reuse Documentation

Release 0.10.1

Free Software Foundation Europe

Dec 17, 2020

CONTENTS

1 reuse **1**
- 1.1 Background . . . 1
- 1.2 Install . . . 2
- 1.3 Usage . . . 2
- 1.4 Maintainers . . . 3
- 1.5 Contribute . . . 4
- 1.6 License . . . 4

2 Usage **5**
- 2.1 Implementation details . . . 5
- 2.2 addheader . . . 5
- 2.3 lint . . . 7

3 Credits **11**
- 3.1 Development Lead . . . 11
- 3.2 Contributors . . . 11
- 3.3 Translators . . . 11

4 Change log **13**
- 4.1 0.10.1 - 2020-05-14 . . . 13
- 4.2 0.10.0 - 2020-04-24 . . . 13
- 4.3 0.9.0 - 2020-04-21 . . . 14
- 4.4 0.8.1 - 2020-02-22 . . . 14
- 4.5 0.8.0 - 2020-01-20 . . . 15
- 4.6 0.7.0 - 2019-11-28 . . . 16
- 4.7 0.6.0 - 2019-11-19 . . . 16
- 4.8 0.5.2 - 2019-10-27 . . . 17
- 4.9 0.5.1 - 2019-10-24 [YANKED] . . . 17
- 4.10 0.5.0 - 2019-08-29 . . . 17
- 4.11 0.4.1 - 2019-08-07 . . . 18
- 4.12 0.4.0 - 2019-08-07 . . . 18
- 4.13 0.3.4 - 2019-04-15 . . . 19
- 4.14 0.3.3 - 2018-07-15 . . . 19
- 4.15 0.3.2 - 2018-07-15 . . . 19
- 4.16 0.3.1 - 2018-07-14 . . . 20
- 4.17 0.3.0 - 2018-05-16 . . . 20
- 4.18 0.2.0 - 2018-04-17 . . . 20
- 4.19 0.1.1 - 2017-12-14 . . . 21
- 4.20 0.1.0 - 2017-12-14 . . . 21
- 4.21 0.0.4 - 2017-11-06 . . . 22

4.22 0.0.3 - 2017-11-06 . . . 22
4.23 0.0.2 - 2017-11-03 . . . 22

5 reuse **23**
5.1 reuse package . . . 23

6 Indices and tables **31**

Python Module Index **33**

Index **35**

CHAPTER

ONE

REUSE

reuse is a tool for compliance with the REUSE recommendations.

- Documentation: https://reuse.readthedocs.io and https://reuse.software
- Source code: https://github.com/fsfe/reuse-tool
- PyPI: https://pypi.python.org/pypi/reuse
- REUSE: 3.0
- Python: 3.6+

1.1 Background

Copyright and licensing is difficult, especially when reusing software from different projects that are released under various different licenses. REUSE was started by the Free Software Foundation Europe (FSFE) to provide a set of recommendations to make licensing your Free Software projects easier. Not only do these recommendations make it easier for you to declare the licenses under which your works are released, but they also make it easier for a computer to understand how your project is licensed.

As a short summary, the recommendations are threefold:

1. Choose and provide licenses
2. Add copyright and licensing information to each file
3. Confirm REUSE compliance

You are recommended to read our tutorial for a step-by-step guide through these three steps. The FAQ covers basic questions about licensing, copyright, and more complex use cases. Advanced users and integrators will find the full specification helpful.

This tool exists to facilitate the developer in complying with the above recommendations.

There are other tools that have a lot more features and functionality surrounding the analysis and inspection of copyright and licenses in software projects. The REUSE helper tool, on the other hand, is solely designed to be a simple tool to assist in compliance with the REUSE recommendations.

1.2 Install

1.2.1 Installation via pip

To install reuse, you need to have the following pieces of software on your computer:

- Python 3.6+
- pip

You then only need to run the following command:

```
pip3 install --user reuse
```

After this, make sure that `~/.local/bin` is in your `$PATH`.

To update reuse, run this command:

```
pip3 install --user --upgrade reuse
```

For full functionality, the following pieces of software are recommended:

- Git
- Mercurial 4.3+

1.2.2 Installation via package managers

There are packages available for easy install on some operating systems. You are welcome to help us package this tool for more distributions!

- Arch Linux (AUR): reuse
- Fedora: reuse
- openSUSE: reuse
- GNU Guix: reuse
- NixOS: reuse

1.2.3 Installation from source

You can also install this tool from the source code, but we recommend the methods above for easier and more stable updates. Please make sure the requirements for the installation via pip are present on your machine.

```
python3 setup.py install
```

1.3 Usage

First, read the REUSE tutorial. In a nutshell:

1. Put your licenses in the `LICENSES/` directory.
2. Add a comment header to each file that says `SPDX-License-Identifier: GPL-3.0-or-later`, and `SPDX-FileCopyrightText: $YEAR $NAME`. You can be flexible with the format, just make sure that the line starts with `SPDX-FileCopyrightText:`.

3. Verify your work using this tool.

To check against the recommendations, use `reuse lint`:

```
~/Projects/reuse-tool $ reuse lint
[...]

Congratulations! Your project is compliant with version 3.0 of the REUSE␣
↪Specification :-)
```

This tool can do various more things, detailed in the documentation. Here a short summary:

- `addheader` — Add copyright and/or licensing information to the header of a file.
- `download` — Download the specified license into the `LICENSES/` directory.
- `init` — Set up the project for REUSE compliance.
- `lint` — Verify the project for REUSE compliance.
- `spdx` — Generate an SPDX Document of all files in the project.

1.3.1 Run in Docker

The `fsfe/reuse` Docker image is available on Docker Hub. With it, you can easily include REUSE in CI/CD processes. This way, you can check for REUSE compliance for each build. In our resources for developers you can learn how to integrate the REUSE tool in Drone, Travis, GitHub, or GitLab CI.

