This manual is for Cuirass version 1.0.0, a build automation server.

Cuirass Reference Manual

Build automation server
for version 1.0.0, 23 March 2021

The Cuirass Developers

Table of Contents

Introduction

Cuirass is a general-purpose build automation server that checks out source files from VCS (Version Control System) repositories, executes build jobs, and stores build results in a database. It provides a web interface to monitor the build results, as well as an HTTP API. Cuirass is also able to send build notifications using different mechanisms such as RSS and email.

Cuirass is inspired by the Hydra (`https://nixos.org/hydra/`) continuous build system. Unlike Hydra, it is built on top of the GNU Guix (`https://www.gnu.org/software/guix/`) functional package manager.

The goal of Cuirass is to prevent software regressions by building a set of package definitions, system images and running periodical tests for various architectures. Cuirass is also responsible for GNU Guix binary substitutes production (see Section "Substitutes" in *Guix*).

Cuirass is deployed on the GNU Guix build farm at `https://ci.guix.gnu.org`. It is also common for Guix users to run their own Cuirass instance to build different sources, using different priorities (see Section "Continuous Integration" in *Guix*).

1 Specifications

The main Cuirass argument is the *specification* file. It describes the repositories that must be used, the build jobs and their priorities between other things.

`specification` [Data Type]

`name` The specification name as a Scheme symbol.

`build` (default: `all`)
The packages to be built by Cuirass. It defaults to `all`, which means that all the discovered packages in the subsequent `channels` field are to be selected.

It is also possible to set this field to:

- `core` Build only the core packages such as `gcc`, `guile` and `glibc`.
- `guix` Build only the Guix modules that are involved in the `guix pull` command.
- `hello` Build only the hello package.
- `(channels . list)` Build only the packages that are part of the given channel `list`. For instance, `(channels my-channel)` will only build the packages that are part of `my-channel` channel.
- `(packages . list)` Build only the specified packages in `list`. For instance, `(packages "strace" "perf")` will only build the packages `strace` and `perf`.
- `(manifests . list)` Build only the packages that are part of the manifests `list`. For instance, `(manifests "etc/manifest")` will only build the packages that are part of the `etc/manifest` file. This file must be provided by exactly one of the channels defined below.

`channels` (default: `(list %default-guix-channel)`)
The channels to be fetched by Cuirass (see Section "Channels" in *Guix*).

`build-outputs` (default: `()`)
The build artifacts that must be saved and proposed to download in the web interface as a list of `build-outputs` records.

`build-outputs` [Data Type]

`job` Save the build outputs of the build jobs which names match the `job` regexp.

`type` The build output type as a string. It is only used to describe the build output in the web interface.

`output` (default: `("out")`)
The job output if it has multiple outputs (see Section "Packages with Multiple Outputs" in *Guix*).

`path` The build output path within the job, as a string.

For instance, let's consider the `binary-tarball.x86_64-linux` job which produces the following output: `/gnu/store/xxx-guix-binary.tar.xz`.

The build output definition below will save the root element (`""`) of the `"out"` output of the `"binary-tarball.x86_64-linux"` job—i.e., the `"xxx-guix-binary.tar.xz"` file.

```
(build-output
 (job "binary-tarball*")
 (type "archive")
 (output "out")
 (path ""))
```

`notifications` (default: `()`)

The list of build notifications that must be sent. For instance:

```
(list (email
       (from "build@cuirass.org")
       (to "notification@myself.org")
       (server "sendmail:///etc/my-mailer.sh")))
```

will send build notifications emails from `build@cuirass.org` to `notifications@myself.org`, using `"sendmail:///etc/my-mailer.sh"` mailer.

The different notification types are described in the Chapter 2 [Notifications], page 5, section.

`priority` (default: `9`)

The specification priority relatively to the other specifications, as an integer ranging from 0 to 9 where 0 is the higher priority and 9 the lowest.

