Scheme-GNUnet manual

Maxime Devos

Email: `maximedevos@telenet.be`

CHAPTER 1

INSTALLATION AND CONTRIBUTING GUIDE

1.1. Building from source

The latest 'official' development version of scheme-GNUnet can be found at https://notabug.org/maximed/scheme-gnunet. It can be downloaded with git. The following software needs to be installed first:

- The Autotools (autoconf and automake)
- GNU Guile (at least version 3)
- Purely Functional Data Structures in Scheme (pfds)
- (Guile) Fibers
- Guile-QuickCheck

For the benefit of dead tree readers, the invisible hyperlinks above are reproduced as visible URLs below.

- `https://www.gnu.org/software/autoconf/`
- `https://www.gnu.org/software/guile/`
- `https://github.com/ijp/pfds/`
- `https://github.com/wingo/fibers/`
- `https://ngyro.com/software/guile-quickcheck.html`

A few patches to guile and guile-fibers are required (some bug fixes, some extra functionality), see `guix.scm`.

Users of GNU Guix can run `guix environment -l guix.scm` in the checkout to create an environment where these dependencies are all present. Scheme-GNUnet uses the standard GNU build system, so to build Scheme-Gnunet, you only need to run

```
autoreconf -vif
./configure
make
make check
```

After building, the documentation is available at `doc/scheme-gnunet.pdf` and `doc/scheme-gnunet.html` in PDF and HTML formats. To get started, you can run the example mini-application at `examples/nse-web.scm` and point your browser at `http://localhost:8089`:

```
$ guile -L . -C . -l examples/nse-web.scm
```

1.1.1. Authenticating new source code

When GNU Guix is present, after pulling the latest Scheme-GNUnet commit, the following command can be run to verify it is authentic:

```
guix git authenticate 431f336edd51e1f0fe059a6f6f2d4c3e9267b7bc "C1F3
3EE2 0C52 8FDB 7DD7  011F 49E3 EE22 1917 25EE"
```

If it isn't authentic, an error message such as the following will be written:

```
Authenticating commits 54a74dc to 431f336 (1 new commits)...
[##########################################################]
guix git: error: commit 431f336edd51e1f0fe059a6f6f2d4c3e9267b7bc
not signed by an authorized key: C1F3 3EE2 0C52 8FDB 7DD7  011F 49E3
EE22 1917 25EE
```

1.2. Writing tests

'How SQLite Is Tested' is a recommended read. Scheme-GNUnet isn't that well-tested but still aims for being free of bugs and having many tests to prevents bugs from being introduced. When adding new code, consider writing test cases. Some things that can be tested and few methods for testing things:

- Run mutation tests. That is, replace in the source code `<` with `<=`, `0` with `1`, a variable reference `i` with a variable reference `j`, swap destination and source arguments … and verify whether the tests detect these little mutations.
- Be exhaustive. If a procedure handles both foos and bars, write test cases that pass the procedure a foo and test cases that pass the procedure a bar. Sometimes Guile-QuickCheck can help with generating many test cases if the input has a regular structure yet many edge cases, see e.g. `tests/cmsg.scm`.
- Verify exception mechanisms! If a procedure is expected to handle I/O errors, simulate I/O errors and end-of-files in all the wrong places. If the procedure can raise exceptions, make sure these exceptions are raised when necessary.

Tests are added in the directory `tests` and to the variable `SCM_TESTS` in `Makefile.am` and use `srfi :64`. To run the test suite, run `make check`.

1.3. Contact

Scheme-GNUnet is currently maintained on NotABug: `https://notabug.org/maximed/scheme-gnunet/`. Issues and pull requests can be reported and submitted here. Alternatively, for discussion about developing Scheme-GNUnet, you can send mails to gnunet-devel@gnu.org and for help about how to use Scheme-GNUnet, you can contact help-gnunet@gnu.org. These are public mailing lists, so don't send anything there you wouldn't mind the whole world to know.

[TODO: verify C GNUnet people are ok with this?]

For security-sensitive issues, you can send a mail directly to the maintainer, Maxime Devos <maximedevos@telenet.be>, optionally encrypted and signed with a GnuPG-compatible system. The maintainer's key fingerprint is C1F3 3EE2 0C52 8FDB 7DD7 011F 49E3 EE22 1917 25EE and a copy of the key can be downloaded from `https://notabug.org/maximed/things/raw/master/Maxime_Devos.pub`.