You can run the helper tool simply by providing the command you want to run (e.g., `lint`, `spdx`). The image's working directory is `/data` by default. So if you want to lint a project that is in your current working directory, you can mount it on the container's `/data` directory, and tell the tool to lint. That looks a little like this:

```
docker run --volume $(pwd):/data fsfe/reuse lint
```

You can also provide additional arguments, like so:

```
docker run --volume $(pwd):/data fsfe/reuse --include-submodules spdx -o out.spdx
```

1.3.2 Run as pre-commit hook

You can automatically run `reuse lint` on every commit as a pre-commit hook for Git. This uses pre-commit. Once you have it installed, add this to the `.pre-commit-config.yaml` in your repository:

```
repos:
-   repo: https://github.com/fsfe/reuse-tool
    rev: latest
    hooks:
    - id: reuse
```

Then run `pre-commit install`. Now, every time you commit, `reuse lint` is run in the background, and will prevent your commit from going through if there was an error.

1.4 Maintainers

- Carmen Bianca Bakker - carmenbianca@fsfe.org

1.5 Contribute

Any pull requests or suggestions are welcome at https://github.com/fsfe/reuse-tool or via e-mail to one of the maintainers. General inquiries can be sent to reuse@lists.fsfe.org.

Interaction within this project is covered by the FSFE's Code of Conduct.

Starting local development is very simple, just execute the following commands:

```
git clone git@github.com:fsfe/reuse-tool.git
cd reuse-tool/
python3 -mvenv venv
source venv/bin/activate
make develop
```

You need to run `make develop` at least once to set up the virtualenv.

Next, run `make help` to see the available interactions.

1.6 License

This work is licensed under multiple licences. Because keeping this section up-to-date is challenging, here is a brief summary as of April 2020:

- All original source code is licensed under GPL-3.0-or-later.
- All documentation is licensed under CC-BY-SA-4.0.
- Some configuration and data files are licensed under CC0-1.0.
- Some code borrowed from spdx/tool-python is licensed under Apache-2.0.

For more accurate information, check the individual files.

CHAPTER

TWO

USAGE

The *overview* documents some basic usage on how to use this tool. It is highly recommended to read the overview first, and you might not even need to read this chapter. This chapter covers details that might not be immediately obvious when using the tool. This chapter does not cover *everything*, assuming that the user is helped enough by `reuse --help` and `reuse <subcommand> --help`.

2.1 Implementation details

This section covers implementation details that are true for the entire tool.

When searching for copyright and licensing tags inside of files, the tool does not strictly limit itself to the header comment as prescribed by the specification. It searches the first 4 kibibytes of the file. This makes sure that the tool can parse any type of plain-text file, even if the comment style is not recognised.

If a file is found to have an unparseable tag, that file is not parsed at all. This is a bug.

The tool does not verify the correctness of copyright notices. If it finds any line containing '©', 'Copyright', or 'SPDX-FileCopyrightText:', then the tag and everything following it is considered a valid copyright notice, even if the copyright notice is not compliant with the specification.

When running the tool, the root of the project is automatically found if the working directory is inside a VCS repository. Otherwise, it treats the working directory as the root of the project. You can override the root of the project with the `--root` optional argument.

Git submodules are automatically ignored unless `--include-submodules` is passed as optional argument.

2.2 addheader

`addheader` makes it possible to semi-automatically add copyright and licensing information into the header of a file. This is useful especially in scenarios where you want to add a copyright holder or license to a lot of files without having to manually edit the header of each file.

> **Warning:** You should be cautious with using `addheader` in automated processes. While nothing is stopping you from using it in your release script, you should make sure that the information it adds is actually reflective of reality. This is best verified manually.

The basic usage is `reuse addheader --copyright="Jane Doe" --license=MIT my_file.py`. This will add the following header to the file (assuming that the current year is 2019):

```
# SPDX-FileCopyrightText: 2019 Jane Doe
#
# SPDX-License-Identifier: MIT
```

You can use as many `--copyright` and `--copyright` arguments, so long as there is at least one such argument.

The REUSE header is placed at the very top of the file. If a different REUSE header already existed—at the top or elsewhere—its tags are copied, and the header is replaced in-place.

Shebangs are always preserved at the top of the file.

2.2.1 Comment styles

The tool normally tries to auto-detect the comment style to use from the file extension of a file, and use that comment style. If the tool is unable to detect the comment style, or if it detects the wrong style, you can override the style using `--style`. The supported styles are:

- AppleScript
- ASPX
- BibTex
- C
- CSS
- Haskell
- HTML
- Jinja
- JSX
- ML
- Python
- TeX

If your comment style is not supported or a file extension is not correctly detected, please open an issue.

2.2.2 Templates

When the tool adds a header to a file, it normally first lists all copyright statements alphabetically, adds a single empty line, and then lists all SPDX License Expressions alphabetically. That is all that the header contains. It is possible to change this behaviour, and use a custom type of header that contains extra text. This is done through Jinja2 templates.

The default template is:

```
{% for copyright_line in copyright_lines %}
{{ copyright_line }}
{% endfor %}

{% for expression in spdx_expressions %}
SPDX-License-Identifier: {{ expression }}
{% endfor %}
```

Templates are automatically commented by the tool, depending on the detected or specified comment style.

You can create your own Jinja2 templates and place them in `.reuse/templates/`. If you create the template `mytemplate.jinja2`, you can use it with `reuse addheader --copyright="Jane Doe" --template=mytemplate foo.py`.

Inside of the template, you have access to the following variables:

- `copyright_lines` — a list of copyright notices (string).
- `spdx_expressions` — a list of SPDX License Expressions (string).

In the future, more variables will be added.