`systems` (default: `(list (%current-system))`)

Build every job for each system in this list. By default only the current system is selected.

2 Notifications

Cuirass supports different build notifications types, that can be passed in the `notifications` field of the specification record, see Chapter 1 [Specifications], page 3.

Cuirass sends build notifications each time a build is broken or fixed.

2.1 Email

Email build notifications can be enabled using the following record.

`email` [Data Type]

- `from` The email `From` field, as a string.
- `to` The email `To` field, as a string.
- `server` The mail server connection string. Cuirass uses the `mailutils` package. Hence the server can be specified as a remote SMTP mailbox (see Section "SMTP Mailboxes" in *Mailutils*) or as a program mailbox (see Section "Program Mailboxes" in *Mailutils*).

2.2 Mastodon

Mastodon build notifications can be enabled using the following record.

`mastodon` [Data Type]

The Mastodon credentials must be defined as Cuirass parameters, see Chapter 3 [Parameters], page 7.

2.3 RSS

Cuirass is proposing a build notification RSS feed at the following URL:

- `http://cuirass-url/events/rss[?specification=spec]` By default build notifications are sent for all specifications. If the `specification` argument is passed, they can be restricted to the *spec* specification.

3 Parameters

Cuirass is able to connect to different external services such as `postgresql` for the database, `zabbix` for machine monitoring and `mastodon` for build notifications. As those services often require using secret credentials, Cuirass can be passed a parameter file.

The parameters file can be passed using the `parameters` command line argument, see Chapter 5 [Invocation], page 11.

Here's an example parameter file:

```
(%cuirass-url "https://ci.guix.gnu.org")
(%zabbix-url "http://127.0.0.1:15412/api_jsonrpc.php")
(%mastodon-instance-name "My Instance")
(%mastodon-instance-url "https://instance.org")
(%mastodon-instance-token "secret-token")
```

`parameters` [Parameters]

`%cuirass-database` (default: `"cuirass"`)
: The Cuirass PostgreSQL database name.

`%cuirass-host` (default: `"/var/run/postgresql"`)
: The Cuirass PostgreSQL database host.

`%cuirass-url` (default: `#f`)
: The URL of the Cuirass web server. This is useful to send absolute links within notifications.

`%zabbix-url` (default: `#f`)
: The URL of the Zabbix monitoring server providing the workers status, if supported.

`%zabbix-user` (default: `"Admin"`)
: The user for Zabbix API authentication.

`%zabbix-password` (default: `"zabbix"`)
: The password for Zabbix API authentication.

`%mastodon-instance-name` (default: `#f`)
: The name of the Mastodon instance used to send build notifications.

`%mastodon-instance-url` (default: `#f`)
: The URL of the Mastodon instance.

`%mastodon-instance-token` (default: `#f`)
: The token used to authenticate on the Mastodon instance.

4 Build modes

Cuirass supports two mechanisms to build derivations.

4.1 With the local Guix daemon

This is the default build mechanism. Once the build jobs are evaluated, they are sent to the local Guix daemon. Cuirass then listens to the Guix daemon output to detect the various build events.

While this mode doesn't require any particular configuration, it doesn't scale well. The scheduling decisions of the Guix daemon are opaque and often suboptimal.

When Cuirass is used to build a large amount of jobs, the remote build mechanism described below should be preferred.

4.2 With the remote build mechanism.

This mode is harder to setup but scales way better. This is the build mode that is used on the GNU Guix build farm at `https://ci.guix.gnu.org`. The build jobs are not submitted to the local Guix daemon. Instead, a remote server dispatches build requests to the connect remote workers, according to the build priorities.

The remote server and the connected workers communicate using ZMQ over TCP. The workers are able to discover the remote server using Avahi.

The built items are exchanged as substitutes (see Section "Substitutes" in *Guix*) by spawning Guix publish servers both on the remote server and on each connected remote worker.