1.4. License

The code of Scheme-GNUnet is available under the Affero General Public License (AGPL), version 3 or later; see individual source files for details. The documentaton is available under the GNU Free Documentation License, see the start of this manual and the likewise-named appendix for details. The AGPL has some unusual conditions w.r.t. applications interacting with the network, please read it carefully.

CHAPTER 2

APPLICATION GUIDE

Scheme-GNUnet doesn't have any example applications, except the half-baked `examples/nse-web.scm`, `gnu/gnunet/scripts/download-store.scm` and `gnu/gnunet/scripts/publish-store.scm`. Over time, we hope we have something to write here, but for now, this chapter is empty.

CHAPTER 3

PROGRAMMING GUIDE

3.1. Concurrency

Scheme-GNUnet uses `guile-fibers` for concurrency, but supports POSIX-style threading as well, using the `(ice-9 threads)` Guile module. More concretely, this means `spawn-fiber` is used by default for starting asynchronous computations and public procedures accept an optional `#:spawn` argument accepting a procedure like `spawn-fiber` or `call-with-new-thread`. 'Conditions' can be used for synchronising concurrent computations, see the documentation of `guile-fibers` for details.

Repeated conditions Scheme-GNUnet has a variant of fibers conditions, named 'repeated conditions', in the module `(gnu gnunet concurrency repeated-condition)`. It is unfortunately ill-documented.

3.2. Configuration

There are a number of modules for accessing GNUnet configurations. Firstly, there is `(gnu gnunet config db)`, which is the module library code would typically use. For testing, one can create an empty configuration with the procedure `hash->configuration` from that module and `make-hashtable` from `(rnrs hashtables)`, using `hash-key` as hash function and `key=?` as comparison function:

```
(import (gnu gnunet config db)
        (gnu gnunet config value-parser)
        (rnrs hashtables))
(define config (hash->configuration (make-hashtable hash-key key=?)))
```

The resulting configuration `config` is initially empty, so set some *keys* in the *section* `nse`, to configure the network-size estimation service:

```
(set-value! identity config "nse" "UNIXPATH" "/tmp/nse.sock")
(set-value! number->string config "cadet" "MAX_ROUTES" 5000)
;; TODO: IP address, time durations, booleans, ...
```

Now read these values back:

```
(read-value value->file-name config "nse" "UNIXPATH")
;; -> /tmp/nse.sock
(read-value value->natural config "cadet" "MAX_ROUTES")
;; -> 5000
```

What if the configuration doesn't have a value for the specified section and key? Then an `&undefined-key-error` results:

```
(read-value value->natural config "kated" "MAX_ROUTES")
;; ->
;; ice-9/boot-9.scm:1685:16: In procedure raise-exception:
;; ERROR:
;;   1. &undefined-key-error:
;;       section: "kated"
;;       key: "MAX_ROUTES"
```

3.2.1. Locating configuration files

There are two – possibly non-existent – configuration files: the *user* configuration and the *system* configuration. The *system* configuration typically contains the paths for services like NSE, CORE, ... A user may choose not to use the services by the system and instead use their own. To do so, the user needs to override the paths in the *user* configuration. [defaults?] The module `(gnu gnunet config fs)` is responsible for determining the location of the configuration files to load and actually load configuration files. For determining the location of the configuration files, the procedures `locate-user-configuration` and `locate-system-configuration` can be used.

Warning 3.1. The C implementation's mechanism for user-system separation seems to work differently.

`(locate-user-configuration #:getenv=getenv)`

This procedure determines the location of the user configuration file, as a string, or `#false` if it could not be determined. If the location of the user configuration file is known, but the file does not exist, it is returned anyway, as a string.

If the environment variable `XDG_CONFIG_HOME` is set, the location of the file `gnunet.conf` in the directory `$XDG_CONFIG_HOME` is returned. If the environment variable is not set, the location of the file at `.config/gnunet.conf` in the home directory specified by the environment variable `HOME` is returned, if that environment variable exists. If both are unset, `#false` is returned.

The values of environment variables is determined with the procedure `getenv`.

`(locate-system-configuration)`

This procedure determines the location of the system configuration file, as a string.

Currently, this is always `/etc/gnunet.conf`.

3.2.2. Loading configuration files

Once the location of the configuration file is known, the file can be opened with the Scheme procedure `open-input-file`, which returns an input port. Then the procedure `load-configuration/port!` can be used to determine all section-key-values triples in the configuration.

