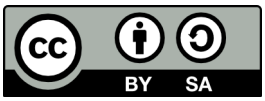
A Philosophy of Free and Open Source Software

Drew DeVault



Contents

Chapter 1

Introduction

When you hear "free software" or "open source", a few ideas may come to mind. You might be a user of free software, and think of prominent projects like Linux or Firefox, and your experiences with them. You may recall some of the social tools utilized by free software projects, such as SourceHut or GitHub or Bugzilla or IRC, some of which are themselves free software, and some of which aren't, and the social systems which surround them. Free and Open Source Software, or "FOSS", is an endlessly complex and diverse system of people and software and politics, and one could imagine that the essence of "free software" or "open source" is found somewhere within the sum of these subtle interactions.

In fact, the essence of free and open source software is simple. This book aims to reveal this essence to you, and explain it in detail. We will use it to develop a rational explanation for why the FOSS ecosystem is the way it is, and to offer a rational argument for what this means for you as a participant in this system. Whether you are a user, contributor, maintainer, business, or some combination of these — and, by the end of this book, you very well may be all four — I hope to offer you some insights which are directly applicable to your work.

The terms "free software" and "open source" are, in truth, largely identical, but there are subtle differences. What do these words actually mean?

1.1 What is free software?

One could go into great depth about free software, and I recommend reading the wealth of literature from the Free Software Foundation and the GNU project on the subject if you care to explore those depths. For our purposes, a short summary will suffice.

Free Software is defined by the four essential freedoms, paraphrased thusly:

0. The freedom to run the program as you wish, for any purpose.
1. The freedom to study how the program works, and change it as you wish.

2. The freedom to redistribute copies so you can help others.

3. The freedom to distribute copies of your modified versions to others.

Software which meets this criteria, generally through the use of an appropriate software license, qualifies as Free Software, and software which does not meet this criteria does not qualify as Free Software.

The criteria are, to some degree, vague, and subject to interpretation. The authors of these criteria have provided some of these interpretations for us. For example, freedom zero does not tolerate the exclusion of any use-cases, such as commercial or military applications. The freedom to study how the program works, and to change it as you wish, presupposes access to the source code, and cannot, for example, be fulfilled with reverse engineering. The blessed interpretation of the second freedom also extends to commercial redistribution.

Distributors of free software may also impose additional obligations, so long as they do not conflict with these freedoms. A common example is prohibiting the use of a trademark to endorse any software which you have modified. Another may be found in the copyleft family of licenses, which famously *require* you to exercise freedom 3 by providing your modifications under similar license terms to anyone who receives your work.

The commercial potential of "free" software has been the subject of some confusion, as "free" often means "free of charge" in other contexts; in this case it means "free as in freedom".

1.2 What is open source?

A competing ideology can be found in the "Open Source" movement, stewarded by the Open Source Initiative. Inspired in part by the confusion "free" software introduces to a commercial interpretation, the movement offers an alternative framework for reasoning about software freedoms.

This is primarily manifest in the Open Source Definition, a document which formally lays out the criteria for evaluating Open Source software. It is much more specific and less subject to interpretation than the Free Software definition. This is a deliberate choice, in order to organize a movement which is more approachable for commercial participants.

The practical differences between free and open source software, however, are subtle enough that, for all intents and purposes, they may be treated as the same. The respective stewards of these ideologies maintain official lists of software licenses which qualify under their frameworks, and except for a small number of obscure licenses, their respective lists are identical.

Though this distinction between free and open source software can be subtle, the underlying philosophical differences are not. Free software is, at its heart, a political and philosophical movement, and open source strives to be apolitical. As this is a political, philosophical book, I will prefer

to use the term "free software" throughout.

I understand that these explanations are, in truth, a bit dense. It is difficult to understand how these high philosophical ideals, their interpretations, and legal tools like licenses and copyright, give rise to the complex FOSS ecosystem you see today. We'll now depart from these philosophical roots to explain the FOSS ecosystem at a higher-level, in terms more relatable to your personal experiences with it. But, as we move on, keep in mind that everything we're talking about is ultimately derived from the principles laid out in these first three pages — nothing more, and nothing less.

There are some who wish that these principles were not so, or wish you to believe an unorthodox interpretation of them. Often this is in the service of their own interests; many commercial entities in particular would be pleased to capitalize on the smashing success of free and open source software without meeting the obligations implied. I will discuss alternatives to free software philosophy in later chapters, but I urge you to take care in your use of the terms "free software" and "open source" for this reason.