It can be enabled this way:

- Start the `cuirass register` process with the `build-remote` command line argument, see Chapter 5 [Invocation], page 11. This way, the registration process does not submit the new build jobs to the local Guix daemon.
- Start the `cuirass remote-server` process to dispatch the build jobs to the connected workers.
- Start at least one `cuirass remote-worker` process on any machine of the local network to actually perform the builds and report their status.

Note that some Cuirass features are only available when using this build mode. That's the case for:

- The build priority support.
- The notification mechanism, see Chapter 2 [Notifications], page 5.
- The transmission of `timeout` and `max-silent-time` package properties to the Guix daemon.
- The live build log mechanism of the Web interface.
- The workers status page of the Web inferface accessible at `http://cuirass-url/workers`.

The easiest way to setup such an infrastructure is to rely on the GNU Guix Cuirass services definitions (see Section "Continuous Integration" in *Guix*).

5 Invocation

5.1 Invoking cuirass register

The usual way to invoke `cuirass` registration process is as follows:

```
cuirass register --specifications specs
```

This starts a Cuirass registration instance building *specs* and storing the results using the default PostgreSQL database.

Additionally the following options can be used.

`--one-shot`
: Instead of executing `cuirass` as a daemon looping over the jobs. Only evaluate and build the specifications once.

`--cache-directory=`*`directory`*
: *directory* is the place where the VCS repositories used by the jobs are stored.

`--specifications=`*`specifications-file`*
`-S` *`specifications-file`*
: Add the specifications defined in *specifications-file* in the job database before launching the evaluation and build processes.

`--database=`*`database`*
`-D` *`database`*
: Use *database* as the database containing the jobs and the past build results. Since Cuirass uses PostgreSQL as a database engine, *database* must be a string such as `"dbname=cuirass host=localhost"`. By default, Cuirass uses the following connection string: `dbname=cuirass host=/var/run/postgresql"`.

`--parameters=`*`parameters-file`*
`-P` *`parameters-file`*
: Read parameters from the given *parameters-file*. The supported parameters are described here (see Chapter 3 [Parameters], page 7).

`--ttl=`*`duration`*
: Cuirass registers build results as garbage collector (GC) roots, thereby preventing them from being deleted by the GC. The `--ttl` option instructs it to keep those GC roots live for at least *duration*—e.g., `1m` for one month, `2w` for two weeks, and so on. The default is 30 days.

  Those GC roots are typically stored in `/var/guix/gcroots/profiles/per-user/`*`user`*`/cuirass`, where *user* is the user under which Cuirass is running.

`--interval=`*`n`*
`-I` *`n`*
: Wait *n* seconds between each poll.

`--use-substitutes`
: This can be useful when you are not interested in building the dependencies of a particular job.

`--threads=`*`n`*
: Use up to *n* kernel threads.

n should be lower than or equal to the number of CPU cores on the machine. In general though, having a large *n* is not very useful since the work of Cuirass is primarily I/O-bound—on the contrary, large values of *n* may increase overhead. The default value should be appropriate for most cases.

`--version`
`-V` Display the actual version of `cuirass`.

`--help`
`-h` Display an help message that summarize all the options provided.

5.2 Invoking cuirass web

The usual way to invoke the `cuirass` web server is as follows:

```
cuirass web
```

This starts a Cuirass web server on the default port. Additionally the following options can be used.

`--database=`*database*
`-D` *database*
Use *database* as the database containing the jobs and the past build results. Since Cuirass uses PostgreSQL as a database engine, *database* must be a string such as `"dbname=cuirass host=localhost"`. By default, Cuirass uses the following connection string: `dbname=cuirass host=/var/run/postgresql"`.

`--parameters=`*parameters-file*
`-P` *parameters-file*
Read parameters from the given *parameters-file*. The supported parameters are described here (see Chapter 3 [Parameters], page 7).