`(load-configuration/port! set-value! port)`

Load the configuration from the input port *port*.

For each variable, call *set-value!* with the section name, variable name and a vector of the form `#(line line-number value)`, where *value* a list of expansible objects.

[document expansible objects][error reporting]

A variable assignment `[section] key=value${var}` can refer to variables defined in the `PATHS` section and variables from the environment. The previously described procedure `load-configuration/port!` will *not* expand such assignements. To expand variable assignments, use the procedure `make-expanded-configuration` instead.

`(make-expanded-configuration load! #:getenv=getenv)`

Make a configuration object. To populate the configuration, all the procedure *load!* with a `set-value!` procedure as expected by `load-configuration/port!`. The values from `set-value!` are added to the confoiguration and every variable is expanded.

To automatically load the defaults, the system configuration and the user configuration, use the thunk `load-configuration`:

`(load-configuration #:getenv=getenv #:files=...)`

Load the defaults, the system configuration and the user configuration and return the resulting configuration object. The list of files to load can be overriden by setting the undocumented *files* keyword argument.

Applications (whether graphical or textual) are recommended to use `load-configuration` by default, as it largely just works.

3.3. Manipulation of network structures

The modules `(gnu gnunet netstruct procedural)` and `(gnu gnunet netstruct syntactic)` can be used for formatting messages to be sent over the network or to a service. The macros `define-type` and `structure/packed` can be used to define new structures, like this:

```
(define-type /:msg:nse:estimate/example
  (structure/packed
   (properties '((message-symbol msg:nse:estimate)))
   (synopsis "Network size estimate")
   (documentation "Some explanation")
   (field (header /:message-header))
   (field (size-estimate ieee-double/big)
          (synopsis "Timestamp for when the estimate was made"))))
```

This example is taken from the `(gnu gnunet nse struct)` module and oversimplified. All its components will gradually explained. First, what actually is a network structure? This question is ambigious, because 'network structure' can refer to either the *value* or the *type*. The *value* is a sequence of octets, i.e., a sequence of numbers in the closed range 0–255. The *type* describes how the *value* is structured.

As an example, consider figure ?. There, the value is `0 12 1 65 64 51 238 123 71 27 58 149` and the type is `/:msg:nse:estimate/example`.

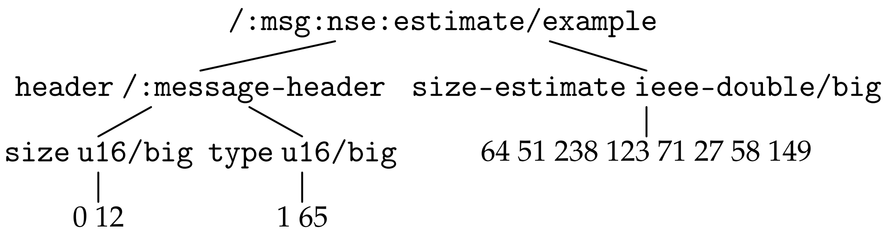

Figure 3.1. A network structure, both *value* and *type*.

This value has a *header* with a *size* '0 0 0 12' of type `u32/big`, so the *size* is 12. The *header* also has a *type* '0 0 1 65' of type `u32/big`, so the type is $256 \times 1 + 65 = 321$. The value also has a *size estimate* of type `ieee-double/big`. The octets `64 51 238 123 71 27 58 149`, interpreted as an IEEE double, form the number 19.93....

3.3.1. Documentation

A network structure can optionally have embedded documentation. More specifically, network structures can optionally have the *synopsis*, *documentation* and *properties* set. The *synopsis* is a short description of what the network structure represents, typically a one-liner. The *documentation* can be more detailed, explaining how the structure works, can be used, is used and should be used. The *properties* form a free-formed association list.

The synopsis, documentation and properties can be set on structured created with `structure/packed` and individual fields and can be accessed with the procedures `documentation`, `synopsis` and `properties`.

3.3.2. Reading and writing

The procedures `read%`, `set%!`, `sizeof` and `select` from `(gnu gnunet netstruct procedural)` or the like-named macros from `(gnu gnunet netstruct syntactic)` can be used for reading and writing network structures. The macros from `syntactic` behave like the procedures from `procedural` with some optimisations at expansion time. The procedures will be demonstrated with the `/:msg:nse:estimate/example` network structure defined previously.