In some cases, you might want to do custom comment formatting. In those cases, you can pre-format your header as a comment. When doing so, suffix your template with `.commented.jinja2`.

An example of a custom template with manual commenting is:

```
/*
{% for copyright_line in copyright_lines %}
 * {{ copyright_line }}
{% endfor %}
{% if copyright_lines and spdx_expressions %}
 *
{% endif %}
{% for expression in spdx_expressions %}
 * SPDX-License-Identifier: {{ expression }}
{% endfor %}
{% if "GPL-3.0-or-later" in spdx_expressions %}
 *
 * This program is free software: you can redistribute it and/or modify it under
 * the terms of the GNU General Public License as published by the Free Software
 * Foundation, either version 3 of the License, or (at your option) any later
 * version.
 *
 * This program is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS
 * FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License along with
 * this program. If not, see <https://www.gnu.org/licenses/>.
{% endif %}
 */
```

2.3 lint

`lint` is the main component of the tool. Summarily, it verifies whether the project is compliant with the REUSE Specification. Its main goal is to find all files that do not have copyright and licensing information in their headers, but it also checks a few other things.

The STDOUT output of `reuse lint` is valid Markdown. Occasionally some logging will be printed to STDERR, which is not valid Markdown.

This is some example output of `reuse lint`:

```
# BAD LICENSES

```

(continues on next page)

(continued from previous page)

```
'bad-license' found in:
* LICENSES/bad-license.txt


# UNUSED LICENSES

The following licenses are not used:
* bad-license


# MISSING COPYRIGHT AND LICENSING INFORMATION

The following files have no copyright and licensing information:
* no-information.txt


# SUMMARY

* Bad licenses: bad-license
* Deprecated licenses:
* Licenses without file extension:
* Missing licenses:
* Unused licenses: bad-license
* Used licenses: Apache-2.0, CC-BY-SA-4.0, CC0-1.0, GPL-3.0-or-later
* Read errors: 0
* Files with copyright information: 57 / 58
* Files with license information: 57 / 58

Unfortunately, your project is not compliant with version 3.0 of the REUSE␣
↪Specification :-(
```

2.3.1 Criteria

These are the criteria that the linter checks against:

Bad licenses

Licenses that are found in `LICENSES/` that are not found in the SPDX License List or do not start with `LicenseRef-` are bad licenses.

Deprecated licenses

If a license has an SPDX License Identifier that has been deprecated by SPDX, the license will show up here.

Licenses without file extension

These are licenses whose file names are a valid SPDX License Identifier, but which do not have a file extension.

Missing licenses

If a license is referred to in a comment header, but the license is not found in the `LICENSES/` directory, then that license is missing.

Unused licenses

Conversely, if a license is found in the `LICENSES/` directory but is not referred to in any comment header, then that license is unused.

Read errors

Not technically a criterion, but files that cannot be read by the operating system are read errors, and need to be fixed.

Files with copyright and license information

Every file needs to have copyright and licensing information associated with it. The REUSE Specification details several ways of doing it. By and large, these are the methods:

- Placing tags in the header of the file.
- Placing tags in a `.license` file adjacent to the file.
- Putting the information in the DEP5 file.

If a file is found that does not have copyright and/or license information associated with it, then the project is not compliant.

CHAPTER

THREE

CREDITS

3.1 Development Lead

- Carmen Bianca Bakker <carmenbianca@fsfe.org>

3.2 Contributors

- Sebastian Schuberth <schuberth@fsfe.org>
- Max Mehl <max.mehl@fsfe.org>
- Matija Šuklje <hook@fsfe.org>
- Greg Kroah-Hartman
- Basil Peace
- Keith Maxwell
- Stefan Bakker <s.bakker777@gmail.com>
- Kirill Elagin <kirelagin@gmail.com>
- John Mulligan <jmulligan@redhat.com>
- Tuomas Siipola <tuomas@zpl.fi>
- Diego Elio Pettenò <flameeyes@flameeyes.com>

3.3 Translators

- Dutch:
  - André Ockers <ao@fsfe.org>
  - Carmen Bianca Bakker <carmenbianca@fsfe.org>
- French:
  - OliBug <act-zebug@riseup.net>
  - Vincent Lequertier <vincent@fsfe.org>
- Galician:
  - pd <euklade@gmail.com>

- German:
  - Max Mehl <max.mehl@fsfe.org>
  - Thomas Doczkal <doczkal@fsfe.org>
- Esperanto:
  - Carmen Bianca Bakker <carmenbianca@fsfe.org>
  - Tirifto <tirifto@posteo.cz>
- Italian:
- Luca Bonissi <lucabon@fsfe.org>
- Portuguese:
  - José Vieira <jvieira33@sapo.pt>
- Spanish:
  - flow <adolflow@sindominio.net>
  - pd <euklade@gmail.com>
  - Roberto Bauglir <bauglir@fsfe.org>
- Turkish:
  - T. E. Kalayci <tekrei@member.fsf.org>

CHAPTER

FOUR

CHANGE LOG

This change log follows the Keep a Changelog spec. Every release contains the following sections:

- `Added` for new features.
- `Changed` for changes in existing functionality.
- `Deprecated` for soon-to-be removed features.
- `Removed` for now removed features.
- `Fixed` for any bug fixes.
- `Security` in case of vulnerabilities.

The versions follow semantic versioning.

4.1 0.10.1 - 2020-05-14

4.1.1 Fixed

- Updated license list to 3.8-106-g4cfec76.

