the clock rate of the board, for example:

```
#define CPU_MHZ 1
```

The board support code file (`boardsupport.c`) is used to define three functions.

- `void initialise_board ()`: Called to set up the board;
- `void start_trigger ()`: Called when we start timing the benchmark, to start any board specific timing mechanism.
- `void stop_trigger ()`: Called when we stop timing the benchmark, to stop any board specific timing mechanism.

Building the benchmarks

Embench is built with the `build_all.py` script, which takes the following arguments.

- `--builddir`: The programs are build out of tree, this specifies the directory in which to build. It may be an absolute or relative directory name; if the latter, it will be relative to the top level directory of the repository. Default value `bd`.
- `--logdir`: A log file is created with detailed information about the build. This specifies the directory in which to place the log file. It may be an absolute or relative directory name; if the latter, it will be relative to the top level directory of the repository. Default value `logs`.

- `--arch`: This mandatory argument specifies the architecture for which the benchmarks are to be built. It corresponds to a directory name in the main `config` directory.
- `--chip`: This mandatory argument specifies the chip being used and corresponds to a directory within the `chips` subdirectory of the architecture configuration directory.
- `--board`: This mandatory argument specifies the board being used and corresponds to a directory within the `boards` subdirectory of the architecture configuration directory.
- `--cc`: The C compiler to be used. Default value `cc`.
- `--ld`: The linker to be used. Default value the same value as for `--cc`
- `--cflags`: A space separated list of additional C flags to be appended to the compiler flags. Default value empty.
- `--ldflags`: A space separated list of additional linker flags to be appended to the linker flags. Default value empty.
- `--cc-define1-pattern`: A Python formatted string pattern with positional arguments to be used when defining a constant on the compiler command line. Default value `-D{0}`.
- `--cc-define2-pattern`: A Python formatted string pattern with positional arguments to be used when defining a constant to a specific value on the compiler command line. Default value `-D{0}={1}`.
- `--cc-incdir-pattern`: A Python formatted string pattern with positional arguments to be used when specifying an include directory on the compiler command line. Default value `-I{0}`.
- `--cc-input-pattern`: A Python formatted string pattern with positional arguments to be used when specifying the input file on the compiler command line. Default value `{0}`.
- `--cc-output-pattern`: A Python formatted string pattern with positional arguments to be used when specifying the output file on the compiler command line. Default value `-o {0}`.
- `--ld-input-pattern`: A Python formatted string pattern with positional arguments to be used when specifying the input file on the linker command line. Default value `{0}`.
- `--ld-output-pattern`: A Python formatted string pattern with positional arguments to be used when specifying the output file on the linker command line. Default value `-o {0}`.
- `--user-libs`: A space separated list of libraries to be appended to the linker command line. The libraries may be absolute file names or arguments to the linker. In the latter case corresponding arguments in `--ldflags` may be needed. For example with GCC or Clang/LLVM if `-l` flags are used in `--user_libs`, then `-L` flags may be needed in `--ldflags`. Default value empty.
- `--dummy-libs`: A space separated list of dummy libraries to be used (for example if system libraries have been disabled through options in `--ldflags`). Dummy libraries have their source in the `support` subdirectory. Thus if `crt0` is specified, there should be a source file `dummy-crt0.c`

in the `support` directory. Default value empty.

- `--cpu-mhz`: The clock rate of the target in MHz. Default value 1.
- `--warmup-heat`: How many times the benchmark code should be run to warm up the caches. Default value 1.
- `--timeout`: The maximum time (in seconds) allowed for the compiler or the linker to run for each invocation. Default value 5.
- `--clean`: Delete all intermediaries and final files from any previous runs of the script.
- `--help`: Provide help on the arguments.

Running the benchmark of code size

Benchmarking code size uses the `benchmark_size.py` script, which takes the following arguments.