`--port=`*num*
`-p` *num* Make the HTTP interface listen on port *num*. Use port 8080 by default.

`--listen=`*host*
Make the HTTP interface listen on network interface for *host*. Use localhost by default.

`--version`
`-V` Display the actual version of `cuirass`.

`--help`
`-h` Display an help message that summarize all the options provided.

5.3 Invoking cuirass remote-server

The `remote-server` command starts a daemon that is able to communicate with `remote-worker` processes. Its role is to answer build requests from the workers, by sending back derivations that must be built.

On build completion it updates the database accordingly and possibly fetches build substitutes. The `remote-server` and `remote-worker` processes communicate using ZMQ over TCP.

Additionally the following options can be used.

`--backend-port=`*port*
: The TCP port for communicating with `remote-worker` processes using ZMQ. It defaults to `5555`.

`--log-port=`*port*
: The TCP port of the log server. It defaults to `5556`.

`--publish-port=`*port*
: The TCP port of the publish server. It defaults to `5557`.

`--parameters=`*parameters-file*
`-P` *parameters-file*
: Read parameters from the given *parameters-file*. The supported parameters are described here (see Chapter 3 [Parameters], page 7).

`--database=`*database*
`-D` *database*
: Use *database* PostgreSQL connection string.

`--cache=`*directory*
: Use *directory* to cache build log files.

`--trigger-substitute-url=`*URL*
: Once a substitute is successfully fetched, trigger substitute baking at *URL*.

`--user=`*user*
: Change privileges to *user* as soon as possible—i.e., once the signing key has been read.

`--public-key=`*file*
`--private-key=`*file*
: Use the specific *file*s as the public/private key pair used to sign the store items being published.

`--version`
`-V`
: Display the actual version of `cuirass`.

`--help`
`-h`
: Display an help message that summarize all the options provided.

5.4 Invoking cuirass remote-worker

The `remote-worker` command starts a daemon that is able to communicate with a `remote-server` process. Its role is to request builds to the `remote-server`, perform them and report their status.

The `remote-worker` is able to discover a `remote-server` process on the local network using Avahi and connect to it.

Additionally the following options can be used.

`--workers=`*count*
: Start *count* parallel workers. It defaults to `1`.

`--publish-port=`*port*
: The TCP port of the publish server. It defaults to `5558`.

`--server=`*ip-address*
: Do not use Avahi discovery and connect to the given `remote-server` IP address.

`--systems=`*systems*
: Only request builds for the given *systems*. It defaults to `(list (%current-system))`.

`--public-key=`*file*
`--private-key=`*file*
: Use the specific *file*s as the public/private key pair used to sign the store items being published.

`--version`
`-V`
: Display the actual version of `cuirass`.

`--help`
`-h`
: Display an help message that summarize all the options provided.

6 Web API

The Cuirass web API is inspired from the Hydra one.

6.1 API description

6.1.1 Evaluation information

Single evaluation

It is possible to query the Cuirass web server for evaluation information. The dedicated API is "/api/evaluation?id=*eval-id*" where *eval-id* is the unique id associated to the evaluation in database.

For instance, querying a local Cuirass web server can be done with `curl` and `jq` to format the JSON response :

```
$ curl -s "http://localhost:8080/api/evaluation?id=1" | jq

{
  "id": 1,
  "specification": "guix-master",
  "status": 0,
  "timestamp": 1615289011,
  "checkouttime": 1615289011,
  "evaltime": 1615289655,
  "checkouts": [
    {
      "commit": "bd311f5a6ccbd4696ac77f0426a036bb375a2f10",
      "channel": "guix",
      "directory": "/gnu/store/6978xw9vs4ybg2pc3g736p1dba2056yl-guix-bd311f5"
    }
  ]
}
```

The nominal output is a JSON object whose fields are described hereafter.

`id` The unique build id.

`specification`
The associated specification name, as a string.