4.2 0.10.0 - 2020-04-24

4.2.1 Added

- Add support for autoconf comment style (listed as m4).
- More file types are recognised:
  - Cython (`.pyx`, `.pxd`)
  - Sass and SCSS (`.sass`, `.scss`)
  - XSL (`.xsl`)
  - Mailmap (`.mailmap`)

4.2.2 Changed

- The Docker image has an entrypoint now. In effect, this means running:

```
docker run -v $(pwd):/data fsfe/reuse lint
```

instead of

```
docker run -v $(pwd):/data fsfe/reuse reuse lint.
```

4.3 0.9.0 - 2020-04-21

4.3.1 Added

- Added support for Mercurial 4.3+.
- A pre-commit hook has been added.
- When an incorrect SPDX identifier is forwarded to `download` or `init`, the tool now suggests what you might have meant.

4.3.2 Changed

- Under the hood, a lot of code that has to do with Git and Mercurial was moved into its own module.
- The Docker image has been changed such that it now automagically runs `reuse lint` on the `/data` directory unless something else is specified by the user.

4.3.3 Fixed

- Fixed a bug with `addheader --explicit-license` that would result in `file.license.license` if `file.license` already existed.
- Fixed a Windows-only bug to do with calling subprocesses.
- Fixed a rare bug that would trigger when a directory is both ignored and contains a `.git` file.

4.4 0.8.1 - 2020-02-22

4.4.1 Added

- Support Jinja (Jinja2) comment style.
- Support all multi-line comment endings when parsing for SPDX information.

4.4.2 Fixed

- Improvements to German translation by Thomas Doczkal.
- No longer remove newlines at the end of files when using `addheader`.
- There can now be a tab as whitespace after `SPDX-License-Identifier` and `SPDX-FileCopyrightText`.

4.5 0.8.0 - 2020-01-20

4.5.1 Added

- Implemented `--root` argument to specify the root of the project without heuristics.
- The linter will complain about licenses without file extensions.
- Deprecated licenses are now recognised. `lint` will complain about deprecated licenses.
- ProjectReport generation (`lint`, `spdx`) now uses Python multiprocessing, more commonly called multi-threading outside of Python. This has a significant speedup of approximately 300% in testing. Because of overhead, performance increase is not exactly linear.
- For setups where multiprocessing is unsupported or unwanted, `--no-multiprocessing` is added as flag.
- `addheader` now recognises many more extensions. Too many to list here.
- `addheader` now also recognises full filenames such as `Makefile` and `.gitignore`.
- Added BibTex comment style.
- Updated translations:
  - Dutch (André Ockers, Carmen Bianca Bakker)
  - French (OliBug, Vincent Lequertier)
  - Galician (pd)
  - German (Max Mehl)
  - Esperanto (Carmen Bianca Bakker)
  - Portuguese (José Vieira)
  - Spanish (Roberto Bauglir)
  - Turkish (T. E. Kalayci)

4.5.2 Changed

- The linter output has been very slightly re-ordered to be more internally consistent.
- `reuse --version` now prints a version with a Git hash on development versions. Towards that end, the tool now depends on `setuptools-scm` during setup. It is not a runtime dependency.

4.5.3 Removed

- `lint` no longer accepts path arguments. Where previously one could do `reuse lint SUBDIRECTORY`, this is no longer possible. When linting, you must always lint the entire project. To change the project's root, use `--root`.
- `FileReportInfo` has been removed. `FileReport` is used instead.

4.5.4 Fixed

- A license that does not have a file extension, but whose full name is a valid SPDX License Identifier, is now correctly identified as such. The linter will complain about them, however.
- If the linter detects a license as being a bad license, that license can now also be detected as being missing.
- Performance of `project.all_files()` has been improved by quite a lot.
- Files with CRLF line endings are now better supported.

4.6 0.7.0 - 2019-11-28

4.6.1 Changed

- The program's package name on PyPI has been changed from `fsfe-reuse` to `reuse`. `fsfe-reuse==1.0.0` has been created as an alias that depends on `reuse`. `fsfe-reuse` will not receive any more updates, but will still host the old versions.
- For users of `fsfe-reuse`, this means:
  - If you depend on `fsfe-reuse` or `fsfe-reuse>=0.X.Y` in your requirements.txt, you will get the latest version of `reuse` when you install `fsfe-reuse`. You may like to change the name to `reuse` explicitly, but this is not strictly necessary.
  - If you depend on `fsfe-reuse==0.X.Y`, then you will keep getting that version. When you bump the version you depend on, you will need to change the name to `reuse`.
  - If you depend on `fsfe-reuse>=0.X.Y<1.0.0`, then 0.6.0 will be the latest version you receive. In order to get a later version, you will need to change the name to `reuse`.

4.7 0.6.0 - 2019-11-19

4.7.1 Added

- `--include-submodules` is added to also include submodules when linting et cetera.
- `addheader` now also recognises the following extensions:
  - .kt
  - .xml
  - .yaml
  - .yml

4.7.2 Changed

- Made the workaround for `MachineReadableFormatError` introduced in 0.5.2 more generic.
- Improved shebang detection in `addheader`.
- For `addheader`, the SPDX comment block now need not be the first thing in the file. It will find the SPDX comment block and deal with it in-place.
- Git submodules are now ignored by default.

- `addheader --explicit-license` now no longer breaks on unsupported filetypes.

4.8 0.5.2 - 2019-10-27

4.8.1 Added

- `python3 -m reuse` now works.

4.8.2 Changed

- Updated license list to 3.6-2-g2a14810.

4.8.3 Fixed

- Performance of `reuse lint` improved by at least a factor of 2. It no longer does any checksums on files behind the scenes.
- Also handle `MachineReadableFormatError` when parsing DEP5 files. Tries to import that error. If the import is unsuccessful, it is handled.

4.9 0.5.1 - 2019-10-24 [YANKED]

This release was replaced by 0.5.2 due to importing `MachineReadableFormatError`, which is not a backwards-compatible change.

