Free software explained

Everything you need to know to use, re-use, and build free software

Drew DeVault
SourceHut

July 20, 2022

I am not a lawyer, so

Target audience:

- Individuals working on free software
- Small organizations working on free software
- Small companies working on/with free software

Not in the target audience:

- Large companies working on proprietary products
- Anyone writing their own license
- Drawing outside of the lines, doing weird stuff

Some definitions

Free software: Software which meets the "four freedoms", defined by the Free Software Foundation; distributed with its source code.

Open source software: Software which meets the Open Source Definition, defined by the Open Source Initiative; distributed with its source code.

Non-free software: Software which is neither of the above.

Source available: Software which includes access to its source code, but does not offer the rights guaranteed by free or open source software.

Proprietary software: Software distributed without the source code, generally held as a trade secret of a commercial entity.

The four freedoms

"Free software" is software distributed in a manner which guarantees the "four freedoms":

0. The freedom to run the program as you wish, for any purpose.
1. The freedom to study and change how the program works.
2. The freedom to redistribute copies so you can help others.
3. The freedom to distribute copies of your modified versions to others.

"Free" as in freedom – not as in a free beer.

Source: "What is Free Software?"
`https://www.gnu.org/philosophy/free-sw.en.html`

The Open Source Definition

The Open Source Definition is similar. Its characteristics, paraphrased, are:

- Unrestricted re-distribution, for sale or otherwise
- Access to the source code
- The right to modify the code
- No discrimination against persons or fields of endeavour
- Some other stuff

Source: "The Open Source Definition"
`https://opensource.org/osd`

Free software? Open source? Which one?

Virtually all free software is also open source software and vice-versa. The difference is in the underlying philosophy of their respective movements. In short:

Free software is about user freedom. The free software movement is a political movement which aims to liberate computer software for the people who use it.

Open source software is about commercial utility. The open source movement is about making software development more efficient by allowing commercial entities to pool their resources (and/or exploit the free labor of enthusiastic hackers).

FOSS: Free and open source software.

Non-free software movements

Some of the more popular ideas which compete with free and open source software are:

Open core
Some companies publish "open core" products, where the "core" is FOSS, but additional products are built on top of it which are not. Examples: Redis, GitLab

Exclusive commercial rights
Some source-available software provides the source code but limits the right to commercial use to a single publisher. Examples: MongoDB's Business Source License, SSPL

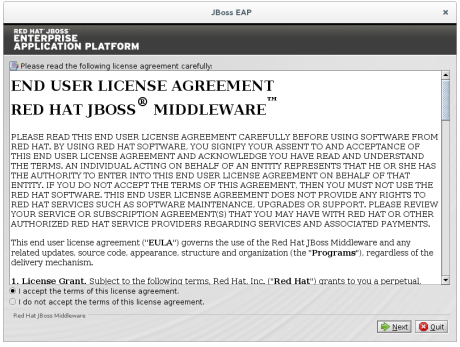
Discrimination against field of endeavour
Movements like "ethical source" aim to prevent software from being used for certain purposes, such as for military use or use by private capitalist corporations.

Non-free software

Important! The terms “open source” and “free software” are used according to the definitions given here, and no others. Beware software that claims to be free or open without preserving your rights, and beware someone who wants you to believe a new interpretation which is more aligned with their financial interests. FOSS is a very successful brand and some people try to take advantage of it without playing by its rules.

How does this work?

Free software is enabled through the use of **free software licenses**.



It's like one of these, except, instead of imposing a bunch of restrictions on you, it guarantees a bunch of *rights* for you.

Though, to be fair, often it also imposes restrictions on you.

Structure of a free software license

A software license explains your rights and obligations with respect to a piece of software. A free software license in particular establishes the following:

- The four freedoms
- Optionally any additional freedoms, such as patent grants
- Obligations, such as attribution or copyleft requirements

Often included is also a *disclaimer of warranty*.

Read the license! It's not that bad. In fact...

The MIT license

The MIT license, continued

The software is provided as is, without warranty of any kind, express or implied, including but not limited to the warranties of merchantability, fitness for a particular purpose and noninfringement. In no event shall the authors or copyright holders be liable for any claim, damages or other liability, whether in an action of contract, tort or otherwise, arising from, out of or in connection with the software or the use or other dealings in the software.

Differences between free software licenses

“Liberal” licenses: No requirement to make derivative works free software

- WTFPL
- MIT, BSD
- Apache 2.0

“Copyleft” licenses: Requirement to make derivative works free software

- MPL
- LGPL
- GPL
- AGPL

Specialized licenses

For works which are not software:

Multimedia: Creative Commons (*partially non-free!*) – copyleft and liberal
Documentation: GNU Free Documentation License – copyleft
Fonts: Open Font License – liberal

Liberal licenses in detail

"Liberal" licenses are generally very similar, and any of them is generally a good choice.