`status` The evaluation status, as an integer. Possible values are :

```
-1 -> started
0 -> succeeded
1 -> failed
2 -> aborted
```

`checkouttime`
The timestamp after channel checkout.

`evaltime` The timestamp after evaluation completion.

`checkouts`
: The evaluation checkouts as a JSON object.

Multiple evaluations

The latest evaluations list can be obtained with the API "/api/evaluations". The output is a JSON array of evaluations. Evaluations are represented as in the "/api/evaluation?id=*eval-id*" API.

This request accepts a mandatory parameter.

`nr` Limit query result to nr elements. This parameter is *mandatory*.

6.1.2 Build information

It is possible to query Cuirass web server for build informations. The dedicated API is "/build/*build-id*" where *build-id* is the unique id associated to the build in database.

The build information can also be queried by output. For example, '`/output/kg9mirg6xbvzcp0a98v7326n1nvvwgsj-hello-2.10`' will return the details of the output, along with the build if available.

```
$ curl -s "http://localhost:8080/build/2" | jq

{
  "id": 2,
  "jobset": "guix",
  "job": "acpica-20150410-job",
  "timestamp": 1501347493,
  "starttime": 1501347493,
  "stoptime": 1501347493,
  "buildoutputs": {
    "out": {
      "path": "/gnu/store/6g3njhfzqpdm335s7qhvmwvs5l7gcbq1-acpica-20150410"
    }
  },
  "system": "x86_64-linux",
  "nixname": "acpica-20150410",
  "buildstatus": 0,
  "weather": 0,
  "busy": 0,
  "priority": 0,
  "finished": 1,
  "buildproducts": null
}
```

If requested *build-id* is not known, the HTTP code 404 is answered with a JSON error message. For example:

```
$ curl -s "http://localhost:8080/build/fff"

{"error" : "Build with ID fff doesn't exist."}
```

The nominal output is a JSON object whose fields are described hereafter.

`id` The unique build id.

`jobset` The associated specification name, as a string.

`job` The associated job-name, as a string.

`timestamp`
Timestamp taken at build creation time.

`starttime`
Timestamp taken at build start time.

`stoptime` Timestamp taken at build stop time.

`buildoutputs`
Build outputs as a JSON object. The keys names are referring to the eventual output names. The associated value is another JSON object which only key is `path`. `path` value is the output directory in store as a string.

`system` System name of the build, as a string.

`nixname` Derivation name, as a string.

`buildstatus`
Build status, as an integer. Possible values are :

```
0 -> succeeded
1 -> failed
2 -> failed dependency
3 -> failed other
4 -> cancelled
```

`weather` Build weather, as an integer.

```
-1 -> unknown
0 -> new-success
1 -> new-failure
2 -> still-succeeding
3 -> still-failing
```

`busy` Whether the build is pending, as an integer.

`priority` Build priority, as an integer.

`finished` Build finished, as an integer.

`buildproducts`
Build products in store as a JSON object.

6.1.3 Build raw log output

It is possible to ask Cuirass for the raw build output log with the API "/build/*build-id*/log/raw" where *build-id* is the unique id associated to the build in database.

The output is a raw text, for example:

```
$ curl http://localhost:8080/build/2/log/raw
```

```
starting phase ‘set-SOURCE-DATE-EPOCH’
phase ‘set-SOURCE-DATE-EPOCH’ succeeded after 0.0 seconds
starting phase ‘set-paths’
...
```

If requested *build-id* is not known, the HTTP code 404 is answered with a JSON error message. For example:

```
$ curl -s "http://localhost:8080/build/fff/log/raw"

{"error" : "Build with ID fff doesn’t exist."}
```

6.1.4 Latest builds

The list of latest builds can be obtained with the API "/api/latestbuilds". The output is a JSON array of builds. Builds are represented as in the "/build/*build-id*" API.

This request accepts a mandatory parameter and multiple optional ones.