4.10 0.5.0 - 2019-08-29

4.10.1 Added

- TeX and ML comment styles added.
- Added `--year` and `--exclude-year` to `reuse addheader`.
- Added `--template` to `reuse addheader`.
- Added `--explicit-license` to `reuse addheader`.
- `binaryornot` added as new dependency.
- Greatly improved the usage documentation.

4.10.2 Changed

- `reuse addheader` now automatically adds the current year to the copyright notice.
- `reuse addheader` preserves the original header below the new header if it did not contain any SPDX information.
- `reuse addheader` now correctly handles `.license` files.

- Bad licenses are no longer resolved to LicenseRef-Unknown. They are instead resolved to the stem of the path. This reduces the magic in the code base.
- `.gitkeep` files are now ignored by the tool.
- Changed Lisp's comment character from ';;' to ';'.

4.11 0.4.1 - 2019-08-07

4.11.1 Added

- `--all` argument help to `reuse download`, which downloads all detected missing licenses.

4.11.2 Fixed

- When using `reuse addheader` on a file that contains a shebang, the shebang is preserved.
- Copyright lines in `reuse spdx` are now sorted.
- Some publicly visible TODOs were patched away.

4.12 0.4.0 - 2019-08-07

This release is a major overhaul and refactoring of the tool. Its primary focus is improved usability and speed, as well as adhering to version 3.0 of the REUSE Specification.

4.12.1 Added

- `reuse addheader` has been added as a way to automatically add copyright statements and license identifiers to the headers of files. It is currently not complete.
- `reuse init` has been added as a way to initialise a REUSE project. Its functionality is currently scarce, but should improve in the future.

4.12.2 Changed

- `reuse lint` now provides a helpful summary instead of merely spitting out non-compliant files.
- `reuse compile` is now `reuse spdx`.
- In addition to `Copyright` and ©, copyright lines can be marked with the tag `SPDX-FileCopyrightText:`. This is the new recommended default.
- Project no longer depends on pygit2.
- The list of SPDX licenses has been updated.
- `Valid-License-Identifier` is no longer used, and licenses and exceptions can now only live inside of the LICENSES/ directory.

4.12.3 Removed

- Removed `--ignore-debian`.
- Removed `--spdx-mandatory`, `--copyright-mandatory`, `--ignore-missing` arguments from `reuse lint`.
- Remove `reuse license`.
- GPL-3.0 and GPL-3.0+ (and all other similar GPL licenses) are no longer detected as SPDX identifiers. Use GPL-3.0-only and GPL-3.0-or-later instead.

4.12.4 Fixed

- Scanning a Git directory is a lot faster now.
- Scanning binary files is a lot faster now.

4.13 0.3.4 - 2019-04-15

This release should be a short-lived one. A new (slightly backwards-incompatible) version is in the works.

4.13.1 Added

- Copyrights can now start with © in addition to `Copyright`. The former is now recommended, but they are functionally similar.

4.13.2 Changed

- The source code of reuse is now formatted with black.
- The repository has been moved from https://git.fsfe.org/reuse/reuse to https://gitlab.com/reuse/reuse.

4.14 0.3.3 - 2018-07-15

4.14.1 Fixed

- Any files with the suffix `.spdx` are no longer considered licenses.

4.15 0.3.2 - 2018-07-15

4.15.1 Fixed

- The documentation now builds under Python 3.7.

4.16 0.3.1 - 2018-07-14

4.16.1 Fixed

- When using reuse from a child directory using pygit2, correctly find the root.

4.17 0.3.0 - 2018-05-16

4.17.1 Changed

- The output of `reuse compile` is now deterministic. The files, copyright lines and SPDX expressions are sorted alphabetically.

4.17.2 Fixed

- When a GPL license could not be found, the correct `-only` or `-or-later` extension is now used in the warning message, rather than a bare `GPL-3.0`.
- If you have a license listed as `SPDX-Valid-License: GPL-3.0-or-later`, this now correctly matches corresponding SPDX identifiers. Still it is recommended to use `SPDX-Valid-License: GPL-3.0` instead.

4.18 0.2.0 - 2018-04-17

4.18.1 Added

- Internationalisation support added. Initial support for:
  - English.
  - Dutch.
  - Esperanto.
  - Spanish.

4.18.2 Fixed

- The license list of SPDX 3.0 has deprecated `GPL-3.0` and `GPL-3.0+` et al in favour of `GPL-3.0-only` and `GPL-3.0-or-later`. The program has been amended to accommodate sufficiently for those licenses.

4.18.3 Changed

- `Project.reuse_info_of` now extracts, combines and returns information both from the file itself and from debian/copyright.
- `ReuseInfo` now holds sets instead of lists.
  - As a result of this, `ReuseInfo` will not hold duplicates of copyright lines or SPDX expressions.
- click removed as dependency. Good old argparse from the library is used instead.

4.19 0.1.1 - 2017-12-14

4.19.1 Changed

- The `reuse --help` text has been tidied up a little bit.

4.19.2 Fixed

- Release date in change log fixed.
- The PyPI homepage now gets reStructuredText instead of Markdown.

4.20 0.1.0 - 2017-12-14

4.20.1 Added

- Successfully parse old-style C and HTML comments now.
- Added `reuse compile`, which creates an SPDX bill of materials.
- Added `--ignore-missing` to `reuse lint`.
- Allow to specify multiple paths to `reuse lint`.
- `chardet` added as dependency.
- `pygit2` added as soft dependency. reuse remains usable without it, but the performance with `pygit2` is significantly better. Because `pygit2` has a non-Python dependency (`libgit2`), it must be installed independently by the user. In the future, when reuse is packaged natively, this will not be an issue.