1.3 Dramatis Personae

Free software, much like Soylent Green, is made of people. You, dear reader, are one such person! Allow me to introduce the rest of the cast.

Users are the beneficiaries of free software. This is the person sitting at the keyboard or tapping at their phone, directly interacting with software. A user may also be a programmer, or an organization the programmer is a member of, who is making use of free software tools like development libraries or database engines.

Maintainers take on formal responsibilities in a free software project, though the extent of these responsibilities can vary. These are the people who are ultimately responsible for an *upstream*, and they make decisions on behalf of the project, such as what changes go into each release. This is a technical role.

Contributors do not take on formal responsibilities in a free software project, but generally maintain a long-term relationship with the project (though "drive-by" contributors may only participate in the short term). They make tangible improvements to a free software project, such as sending patches to update the code or improve the documentation.

Participants are all of the above: anyone who engages with the social systems surrounding free software. This includes users who report bugs, experts who offer support in project chat rooms, programmers and designers who participate in building free software, and maintainers who take on special responsibilities like release management.

There are others, to be discussed later. This includes formal organizations like **corporations** and **non-profits**, **upstreams** and **downstreams**, narrower roles like **moderation** or **outreach**, or

adjacent roles like **distributions**.

I will tell you outright that the ultimate purpose of this book is to encourage you, the reader, to take on these roles — ideally more than one. I want you to feel comfortable stepping into any of these roles as the occasion demands, and to be able to understand and empathise with people in these roles when you interact with them.

In fact, a productive relationship with free software *requires* this.

Chapter 2

Free software licenses

Software licenses are the primary means of putting free software's ideas into practice, and it's important that you understand the licenses you will encounter in the free software ecosystem. The nature of these licenses not only offers you legal rights and imposes upon you legal obligations, but also influences the relationship between the participants and hints at the goals of the project and its leadership.

The right to copy ideas is inherent; ideas are not scarce and copying does not consume finite resources like the transfer of material property does. In a perfect world, licenses would not be necessary to govern this. However, an idea of ideas, or an anti-idea, arose to disrupt this process. Intellectual Property is a legal concept which imposes an artificial scarcity on ideas, allowing an individual or organization to monopolize them for commercial profit. Additional legal systems have been devised to further constrain these rights, such as the anti-reverse-engineering legislation in the Digital Millennium Copyright Act (DMCA), or the patent system, which concerns "inventions".

These anti-ideas are at odds with the goals of free software, and a different understanding of intellectual property and commercialization is required to understand free software. The second freedom, the freedom to redistribute copies or modifications, is naturally in conflict with copyright law. Software licenses are designed to utilize copyright law against itself, to allow our software to be freely shared. Perhaps one day, copyright reform will render licenses obsolete. Until that day comes, we must work within the constraints of the societies of which we are a part. We also use licenses to apply further constraints, such as disclaiming liability over the software, or enforcing the proliferation of free software via copyleft. Essentially, licenses allow us to opt-out of copyright, or to repurpose it to facilitate the praxis of free software philosophy.

Through licenses, ownership over free software projects is (usually)[1] shared with their com-

[1] A subset of free software utilizes tools like a Contributor License Agreement to change this paradigm, usually for commercial reasons. I will address this later; it will suffice for now to say that I do not like this idea.

munities. Each contributor receives a free software license to use the project, and provides their contribution under the same terms, without giving up their copyright. Thus, the project is collectively owned by its contributors, and questions over how to utilize this copyright require the consent of the community. Additionally, users (non-contributors) who receive a copy of the software receive a perpetual free software license for that version, and should the upstream community choose to leverage their copyright in a way the user dislikes, they retain the right to use, modify, and redistribute the software they received regardless. As such, licenses create a framework of rights which is designed to preserve the four essential freedoms for all participants.

Licenses can also expand this framework to suit certain goals, and these extensions create a taxonomy of licenses called "families". Some licenses, collectively called "liberal" or "permissive" licenses, keep these additions narrow, such as limiting the use of trademarks or disclaiming liability. The "copyleft" family of licenses goes further by adding clauses that require licensees to provide their modifications to users under similar license terms if they exercise the third freedom — the so-called "viral" clause. Between these two options exists a spectrum in which various licenses use different licensing terms in the service of different goals, and thus appeal to different groups of users and contributors within the free software ecosystem.