`nr` Limit query result to nr elements. This parameter is *mandatory*.

`jobset` Filter query result to builds with the given `jobset`.

`job` Filter query result to builds with the given `job` name.

`system` Filter query result to builds with the given `system`.

For example, to ask for the ten last builds:

```
$ curl "http://localhost:8080/api/latestbuilds?nr=10"
```

or the five last builds where jobset “guix”:

```
$ curl "http://localhost:8080/api/latestbuilds?nr=5&jobset=guix"
```

If no builds matching given parameters are found, an empty JSON array is returned.

6.1.5 Queued builds

The list of queued builds can be obtained with the API "/api/queue". The output is a JSON array of builds. Builds are represented as in the "/build/*build-id*" API.

This request accepts a mandatory parameter.

`nr` Limit query result to nr elements. This parameter is *mandatory*.

7 Database schema

Cuirass uses a PostgreSQL database to store information about jobs and past build results, but also to coordinate the execution of jobs.

The database contains the following tables: `Specifications`, `Checkouts`, `Evaluations`, `Builds`, `Outputs`, `Metrics`, `BuildProducts`, `Events` and `Workers`. The purpose of each of these tables is explained below.

7.1 Specifications

This table stores specifications describing the repositories from whence Cuirass fetches code and the environment in which it will be processed. Entries in this table must have values for the following text fields:

`name` This field holds the name of the specification. This field is also the primary key of this table.

`channels` The channels to be fetched by Cuirass as an SEXP string.

`build_ouputs`
The build outputs to be saved by Cuirass as an SEXP string.

`notifications`
The build notifications to be sent by Cuirass as an SEXP string.

`priority` The specification priority relatively to the other specifications, as an integer ranging from 0 to 9 where 0 is the higher priority and 9 the lowest.

`systems` The systems for which build jobs must be evaluated, as a comma separated list.

7.2 Checkouts

When a specification is processed, the repositories must be downloaded at a certain revision as specified. The download is called a checkout. The `Checkouts` table stores the new checkouts for every specification when it is being processed.

The `Checkouts` table has the following columns:

`specification`
The specification associated with the checkout.

`revision` The revision of the checkout. Within the same specification, two checkouts can't be identical: they can't have the same revision.

`evaluation`
The evaluation that was triggered by the addition of that new checkout.

`channel` The channel associated with the checkout.

`directory`
The directory into which the checkout was extracted.

`timestamp`
The checkout insertion timestamp.

7.3 Evaluations

An evaluation relates a specification with the revision of the repository specified therein. Builds (see below) belong to a specific evaluation.

The `Evaluations` table has the following columns:

`id` This is an automatically incrementing numeric identifier.

`specification`
This field holds the `name` of a specification from the `Specifications` table.

`status` This integer field hold the evaluation status. Possible values are:

- started (`-1`)
- succeeded (`0`)
- failed (`1`)
- aborted (`2`)

`timestamp`
The timestamp at evaluation insertion.

`checkout` The timestamp after channel checkout.

`evaltime` The timestamp after evaluation completion.

7.4 Builds

This table holds records of the derivations and their build status. Note that a job will be registered here only if its derivation doesn't already exist.

`derivation`
This text field holds the absolute name of the derivation file that resulted in this build.

`evaluation`
This integer field references the evaluation identifier from the `Evaluations` table, indicating to which evaluation this build belongs.

`job_name` This text field holds the name of the job.

`system` This text field holds the system name of the derivation.

`nix_name` This text field holds the name of the derivation —e.g., `coreutils-8.24`.

`worker` This text field references the name of worker performing the build from the `Workers` table.

`log` This text field holds the absolute file name of the build log file.

`status` This integer field holds the build status of the derivation.

`last_status`
This integer field holds the build status of the previous job evaluation.