4.20.2 Changed

- Updated to version 2.0 of the REUSE recommendations. The most important change is that `License-Filename` is no longer used. Instead, the filename is deducted from `SPDX-License-Identifier`. This change is **NOT** backwards compatible.
- The conditions for linting have changed. A file is now non-compliant when:
  - The license associated with the file could not be found.
  - There is no SPDX expression associated with the file.
  - There is no copyright notice associated with the file.
- Only read the first 4 KiB (by default) from code files rather than the entire file when searching for SPDX tags. This speeds up the tool a bit.
- `Project.reuse_info_of` no longer raises an exception. Instead, it returns an empty `ReuseInfo` object when no reuse information is found.
- Logging is a lot prettier now. Only output entries from the `reuse` module.

4.20.3 Fixed

- `reuse --ignore-debian compile` now works as expected.
- The tool no longer breaks when reading a file that has a non-UTF-8 encoding. Instead, `chardet` is used to detect the encoding before reading the file. If a file still has errors during decoding, those errors are silently ignored and replaced.

4.21 0.0.4 - 2017-11-06

4.21.1 Fixed

- Removed dependency on `os.PathLike` so that Python 3.5 is actually supported

4.22 0.0.3 - 2017-11-06

4.22.1 Fixed

- Fixed the link to PyPI in the README.

4.23 0.0.2 - 2017-11-03

This is a very early development release aimed at distributing the program as soon as possible. Because this is the first release, the changelog is a little empty beyond "created the program".

The program can do roughly the following:

- Detect the license of a given file through one of three methods (in order of precedence):
  - Information embedded in the .license file.
  - Information embedded in its header.
  - Information from the global debian/copyright file.
- Find and report all files in a project tree of which the license could not be found.
- Ignore files ignored by Git.
- Do some logging into STDERR.

CHAPTER

FIVE

REUSE

5.1 reuse package

5.1.1 Submodules

reuse.download module

Functions for downloading license files from spdx/license-data-list.

reuse.download.**add_arguments**(*parser*)
Add arguments to parser.

Return type None

reuse.download.**download_license**(*spdx_identifier*)
Download the license text from the SPDX repository.

Parameters **spdx_identifier** (str) – SPDX identifier of the license.

Raises **requests.RequestException** – if the license could not be downloaded.

Return type str

Returns The license text.

reuse.download.**put_license_in_file**(*spdx_identifier*, *destination*)
Download a license and put it in the destination file.

This function exists solely for convenience.

Parameters

- **spdx_identifier** (str) – SPDX License Identifier of the license.
- **destination** (PathLike) – Where to put the license.

Raises

- **requests.RequestException** – if the license could not be downloaded.
- **FileExistsError** – if the license file already exists.

Return type None

reuse.download.**run**(*args*, *project*, *out=<_io.TextIOWrapper name='<stdout>' mode='w' encoding='UTF-8'>*)
Download license and place it in the LICENSES/ directory.

Return type int

reuse.header module

Functions for manipulating the comment headers of files.

exception reuse.header.**MissingSpdxInfo**
Bases: Exception

Some SPDX information is missing from the result.

reuse.header.**add_arguments**(*parser*)
Add arguments to parser.

Return type None

reuse.header.**create_header**(*spdx_info*, *header=None*, *template=None*, *template_is_commented=False*, *style=None*)
Create a header containing *spdx_info*. *header* is an optional argument containing a header which should be modified to include *spdx_info*. If *header* is not given, a brand new header is created.

template, *template_is_commented*, and *style* determine what the header will look like, and whether it will be commented or not.

Raises

- **CommentCreateError** – if a comment could not be created.
- ***MissingSpdxInfo*** – if the generated comment is missing SPDX information.

Return type str

reuse.header.**find_and_replace_header**(*text*, *spdx_info*, *template=None*, *template_is_commented=False*, *style=None*)
Find the first SPDX comment block in *text*. That comment block is replaced by a new comment block containing *spdx_info*. It is formatted as according to *template*. The template is normally uncommented, but if it is already commented, *template_is_commented* should be True.

If both *style* and *template_is_commented* are provided, *style* is only used to find the header comment.

If the comment block already contained some SPDX information, that information is merged into *spdx_info*.

If no header exists, one is simply created.

text is returned with a new header.

Raises

- **CommentCreateError** – if a comment could not be created.
- ***MissingSpdxInfo*** – if the generated comment is missing SPDX information.

Return type str

reuse.header.**run**(*args*, *project*, *out=<_io.TextIOWrapper name='<stdout>' mode='w' encoding='UTF-8'>*)
Add headers to files.

Return type int

reuse.init module

Functions for REUSE-ifying a project.

reuse.init.**add_arguments**(*parser*)
Add arguments to parser.

reuse.init.**prompt_licenses**(*out=<_io.TextIOWrapper name='<stdout>' mode='w' encoding='UTF-8'>*)
Prompt the user for a list of licenses.

Return type List[str]

reuse.init.**run**(*args*, *project*, *out=<_io.TextIOWrapper name='<stdout>' mode='w' encoding='UTF-8'>*)
List all non-compliant files.

reuse.lint module

All linting happens here. The linting here is nothing more than reading the reports and printing some conclusions.

reuse.lint.**add_arguments**(*parser*)
Add arguments to parser.

reuse.lint.**lint**(*report*, *out=<_io.TextIOWrapper name='<stdout>' mode='w' encoding='UTF-8'>*)
Lint the entire project.

Return type bool

reuse.lint.**lint_bad_licenses**(*report*, *out=<_io.TextIOWrapper name='<stdout>' mode='w' encoding='UTF-8'>*)
Lint for bad licenses. Bad licenses are licenses that are not in the SPDX License List or do not start with LicenseRef-.

Return type Iterable[str]

reuse.lint.**lint_deprecated_licenses**(*report*, *out=<_io.TextIOWrapper name='<stdout>' mode='w' encoding='UTF-8'>*)
Lint for deprecated licenses.