MIT or **BSD**: Sane default choice for individual or community-maintained liberal projects. Short enough to read in 1-2 minutes: check which of these suits your needs.

Apache 2.0: More commercial friendly, includes clearer provisions for trademark and patent use. Good choice for businesses.

Copyleft licenses: GPL family

Copyleft licenses are more distinct from each other and require more careful consideration.

GNU General Public License (GPL): Recommended for programs

GNU Lesser General Public License (LGPL): Recommended for most libraries

GNU Affero General Public License (LGPL): Recommended for internet services

The main difference between these relates to *distribution* and "*virality*".

Mozilla Public License

The Mozilla Public License (MPL) is a copyleft license with file-based "virality" semantics rather than project-based. This makes it easier to work with libraries in newer languages, which use static linking or vendoring.

Which licenses do I use?

I generally use the following licenses:

Software: GPL, AGPL, or MPL, as appropriate
Multimedia: Creative commons

Occasionally MIT or BSD if I want universal adoption, or WTFPL or CC-0 if I don't care.

How do you apply a license to your work?

Simple approach: Find the plaintext version of the license you want to use and add it to your repository in a file called COPYING or LICENSE.

Caveats: Certain licenses such as AGPL call for you to link to the source code and license information on a website or similar. Some people like to add license headers to their code (recommended for MPL).

```
// License: MPL-2.0
// (c) 2022 Armin Preiml <apreiml@strohwolke.at>
// (c) 2022 Drew DeVault <sir@cmpwn.com>
```

Using free software to make more free software

If you're cool with the same license: Update the copyright headers as appropriate.

```
$ cat COPYING
Copyright (c) 2017-2022 Drew DeVault
Copyright (c) 2014 Jari Vetoniemi

Permission is hereby granted, free of charge, ...
```

Using free software to make more free software

If you use a different license or need to incorporate multiple works: You need to consider sublicensing and license compatibility.

Liberal licenses: As a rule of thumb, most popular liberal licenses can usually be combined with any other work, regardless of license, so long as the terms (e.g. attribution) are upheld.

Copyleft licenses: Copyleft licenses can be more complicated to integrate with each other, and their virality clauses require additional considerations for compliance.

For GPL compatibility, see Various Licenses and Comments about Them
`https://www.gnu.org/licenses/license-list.html`

Reusing liberally licensed software

If your software is free software: Include all of the applicable licenses and copyright statements in your source code, alongside your own.

If your software is non-free: Distribute all applicable licenses and copyright statements to your end-users.

For more complex scenarios: Consider adopting the REUSE specification.
`https://reuse.software/tutorial/`

Reusing copyleft software

The **combined work** becomes copyleft and the combined set of obligations from all applicable licenses are in force. The licenses must be *compatible*, meaning that they don't have contradictory terms: no mixing GPL and CDDL, for example.

The liberally licensed parts remain liberally licensed, and may be developed independently. However, these parts, combined with copyleft parts, are also copyleft in context.

Non-free software is well-advised to steer clear of incorporating copyleft works.

Changing a free software license later

The sublicensing approach: Projects with a liberal license can simply change it with little fanfare. The old license applies to old code and the new license applies to new code. This is similar to incorporating the old code into a new work.

The re-licensing approach: To change the license of the old code, you must receive written consent from all authorized copyright holders in the project, i.e. all prior contributors. This approach is *required* for removing copyleft provisions.

In either case, anyone who recieved a copy of the old code under the previous license can continue to treat it as if only the old license applies.

Copyright assignment

One approach to re-licensing is to get contributors to agree upfront, prior to their contribution, commonly through a **Contributor License Agreement**. These either directly assign copyright over the contributor's work to a single stakeholder, or give a single stakeholder the right to unilaterally change the license.

This is a dick move. Pulling the rug out from under a free software project to change its license later is a violation of the social contract of free software. I strongly recommend that you, as a contributor, refuse to sign CLAs which leave this door open.

The politics of license changes

A software license establishes a legal and social contract and a set of expectations shared with the community, between the leadership, contributors, users, and others. Changing that dynamic is a serious decision.

Only change your license when it benefits the community as a whole, rather than a select few stakeholders. Free software does not belong to you — it belongs to *us*.

Further reading on my blog

Open Source means surrendering your monopoly over commercial exploitation
`https://drewdevault.com/2021/01/20/FOSS-is-to-surrender-your-monopoly.html`

Open Source is defined by the OSI's Open Source Definition
`https://drewdevault.com/2022/03/01/Open-source-is-defined-by-the-OSD.html`

Free software licenses explained: MIT
`https://drewdevault.com/2022/02/07/Free-software-licenses-MIT.html`

On commercial forks of FOSS projects
`https://drewdevault.com/2021/12/18/Commercial-forks-of-FOSS-projects.html`

The complete guide for open sourcing video games
`https://drewdevault.com/2021/03/23/Open-sourcing-video-games.html`