First, create a memory slice with `make-slice/read-write` from `(gnu gnunet utils bv-slice)`. The required size can be determinded with `(sizeof /:msg:nse:estimate/example '())`. The role of `'()` will be explained later.

```
(import (gnu gnunet netstruct procedural)
        (gnu gnunet netstruct syntactic)
        (gnu gnunet utils bv-slice)
        (gnu gnunet util struct))

(define-type /:msg:nse:estimate/example
  (structure/packed
   (field (header /:message-header))
   (field (size-estimate ieee-double/big))))

(define message
  (make-slice/read-write (sizeof /:msg:nse:estimate/example '())))
```

The fields of message can be set with `(set%! netstruct '(field ...) slice value)`. The following code sets all the fields:

```
(set%! /:msg:nse:estimate/example '(header size) message
       (sizeof /:msg:nse:estimate/example '()))
(set%! /:msg:nse:estimate/example '(header type) message 165)
(set%! /:msg:nse:estimate/example '(size-estimate) message 19.2)
```

The size of an individual field can be determined with `(sizeof netstruct '(field ...))`. For example, the following code determines the size of the 'size' field in the header:

```
(sizeof /:msg:nse:estimate/example '(header size)) ; 12
```

The fields can also be read:

```
(read% /:msg:nse:estimate/example '(header size) message) ; 12
(read% /:msg:nse:estimate/example '(header type) message) ; 165
(read% /:msg:nse:estimate/example '(size-estimate) message) ; 19.2
```

3.3.3. Primitive types

There are a number of pre-defined types. First, there is `u8`, a single octet that is interpreted as an integer in the closed range $[0, 255]$. There are also types `uN/endian` for $N \in \{16, 32, 64\}$ and endian $\in \{\text{little}, \text{big}\}$, which interprets $N/8$ octets as integers in the closed range $[0, 2^N - 1]$. The types `ieee-double/big` and `ieee-double/little` are 8 octets long and represent floating-point numbers in IEEE 754 format ('binary64').

3.3.4. Packing

In contrast to C structures, Scheme-GNUnet network structures are always packed — there are no 'gaps' between fields.

3.4. Communication with services

To connect with a GNUnet service — this applies to both the C and Scheme implementation, the GNUnet service must bind a local domain socket[3.1] somewhere on the file system and the client (possibly another service) must connect to it. Connections to a service can be made with the `connect/fibers` procedure from `(gnu gnunet mq-impl stream)`, like this:

```
(define mq (connect/fibers config "nse" handlers error-handler))
```

3.4.1. Asynchronously connecting

This is an asynchronous operation: it will 'complete' immediately and the connection will actually be formed in the background. When the connection has actually be formed, the `error-handler` is called with the symbol `connection:connected`. To demonstrate, the following code asynchronously connects to the NSE service, and prints the text `"connected!"` when the connection has actually been formed.

3.1. The C implementation supports Internet sockets as well.

```
;; XXX test this, explain 'config' ...
(define (error-handler error . args)
  (case error
    ((connection:connected)
     (format #t "connected!~%"))
    (else (format #t "unknown error: ~a ~a~%" error args))))

(define mq
  (connect/fibers config "nse" (message-handlers) error-handler))
```

3.4.2. Message handler

When a message is received by the message queue, the corresponding message handler is invoked. Message handlers can be constructed with the `message-handler` macro and the `make-message-handler` procedure from `(gnu gnunet nse client)`, as follows:

```
(import (gnu gnunet mq handler)
        (gnu extractor enum)
        (gnu gnunet message protocols)
        (gnu gnunet util struct)
        (gnu gnunet utils bv-slice)
        (gnu gnunet netstruct syntactic))

(define handler/syntactic
  (message-handler
   (type (symbol-value message-type msg:util:dummy))
   ((interpose code) code)
   ((well-formed? slice)
    (= (slice-length slice)
       (sizeof /:message-header '())))
   ((handle! slice)
    (pk 'message: slice))))

(define handler/procedural
  (make-message-handler
   (symbol-value message-type msg:util:dummy)
   (lambda (thunk) (thunk))
   (lambda (slice)
     (= (slice-length slice)
        (sizeof /:message-header '())))
   (lambda (slice)
     (pk 'message: slice))))
```

As illustrated in the example code above, a message handler has four components: the *type* of message it handles, an *interposer* which will be explained later, the *verifier* deciding if a message is well-formed and the *handler procedure*.