- `--format`: Specifies the file format of executable executable and/or object files. The option are `elf` and `macho`. Default value `elf`.
- `--builddir`: The programs are build out of tree, this specifies the directory in which the programs were built. It may be an absolute or relative directory name; if the latter, it will be relative to the top level directory of the repository. Default value `bd`.
- `--logdir`: A log file is created with detailed information about the benchmark run. This specifies the directory in which to place the log file. It may be an absolute or relative directory name; if the latter, it will be relative to the top level directory of the repository. Default value `logs`.
- `--baselinedir <dir>`: Specifies the directory in which reference size data can be found. May be an absolute or relative directory name. If it is relative then it will be interpreted as relative to the top level directory of the repository. The default value is `baseline-data`, and size data will be sourced from `baseline-data/size.json`.
- `--relative` or `--absolute`: If `--relative` is specified, present benchmark results relative to the baseline architecture. If `--absolute` is specified, present absolute benchmark results. If neither is specified, present relative results, because this is the defined norm for Embench.
- `--text`: A space separated list of sections containing code. Default value for elf format files `.text`; for macho format files `__text`.
- `--data`: A space separated list of sections containing non-zero initialized writable data. Default value for elf format files `.data`; for macho format files `__data`. The option `--metric` with the respective section type needs to be used in order to have the sections added to the size calculation.
- `--rodata`: A space separated list of sections containing read only data. Default value for elf format files `.rodata`; for macho format files `__cstring __const`. The option `--metric` with the respective section type needs to be used in order to have the sections added to the size calculation.
- `--bss`: A space separated list of sections containing zero initialized data.

Default value for elf format files `.bss`, for macho format files `__bss`. The option `--metric` with the respective section type needs to be used in order to have the sections added to the size calculation.

- `--metric`. A space separated list of section types to include when calculating the benchmark metric. Any section listed with the options `--text`, `--data`, `--rodata` and `--bss` is included in the respective section type. Permitted values ares `text`, `data`, `rodata`, `bss`. Default value `text`.
- `--text-output`: Output the text in a plain text format. This is the default.
- `--json-output`: Output the results in json format, instead of the default plain text format.
- `--baseline-output`: Output results in a format suitable for use as baseline data instead of the default text format. This can be used instead of the reference data in `baseline-data/size.json`.
- `--help`: Provide help on the arguments.

Running the benchmark of code speed

Benchmark code speed uses the `benchmark_speed.py` script, which takes the following general arguments.

- `--builddir`: The programs are build out of tree, this specifies the directory in which the programs were built. It may be an absolute or relative directory name; if the latter, it will be relative to the top level directory of the repository. Default value `bd`.
- `--logdir`: A log file is created with detailed information about the build. This specifies the directory in which to place the log file. It may be an absolute or relative directory name; if the latter, it will be relative to the top level directory of the repository. Default value `logs`.
- `--baselinedir <dir>`: Specifies the directory in which reference speed data can be found. May be an absolute or relative directory name. If it is relative then it will be interpreted as relative to the top level directory of the repository. The default value is `baseline-data`, and speed data will be sourced from `baseline-data/speed.json`.
- `--relative` or `--absolute`: If `--relative` is specified, present benchmark results relative to the baseline architecture. If `--absolute` is specified, present absolute benchmark results. If neither is specified, present relative results, because this is the defined norm for Embench.
- `--text-output`: Output the text in a plain text format. This is the default.
- `--json-output`: Output the results in json format, instead of the default plain text format.
- `--baseline-output`: Output results in a format suitable for use as baseline data instead of the default text format. This can be used instead of the reference data in `baseline-data/speed.json`.

- --target-module <target module>: This mandatory argument specifies a python module in the `pylib` directory with definitions of routines to run the benchmark. Note that the argument specifies the name of module (e.g. `run_stm32f4-discovery`) not the name of the file that contains the module (e.g. `run_stm32f4-discovery.py`).
- `--timeout`: The maximum time (in seconds) allowed for each benchmark program to run. Default value 30.
- `--help`: Provide help on the arguments.

There is so much variation in how a benchmark can be run that the detailed implementation is left to a python module specified by `--target-module`. This module may define additional arguments. If this module has been specified when `--help` has also been specified, help will be provided on the target module's arguments.

Recording reliable results

For each benchmark run, you must record:

- details of the platform used, including its clock speed;
- if the platform is simulated or real. If simulated, an indication of accuracy must be stated (i.e., fully cycle accurate or cycle approximate);
- for a simulated platform, configuration options of the chip should be stated (e.g., branch predictor size, cache sizes, and so on);
- details of the chip on the platform, including its precise architecture variant;
- details of the compiler tool chain used, typically the version of each component and library, or for development tool chains the repository commit ID of each component;
- the compiler and linker flags used for the benchmarks, which should be the same for each benchmark program; and
- the version of Embench used.