Note

> There are some allegiances present along this spectrum. The Free Software movement is more associated with copyleft, and the Open Source movement is more associated with permissive licenses. Additionally, commercial users tend to prefer permissive licenses, which generally make monetization of the software easier. These are trends, but not rules: free software includes permissive licenses and open source includes copyleft, and commercial users are found all along the spectrum.

In this chapter, we will introduce a number of popular licenses across this spectrum, and explain their goals and audience. We will seek to establish in you an understanding of how the software licenses you are subject to work, the rights they provide you, and how to comply with their obligations, and help you understand how to choose the right license for your own projects.

Though I will do my best to explain these licenses to you, I am not a lawyer and have no formal legal expertise. I am a hacker – I wouldn't be caught dead wearing a suit. If you are working with licenses in a commercial environment, or are unsure of your interpretation of a software license, or feel that your rights are being infringed upon, I advise you to consult a lawyer for help. You can find one by reaching out to your local bar association (or regional equivalent); they will refer you to a suitable professional. I understand that it is entirely unhackish to associate with lawyers, but they are a necessary evil under the thumb of our legal system. Contemporary research suggests that lawyers are people, too[citation needed], so let's give them a fair chance.

2.1 Permissive Licenses

Let's examine more closely the "permissive" family of software licenses. For our purposes, we will define this family as those licenses which permit the recipient to modify the software, but do not require them to share their changes under a similar license. The licenses we'll cover are the MIT/X11 license, various BSD licenses, and Apache 2.0. There are many others, but these are the most common. How do they work?

2.1.1 The MIT license

The MIT/X11 license is one of the most popular permissive licenses, as well as one of the most popular free software licenses in general. It's very short and easily understood. I have reproduced it here in its entirety, and I encourage you to read it yourself.

> Copyright © <year> <copyright holders>
>
> Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the Software), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:
>
> The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.
>
> 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.

Let us ask if this license fulfills the four freedoms. It provides for freedom zero, the right to use the software for any purpose, in the first sentence, by granting permission, free of charge, to any person who obtains a copy of the software to "use" it or "copy" it. Freedom one, the right to study and change the program, is accomodated by including the source code with the license, and by offering you the right to "modify" or "merge" it. It allows for redistribution, including commercial redistribution, by granting the right to "distribute" and "sell" copies of it. The fourth freedom, the right to distribute your modified versions to others, is not explicitly addressed, but is possible

thanks to granting, "without limitation", the right to copy, modify, and distribute the software. We conclude that this license fulfills the four freedoms: it is a free software license.

What else does it do?

The rights granted are subject to one condition: you must include the license text and the copyright notice in any copy of or substantial portion of the software. This kind of clause is called an "attribution" clause. Anyone who copies or distributes any portion of the software, perhaps by incorporating portions of it into their own software, must meet this obligation.

The license continues in all capital letters, suggesting that the authors found this next bit particularly important. Let's take extra care in reading and understanding it correctly. The final clause is a disclaimer of warranty, and sets the expectations for your relationship with the entity you received the software from — namely, that you cannot expect anything at all.

This is an important point to understand about free software. It comes with no warranty of any kind, and no guarantees of fitness for purpose. It is provided "as is". This means that you are not entitled to support or a working product. You are entitled only to the software "as is" – that is, the way it was given to you – and nothing more. If it does not work correctly, or is not suitable for your needs, you cannot hold the authors liable for this. This arrangement is ubiquitous among all free and open source software licenses.

The implications are profound. We will examine this in great depth in later chapters.

For the time being, I will share only one of these implications: that the people you work with in free software do not have to be there. Free software is made of people, and those people are usually volunteers. You may find them in chat rooms, on mailing lists, and around bug trackers, offering you help. Remember to treat these people with respect, gratitude, and professionalism, because you are not entitled to their attention, and those who are rude or disrespectful are denied this privilege.

2.1.2 The BSD family

The BSD family of licenses is another popular permissive approach, created for the Berkeley Software Distribution, a family of Unix-compatible operating systems. Originally, this license had four conditions, but several variants exist which differ only in which conditions they include.

Like the MIT license, the BSD family is very simple and easy to read. I've included the full text of the "3-clause" BSD license here:

Copyright <year> <copyright holder>

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The 3-clause variant is the most common BSD license in modern use. An earlier "4-clause" BSD license is essentially obsolete. It included a controversial "advertising" clause:

3. All advertising materials mentioning features or use of this software must display the following acknowledgement: This product includes software developed by the <copyright holder>.