`weather` This integer field holds the weather of the build. Possible values are:

- unknown (`-1`)

- new-success (0)
- new-failure (1)
- still-succeeding (2)
- still-failing (3)

`priority` The build priority relatively to the other builds with the same `job_name`, as an integer ranging from 0 to 99 where 0 is the higher priority and 99 the lowest.

`max_silent`
This integer field holds the number of seconds of silence after which a build process times out.

`timeout` This integer field holds the number of seconds of activity after which a build process times out.

`timestamp`
This integer field holds a timestamp taken at build creation time.

`starttime`
This integer field holds a timestamp taken at build start time. Currently, it has the same value as the `timestamp` above.

`stoptime` This integer field holds a timestamp taken at build stop time. Currently, it has the same value as the `timestamp` above.

7.5 Outputs

This table keep tracks for every eventual build outputs. Each build stored in `Builds` table may have zero (if it has failed), one or multiple outputs.

`derivation`
This field holds the `derivation` of a build from the `Builds` table.

`name` This text field holds the name of the output.

`path` This text field holds the path of the output.

7.6 Metrics

This table contains several metrics that are recorded by the `metrics` fiber periodically.

`field` This text field holds the application field of the metric.

`type` This integer field holds the type of the metric.

`path` This float field holds the value of the metric.

`evaltime` The metric insertion timestamp.

7.7 BuildProducts

This table contains the saved build products, that are proposed to download through the web interface.

`build` This integer field holds a reference to the build `id` from the `Builds` table, the build product belongs to.

`type` This text field holds the build product type.

`file_size` This integer field holds build product size in bytes.

`checksum` This text field holds the build product checksum.

`path` This text field holds the build product absolute store path.

7.8 Notifications

This table contains the notifications that are queued for sending.

`id` This is an automatically incrementing numeric identifier.

`type` This text field holds the SEXP representation of the notification.

`build` This integer fields references the build id associated with the notification.

7.9 Workers

This table contains the registered workers when Cuirass is using the remote building mechanism.

`name` This text field holds the worker name. This field is also the primary key of this table.

`address` This text field holds the worker IP address.

`machine` This text field holds the worker machine name.

`systems` This text field holds the systems that are supported by the worker, as a comma separated list of systems.

`last_seen` This integer field holds the timestamp of the last communication with the worker.

8 Contributing

Everyone is welcome to contribute to Cuirass. You can report bugs, send patches and share your ideas with others by sending emails the mailing list.

Development is done using the Git distributed version control system. Thus, access to the repository is not strictly necessary. We welcome contributions in the form of patches as produced by `git format-patch`. Please write commit logs in the ChangeLog format (see Section "Change Logs" in *GNU Coding Standards*); you can check the commit history for examples.

When posting a patch to the mailing list, use '`[PATCH] ...`' as a subject. You may use your email client or the `git send-email` command. We prefer to get patches in plain text messages, either inline or as MIME attachments. You are advised to pay attention if your email client changes anything like line breaks or indentation which could potentially break the patches.

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

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

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

1. APPLICABILITY AND DEFINITIONS

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

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

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

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

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

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

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

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

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

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

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

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

2. VERBATIM COPYING

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

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

3. COPYING IN QUANTITY

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

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

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

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

4. MODIFICATIONS

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

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

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

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

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

D. Preserve all the copyright notices of the Document.

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

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

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

H. Include an unaltered copy of this License.

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

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

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

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

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

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

O. Preserve any Warranty Disclaimers.

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

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

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

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

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

5. COMBINING DOCUMENTS

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

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

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

6. COLLECTIONS OF DOCUMENTS

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

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

7. AGGREGATION WITH INDEPENDENT WORKS

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

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

8. TRANSLATION

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

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

9. TERMINATION

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

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

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

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

10. FUTURE REVISIONS OF THIS LICENSE

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

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

11. RELICENSING

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

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

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

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

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

ADDENDUM: How to use this License for your documents

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

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

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

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

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

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

Concept Index