For clarification compiler flags, whose effect is to vary the choice and parameters of optimization passes on a per program (or per compilation unit or function) basis are permitted. For example flags which use machine learning techniques to match source code styles with the a choice of optimization passes.

The philosophy of recorded data should be such that anyone else can take the same platform and toolchain and duplicate the results.

Statistics of computing benchmarks

These computations are carried out by the benchmark scripts.

Computing a benchmark value for speed

Carry out the following steps.

- For each benchmark record the time take to execute between `start_trigger` and `stop_trigger`, which should be a few seconds.
- This time should be recorded using hardware internal to the device being benchmarked - e.g., CPU cycle counter or other fast timer (i.e., running at a significantly higher rate than the benchmark takes to run).
- For each benchmark divide the time taken by the value used for `CPU_MHZ` in the configuration to give a normalized time value.
- For each benchmark, compute its speed relative to the reference platform - see Reference platform - by dividing the normalized time value of the reference benchmark by the normalized time calculated in the previous step.
- Calculate the geometric mean, geometric standard deviation and range of one geometric standard deviation of the relative speeds.

The benchmark value is the geometric mean of the relative speeds. A larger value means a faster platform. The range gives an indication of how much variability there is in this performance.

In addition the geometric mean may then be divided by the value used for CPU_MHZ, to yield an Embench score per MHz. This is an indication of the efficiency of the platform in carrying out computation.

Computing a benchmark value for code size

Benchmarks should be compiled with dummy versions of all standard libraries. Carry out the following steps:

- For each benchmark record the size of all `.text` sections.
- For each benchmark, compute its size relative to the reference platform - see Reference platform - by dividing the size recorded in the previous step by the size of the corresponding reference benchmark.
- Calculate the geometric mean, geometric standard deviation and range of one geometric standard deviation of the relative size.

The benchmark value is the geometric mean of the relative size. A larger value means code is larger. The range gives an indication of how much variability there is in this measurement.

NOTE The computation of the relative value is inverted compared to the computation for speed. This means that for size, small is good.

NOTE Older versions of the GNU *size* program report the size of `.text` + `.rodata` section. In measuring the size, ensure you use a modern version of *size* which supports the `-G` flag, which will yield the size of just `.text` sections.

Reference platform

The reference CPU is an Arm Cortex M4 processor without the floating point unit (FPU). The reference platform is a ST Microelectronics STM32F4 Discovery Board - see www.st.com/en/evaluation-tools/stm32f4discovery.html using its default clock speed of 16MHz. The processor on this board does contain a FPU, but this is disabled by use of appropriate compiler options (see below). The code is compiled using GCC 9.2.0, GNU binutils 2.33.1 and newlib 3.3.0. Newlib is configured for small code size - known as the "nanolib" configuration.

The board can be obtained directly from ST Microelectronics, using the above link, but is also widely available from other suppliers, including Farnell, RS Electronics, Mouser and several suppliers on Amazon.

For the speed benchmarks, the compiler flags used are:

```
-O2 -ffunction-sections -march=armv7-m -mcpu=cortex-m4 -mfloat-abi=soft -mthumb
```

For the size benchmarks, the compiler flags used are:

```
-Os -ffunction-sections -march=armv7-m -mcpu=cortex-m4 -mfloat-abi=soft -mthumb
```

In both cases the linker flags are:

```
-T<embench-iot-root>/config/arm/boards/stm32f4-discovery/STM32F407XG.ld
-O2 -Wl,-gc-sections -march=armv7-m -mcpu=cortex-m4 -mfloat-abi=soft -mthumb
-specs=nosys.specs
```

The directory `<embench-iot-root>` is the root directory of the `embench-iot` repository.

Speed is measured using the platform's internal cycle counter registers, which can be converted to time from knowledge of the platform's clock speed.

The reference results can be found in the files `baseline-data/speed.json` and `baseline-data/size.json` in the repository. For convenience the key values are recorded here:

Benchmark	*Speed*	*Size*
`aha-mont64`	4,004	1,072
`crc32`	4,010	284
`cubic`	3,931	1,584
`edn`	4,010	1,324
`huffbench`	4,120	1,242
`matmult-int`	3,985	492
`minver`	3,998	1,168
`nbody`	2,808	950
`nettle-aes`	4,026	2,148
`nettle-sha256`	3,997	3,396
`nsichneu`	4,001	11,968

Benchmark	*Speed*	*Size*
`picojpeg`	4,030	6,964
`qrduino`	4,253	5,814
`sglib-combined`	3,981	2,272
`slre`	4,010	2,200
`st`	4,080	1,000
`statemate`	4,001	4,484
`ud`	3,999	720
`wikisort`	2,779	4,296

NOTE Speed is measured in milliseconds, size is total size of all `.text` sections in bytes.