Return type Iterable[str]

reuse.lint.**lint_files_without_copyright_and_licensing**(*report*, *out=<_io.TextIOWrapper name='<stdout>' mode='w' encoding='UTF-8'>*)
Lint for files that do not have copyright or licensing information.

Return type Iterable[str]

reuse.lint.**lint_licenses_without_extension**(*report*, *out=<_io.TextIOWrapper name='<stdout>' mode='w' encoding='UTF-8'>*)
Lint for licenses without extensions.

Return type Iterable[str]

reuse.lint.**lint_missing_licenses**(*report*, *out=<_io.TextIOWrapper name='<stdout>' mode='w' encoding='UTF-8'>*)
Lint for missing licenses. A license is missing when it is referenced in a file, but cannot be found.

Return type Iterable[str]

reuse.lint.**lint_read_errors**(*report*, *out=<_io.TextIOWrapper name='<stdout>' mode='w' encoding='UTF-8'>*)
Lint for read errors.

Return type Iterable[str]

reuse.lint.**lint_summary**(*report*, *out=<_io.TextIOWrapper name='<stdout>' mode='w' encoding='UTF-8'>*)
Print a summary for linting.

Return type None

reuse.lint.**lint_unused_licenses**(*report*, *out=<_io.TextIOWrapper name='<stdout>' mode='w' encoding='UTF-8'>*)
Lint for unused licenses.

Return type Iterable[str]

reuse.lint.**run**(*args*, *project*, *out=<_io.TextIOWrapper name='<stdout>' mode='w' encoding='UTF-8'>*)
List all non-compliant files.

reuse.project module

Module that contains the central Project class.

class reuse.project.**Project**(*root*, *include_submodules=False*)
Bases: object

Simple object that holds the project's root, which is necessary for many interactions.

all_files(*directory=None*)
Yield all files in *directory* and its subdirectories.

The files that are not yielded are:

- Files ignored by VCS (e.g., see .gitignore)
- Files/directories matching IGNORE_*_PATTERNS.

Return type Iterator[Path]

relative_from_root(*path*)
If the project root is /tmp/project, and *path* is /tmp/project/src/file, then return src/file.

Return type Path

property root
Path to the root of the project.

Return type Path

spdx_info_of(*path*)
Return SPDX info of *path*.

This function will return any SPDX information that it can find, both from within the file and from the .reuse/dep5 file.

Return type *SpdxInfo*

reuse.project.**create_project**()
Create a project object. Try to find the project root from CWD, otherwise treat CWD as root.

Return type *Project*

reuse.report module

Module that contains reports about files and projects for linting.

class reuse.report.**FileReport**(*name*, *path*, *do_checksum=True*)
Bases: object

Object that holds a linting report about a single file. Importantly, it also contains SPDX File information in spdxfile.

classmethod generate(*project*, *path*, *do_checksum=True*)
Generate a FileReport from a path in a Project.

Return type *FileReport*

to_dict()
Turn the report into a json-like dictionary.

class reuse.report.**ProjectReport**(*do_checksum=True*)
Bases: object

Object that holds linting report about the project.

bill_of_materials()
Generate a bill of materials from the project.

See https://spdx.org/specifications.

Return type str

property files_without_copyright
Iterable of paths that have no copyright information.

Return type Iterable[PathLike]

property files_without_licenses
Iterable of paths that have no license information.

Return type Iterable[PathLike]

classmethod generate(*project*, *do_checksum=True*, *multiprocessing=True*)
Generate a ProjectReport from a Project.

Return type *ProjectReport*

to_dict()
Turn the report into a json-like dictionary.

property unused_licenses
Set of license identifiers that are not found in any file report.

Return type Set[str]

property used_licenses
Set of license identifiers that are found in file reports.

Return type Set[str]

reuse.spdx module

Compilation of the SPDX Document.

reuse.spdx.**add_arguments**(*parser*)
Add arguments to the parser.

Return type None

reuse.spdx.**run**(*args*, *project*, *out=<_io.TextIOWrapper name='<stdout>' mode='w' encoding='UTF-8'>*)
Print the project's bill of materials.

Return type int

reuse.vcs module

This module deals with version control systems.

class reuse.vcs.**VCSStrategy**(*project*)
Bases: abc.ABC

Strategy pattern for version control systems.

abstract classmethod find_root(*cwd=None*)
Try to find the root of the project from *cwd*. If none is found, return None.

Raises **NotADirectoryError** – if directory is not a directory.

Return type Optional[Path]

abstract classmethod in_repo(*directory*)
Is *directory* inside of the VCS repository?

Raises **NotADirectoryError** – if directory is not a directory.

Return type bool

abstract is_ignored(*path*)
Is *path* ignored by the VCS?

Return type bool

class reuse.vcs.**VCSStrategyGit**(*project*)
Bases: *reuse.vcs.VCSStrategy*

Strategy that is used for Git.

classmethod find_root(*cwd=None*)
Try to find the root of the project from *cwd*. If none is found, return None.

Raises **NotADirectoryError** – if directory is not a directory.

Return type Optional[Path]

classmethod in_repo(*directory*)
Is *directory* inside of the VCS repository?

Raises **NotADirectoryError** – if directory is not a directory.

Return type bool

is_ignored(*path*)
Is *path* ignored by the VCS?

Return type bool

class reuse.vcs.**VCSStrategyHg**(*project*)
Bases: *reuse.vcs.VCSStrategy*

Strategy that is used for Mercurial.

classmethod find_root(*cwd=None*)
Try to find the root of the project from *cwd*. If none is found, return None.

Raises **NotADirectoryError** – if directory is not a directory.

Return type Optional[Path]

classmethod in_repo(*directory*)
Is *directory* inside of the VCS repository?

Raises **NotADirectoryError** – if directory is not a directory.