The verifier is passed a bytevector slice with the message and should return `#true` if the message is well-formed and `#false` if it isn't. It may assume that the length of the slice corresponds to the length *in* the message header and is at least the length *of* the message header and that the type in the message header corresponds to the type of the message handler. Messages will only be passed to the handler procedue if the verifiers returns `#true`.

The handler procedure is passed a bytevector slice with the message, but only if the verifier considers it well-formed. The handler procedure and verifier are run from the *interposer*. The interposer is passed a thunk to execute and may e.g. install exception handlers and parameterise parameters. It can change the current input, output and error ports for example.

[document the message type database, various procedures]

3.4.3. Message type database

The module `(gnu gnunet message protocols)` has a mapping of symbolic names of every message type known to scheme-GNUnet to their numeric value. To use it, the macro `symbol-value` from `(gnu extractor enum)` is required and possibly `value->index` as well. To determine the numeric value of the message type `msg:nse:estimate`, one would write:

```
(define numeric-type
  (value->index (symbol-value message-type msg:nse:estimate)))
```

[other various enum procedures for introspection, documentation, ...?]

[how to define new message types]

3.4.4. Error handler

The message queue implementation usually just sends and receives messages, but some exceptional situations cannot be communicated with `send-message!` or `inject-message!`. For those, there is the `inject-error!` procedure. This variadic procedure accepts a message queue to inject the error into, a *key* (usually a symbol) describing the exceptional situation and rest arguments. It calls the *error handler* of the message queue with the key and rest arguments. The following errors can currently be reported by the built-in message queue implementations:

`connection:connected`

The connection to the server has been established.

`connection:interrupted`

The message queue has been closed before the connection to the server could be established.

`input:regular-end-of-file`

The connection has been closed by the server.

For people wondering about what happens if a connection becomes half-duplex: GNUnet does not have a notion of half-duplex message streams. If it is detected the underlying stream became half-duplex anyways, it will be treated as closed by scheme-GNUnet, resulting in this error. However, note that currently broken pipes cannot be reliably detected.

`input:premature-end-of-file`

The connection was closed by the server while a message was still being read. This can happen if the server was stopped while it was still sending the rest of the message.

`input:overly-small` *`type`* *`size`*

The message size in the header was smaller than the minimal message size. Sometimes, but not always, the message type *`type`* and message size *`size`* are available (as exact naturals). When they are not available, *`type`* and *`size`* are `#false` instead. This can only happen if the server or connection to the server is buggy.

`logic:no-handler` *`type . rest`*

The received message of type *`type`* (as an integer) does not have a corresponding message handler. *`rest`* is currently unspecified.

`logic:ill-formed` *`type . rest`*

The received message of type (as an integer) is ill-formed according to the message handler. *`rest`* is currently unspecified.

Consider automatically reconnecting after `input:regular-end-of-file` and `input:premature-end-of-file`, to allow the server to restart without having to manually restart every individual application. To report errors, see the section ? Error reporting.

3.4.5. Ordering of injected errors and messages and sent messages

This section describes how injected errors and messages and sent messages are ordered with respect to each other in the default message queue implementation. Messages are handled or corresponding `logic:no-handler` or `logic:ill-formed` errors are injected in the order that the messages are received. Before messages are read, `connection:connected` is injected. This error is injected at most once.

Soon after all messages are read (and therefore *soon* after all handled messages or corresponding errors), the error `input:regular-end-of-file`, `input:overly-small` or `input:premature-end-of-file` is injected. Only one of those errors can be injected for the entire lifetime of the message queue.

Be aware that *soon* is not *immediate* here! For example, it is possible for a message to be received, the port closed, a message queued for sending, the closing of the port being detected by the write fiber, `input:regular-end-of-file` being injected from the write fiber and the read fiber handling the received message, and the read fiber exiting because the port is closed, in that order.

Messages are sent (and received on the other side) in the order they were enqueued for sending. Likewise, the notify-sent callback of enqueued messages are called in order. If the notify-sent callback is called, it is before the message is received by the other side. The message and its notify-sent callback are only received by the other side and called after the message has been injected and `connection:connected` has been injected. It is possible for the notify-sent callback to be called without the message being received by the other side, e.g. if the port was closed during the notify-sent callback.

If a message is received by the other side, all previously-sent messages have be received before. If a notify-sent callback is invoked, all notify-sent callbacks of previous messages have been invoked before, except the messages that are eventually cancelled.

The errors `logic:no-handler` and `logic:ill-formed` are not fatal: later messages can still be read and handled. If `connection:interrupted` is injected, no other errors are ever injected, whether in the past or in the future. This error can only be injected once.