Documentation

The documentation is recorded in the file `doc/README.md` using GitHub flavored Markdown. Whilst very simple in the formatting it permits, this format is used for portability.

Building the documentation

The documentation can be viewed directly on GitHub - see `doc/README.md`. However if *hunspell*, *pandoc* and associated utilities are installed, then a *Makefile* is provided which can do the following on Linux like systems:

- spell check the document, listing any misspelled words (`make spell`);
- insert a table of comments into the document (`make toc`);
- build a HTML version of the document (`make html`);
- build a PDF version of the document (`make pdf`);

Any custom words for spell checking should be added to the file `custom.wordlist`.

Adding a New Board to Embench

Where to add files

If the board uses a completely new architecture, you will need to create a new subdirectory within the `config` directory.

```
cd config
mkdir ARCH
```

The architecture name comes from the first part of the host triplet (the `--host` configuration argument).

Within this *ARCH* directory create two separate directories for board and chip configurations

```
cd ARCH
mkdir boards
mkdir chips
```

If the architecture already has a board defined, these directories will already exist.

Then for your new board, create a directory in the `chips` directory for the chip it will use (if the directory does not already exist).

```
cd chips
mkdir CHIPNAME
```

The *CHIPNAME* corresponds to the argument given to `--with-chip` when configuring.

Similarly create a directory in the `board` directory for the new board. Since this is a new board, this directory will not already exist.

```
cd boards
mkdir BOARDNAME
```

The *BOARDNAME* corresponds to the argument given to `--with-board` when configuring.

Configuration files

Configuration data may be defined for the architecture, for the chip and for the board. These files are found respectively in

```
config/ARCH/arch.cfg
config/ARCH/boards/BOARDNAME/board.cfg
config/ARCH/chips/CHIPNAME/chip.cfg
```

The format and content of these files is described in Configuring the benchmarks.

Header files

There are two standard header files which may be defined:

```
config/ARCH/boards/BOARDNAME/boardsupport.h
config/ARCH/chips/CHIPNAME/chipsupport.h
```

These are combined into the general header `support.h` which is included by all benchmarks, and defines values which may be used by the benchmarks.

Board specific code that is to be linked in to the benchmarks should be defined in `config/ARCH/boards/BOARDNAME/boardsupport.c`. This file should define the following functions:

- `initialize_board` which is called to initialize the board;
- `start_trigger` which is called at the start of the test run; and
- `stop_trigger` which is called at the end of the test run.

It is usual for this file to include `support.h` to pick up any board and chip specific definitions that may prove useful.

Typically the tests are run using GDB and a remote GDB server to load the programs into a remote target. This can set breakpoint on `start_trigger` and `stop_trigger` to start and stop timing. In this case, these two function need no actual content, and the following is a sufficient implementation:

```
void
start_trigger ()
{
  __asm__ volatile ("" : : : "memory");
}

void
stop_trigger ()
{
  __asm__ volatile ("" : : : "memory");
}
```

By marking the inline assembly volatile and clobbering memory, we guarantee a function which will just contain a return statement.

The GNU Free Documentation License version 1.2, November 2002

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

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

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.

Applicability and definitions

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

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

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

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

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

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

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

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

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

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

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.

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.

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:

- 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.
- 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.
- State on the Title page the name of the publisher of the Modified Version, as the publisher.
- Preserve all the copyright notices of the Document.
- Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- 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.
- Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- Include an unaltered copy of this License.
- 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.
- 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.

- 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.
- 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.
- Delete any section Entitled “Endorsements”. Such a section may not be included in the Modified Version.
- Do not retitle any existing section to be Entitled “Endorsements” or to conflict in title with any Invariant Section.
- 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.

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."

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.

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.

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.

Termination

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

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 www.gnu.org/copyleft

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

Addendum: how to use this License for your documents

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

```
Copyright (C)  YEAR  YOUR NAME.

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

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