Return type bool

is_ignored(*path*)
Is *path* ignored by the VCS?

Return type bool

class reuse.vcs.**VCSStrategyNone**(*project*)
Bases: *reuse.vcs.VCSStrategy*

Strategy that is used when there is no VCS.

classmethod find_root(*cwd=None*)
Try to find the root of the project from *cwd*. If none is found, return None.

Raises **NotADirectoryError** – if directory is not a directory.

Return type Optional[Path]

classmethod in_repo(*directory*)
Is *directory* inside of the VCS repository?

Raises **NotADirectoryError** – if directory is not a directory.

Return type bool

is_ignored(*path*)
Is *path* ignored by the VCS?

Return type bool

reuse.vcs.**find_root**(*cwd=None*)
Try to find the root of the project from *cwd*. If none is found, return None.

Raises **NotADirectoryError** – if directory is not a directory.

Return type Optional[Path]

5.1.2 Module contents

reuse is a tool for compliance with the REUSE recommendations.

exception reuse.**IdentifierNotFound**
Bases: *reuse.ReuseException*

Could not find SPDX identifier for license file.

exception reuse.**ReuseException**
Bases: Exception

Base exception.

class reuse.**SpdxInfo**(*spdx_expressions*, *copyright_lines*)

Bases: tuple

Simple structure for holding SPDX information.

The two iterables MUST be sets.

property copyright_lines

Alias for field number 1

property spdx_expressions

Alias for field number 0

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

r

reuse, 29
reuse.download, 23
reuse.header, 24
reuse.init, 24
reuse.lint, 25
reuse.project, 26
reuse.report, 27
reuse.spdx, 27
reuse.vcs, 28

INDEX

A

add_arguments() (*in module reuse.download*), 23
add_arguments() (*in module reuse.header*), 24
add_arguments() (*in module reuse.init*), 24
add_arguments() (*in module reuse.lint*), 25
add_arguments() (*in module reuse.spdx*), 27
all_files() (*reuse.project.Project method*), 26

B

bill_of_materials() (*reuse.report.ProjectReport method*), 27

C

copyright_lines() (*reuse.SpdxInfo property*), 30
create_header() (*in module reuse.header*), 24
create_project() (*in module reuse.project*), 26

D

download_license() (*in module reuse.download*), 23

F

FileReport (*class in reuse.report*), 27
files_without_copyright() (*reuse.report.ProjectReport property*), 27
files_without_licenses() (*reuse.report.ProjectReport property*), 27
find_and_replace_header() (*in module reuse.header*), 24
find_root() (*in module reuse.vcs*), 29
find_root() (*reuse.vcs.VCSStrategy class method*), 28
find_root() (*reuse.vcs.VCSStrategyGit class method*), 28
find_root() (*reuse.vcs.VCSStrategyHg class method*), 28
find_root() (*reuse.vcs.VCSStrategyNone class method*), 29

G

generate() (*reuse.report.FileReport class method*), 27
generate() (*reuse.report.ProjectReport class method*), 27

I

IdentifierNotFound, 29
in_repo() (*reuse.vcs.VCSStrategy class method*), 28
in_repo() (*reuse.vcs.VCSStrategyGit class method*), 28
in_repo() (*reuse.vcs.VCSStrategyHg class method*), 29
in_repo() (*reuse.vcs.VCSStrategyNone class method*), 29
is_ignored() (*reuse.vcs.VCSStrategy method*), 28
is_ignored() (*reuse.vcs.VCSStrategyGit method*), 28
is_ignored() (*reuse.vcs.VCSStrategyHg method*), 29
is_ignored() (*reuse.vcs.VCSStrategyNone method*), 29

L

lint() (*in module reuse.lint*), 25
lint_bad_licenses() (*in module reuse.lint*), 25
lint_deprecated_licenses() (*in module reuse.lint*), 25
lint_files_without_copyright_and_licensing() (*in module reuse.lint*), 25
lint_licenses_without_extension() (*in module reuse.lint*), 25
lint_missing_licenses() (*in module reuse.lint*), 25
lint_read_errors() (*in module reuse.lint*), 25
lint_summary() (*in module reuse.lint*), 25
lint_unused_licenses() (*in module reuse.lint*), 26

M

MissingSpdxInfo, 24

P

Project (*class in reuse.project*), 26
ProjectReport (*class in reuse.report*), 27
prompt_licenses() (*in module reuse.init*), 24

put_license_in_file() (*in module reuse.download*), 23

R

relative_from_root() (*reuse.project.Project method*), 26
reuse (*module*), 29
reuse.download (*module*), 23
reuse.header (*module*), 24
reuse.init (*module*), 24
reuse.lint (*module*), 25
reuse.project (*module*), 26
reuse.report (*module*), 27
reuse.spdx (*module*), 27
reuse.vcs (*module*), 28
ReuseException, 29
root() (*reuse.project.Project property*), 26
run() (*in module reuse.download*), 23
run() (*in module reuse.header*), 24
run() (*in module reuse.init*), 25
run() (*in module reuse.lint*), 26
run() (*in module reuse.spdx*), 28

S

spdx_expressions() (*reuse.SpdxInfo property*), 30
spdx_info_of() (*reuse.project.Project method*), 26
SpdxInfo (*class in reuse*), 29

T

to_dict() (*reuse.report.FileReport method*), 27
to_dict() (*reuse.report.ProjectReport method*), 27

U

unused_licenses() (*reuse.report.ProjectReport property*), 27
used_licenses() (*reuse.report.ProjectReport property*), 27

V

VCSStrategy (*class in reuse.vcs*), 28
VCSStrategyGit (*class in reuse.vcs*), 28
VCSStrategyHg (*class in reuse.vcs*), 28
VCSStrategyNone (*class in reuse.vcs*), 29