[I/O errors]

[envelopes]

3.4.6. Disconnecting

A message queue can be closed with the `close-queue!` procedure from `(gnu gnunet mq)`. In the default message queue implementation, this asynchronously closes the port and stops associated fibers. Closing ports when they won't be used anymore is important for limiting resource consumption, especially for servers that can have many connections. Closing message queues is an idempotent operation: closing a message queue twice is the same as closing it once. If a message queue is closed before a connection could be formed, `connection:interrupted` is injected instead of `connection:connected` and `connection:regular-end-of-file`.

3.5. Error reporting

Errors can be reported with the procedure `report-error` from the module `(gnu gnunet mq error-reporting)`. It can be called as `(report-error key argument ...)`, e.g. `(report-error ’logic:no-handler 3)`. By default, it reports the error to the current error port. If this is not desired, the output can be sent to another port by setting the parameter `textual-error-reporting-port`. If textual error reporting is not desired, the parameter `error-reporter` can be set to a procedure with the same interface as `report-error`. Such a procedure could e.g. open a GUI dialog, sent the message to the system logger or ignore the error.

Error messages are translated for the current locale.[TODO actually call bindtextdomain]

3.6. Estimation of the size of the network

GNUnet has a service that roughly estimates the size of the network – i.e., the number of peers. The module `(gnu gnunet nse client)` can be used to interact with this service. The connection is made with the procedure `connect`, which is accepts a *configuration* (see [reference]) and some optional keyword arguments. This procedure can be called as `(connect config #:updated updated #:connected connected #:disconnected disconnected)`. It returns a *NSE server object*.

The connection is made asynchronously; the thunk *connected* will be called when the connection has actually been made. Whenever a new estimate becomes available, the (optional) procedure *updated* is called with the new *estimate*. Alternatively, the procedure `estimate` can be called on the server object to return the latest available estimate. If the *server object* doesn't have an estimate yet, that procedure will return `#false` instead of an estimate.

When the connection is lost, the (optional) thunk *disconnected* is called and `(gnu gnunet nse client)` will retry connecting. To close the current connection, if any, and stop reconnecting, the idempotent procedure `disconnect!` can be called on the server object.

[input, validation, I/O errors?]

The estimate object has a number of accessors:

`(estimate:logarithmic-number-peers estimate)`

The base-2 logarithm of the number of peers (estimated), as a positive flonum, possibly zero or infinite

`(estimate:number-peers estimate)`

The number of peers (estimated), as a flonum, at least `1.0` and possibly infinite. This is not necessarily an (inexact) `integer?`, as it is only an estimate.

`(estimate:timestamp estimate)`

A timestamp when the estimate was made [something about epoch?]

`(estimate:standard-deviation estimate)`

The estimated standard deviation on the base-2 logarithm of peers, calculated over the last 64 rounds, with the $\frac{N}{N-1}$ correction. This is a positive flonum, possibly zero or infinite.

Assuming the network size is stable and the errors on the logarithmic estimate are normally distributed, the procedure `estimate:standard-deviation` can be used to put probablistic error bounds on the number of peers on the network. [example]

CHAPTER 4

IMPLEMENTATION DETAILS

TODO

Appendix A

GNU Free Documentation License

Version 1.3, 3 November 2008

0. Preamble

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

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

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

1. Applicability and definitions

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

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

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

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

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

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

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

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

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

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

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

2. VERBATIM COPYING

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

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

3. COPYING IN QUANTITY

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

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

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

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

4. MODIFICATIONS

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

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

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

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

D) Preserve all the copyright notices of the Document.

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

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

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

H) Include an unaltered copy of this License.

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

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

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

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

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

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

O) Preserve any Warranty Disclaimers.

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

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

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

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

5. COMBINING DOCUMENTS

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

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

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

6. COLLECTIONS OF DOCUMENTS

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

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

7. AGGREGATION WITH INDEPENDENT WORKS

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

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

8. TRANSLATION

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

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

9. TERMINATION

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

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

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

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

10. FUTURE REVISIONS OF THIS LICENSE

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

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

11. RELICENSING

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

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

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

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

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

ADDENDUM: How to use this License for your documents

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

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

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

```
    with the Invariant Sections being LIST THEIR TITLES, with the
    Front-Cover Texts being LIST, and with the Back-Cover Texts being
LIST.
```

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

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

Index