In addition to 3-clause BSD, other variants are still in relatively common use. The "2-clause" license omits the non-endorsement clause, allowing the recipient to use the product name without prior written permission. This variant is also called the "Simplified" BSD license, and is functionally equivalent to the ISC and MIT licenses.

A "zero clause" variant also exists, which drops even the attribution clause, leaving only the disclaimer of warranty. This version is known as "0BSD", and is in an ultra-permissive family of

licenses which are considered "public domain equivalent". We'll talk about this license family in more detail under "specialized licenses" later on.

2.1.3 Apache 2.0

The last "permissive" license that we'll be looking at is Apache 2.0. This license is, as the name suggests, the favored license of the Apache Foundation. The Apache 2.0 license carves out some special considerations for trademarks and patents, and has enjoyed popularity among free software projects maintained by commercial entities as such.

This license is a bit too large to include outright, so we'll just quote a few important sections for further study. Despite its somewhat longer length, Apache 2.0 is a reasonably easy license to read, so I encourage you to seek it out yourself and read through it.

There are a few key details which make Apache 2.0 attractive to corporations:

- Apache 2.0 begins with a list of definitions, unambigiously defining terms like "source", "contribution", and "derivative work".
- An explicit grant of copyright license is provided, as well as a grant for all applicable patents, as well as an escape hatch in the event that the recipient leverages its own patents. This is a smart move when either party holds patents, and protects both parties. However, Apache 2.0 explicitly *does not* grant use of the copyright holder's trademarks.
- Contributors explicitly agree to license their contributions under the Apache 2.0 terms, and a system of accountability is established for tracking copyrights and changes across forks and vendors.

Apache 2.0 also makes explicit some unstated assumptions about how the disclaimer of warranty works in free software, the full implications of which will be covered later.

There are some important differences between the Apache 2.0 license and the others we've discussed in the way that the license is used. The four freedoms are upheld, but with some constraints on how to exercise them. Every source code file in an Apache 2.0 licensed project generally opens with a large comment which includes a boilerplate declaration of the file's license, along with a list of copyrights. You'll see something like this in practice:

```
// Copyright 2021 Drew DeVault
// Copyright 2016-2019 Jane Doe
// Copyright 2014 Foobars, LLC
//
// Licensed under the Apache License, Version 2.0 (the "License");
// you may not use this file except in compliance with the License.
// You may obtain a copy of the License at
//
//     http://www.apache.org/licenses/LICENSE-2.0
//
// Unless required by applicable law or agreed to in writing, software
// distributed under the License is distributed on an "AS IS" BASIS,
// WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
// See the License for the specific language governing permissions and
// limitations under the License.

#include <stdio.h>
#include <unistd.h>
...
```

When modifying this file, complying with Apache 2.0 generally involves adding your copyright to the list. Thus, the copyrights of the file are always accounted for.

2.2 Copyleft licenses

On the opposite end of the spectrum lie the "copyleft" family of licenses, so-called as they provide for an ideology which is the opposite of "copyright". This is emphasized by their symbol, ↄ, the reverse of ©.

The defining trait of a copyleft license is the requirement that any derivative works are *also* made available using a copyleft license. Thus, anyone who incorporates a copyleft work into their own projects is under some degree of obligation to make their modifications public. Some of these licenses also include a "viral" clause, which "infects" the entire work and obligates the author to publish the whole thing under a copyleft license.

The main means by which copyleft licenses distinguish themselves from one another is in how much of the derivative work is required to be copyleft, and in how a "derivative work" is defined. There are two license families I'm going to address in this section: the GNU Public License, and the Mozilla Public License. I will also briefly cover Oracle's Common Development and Distribution License.

2.2.1 The GNU Public License

GNU is an organization closely tied with the Free Software movement, and they work together with the Free Software Foundation to maintain their licenses. GNU is responsible for the GNU Operating System, which, in its most popular form, manifests as the userspace utilities on GNU/Linux systems. The GNU family of licenses is designed in service of this organization's political goals: furthering the growth and ubiquity of free software. These licenses are specifically designed to proliferate free software, and are to prevent free software from being incorporated into nonfree derivative works.

There are several licenses published by GNU, all of which are "viral". The ones which are relevant today are:

- General Public License 2.0 (GPL 2.0)
- General Public License 3.0 (GPL 3.0)
- Lesser General Public License 3.0 (LGPL 3.0)
- Affero General Public License 3.0 (AGPL 3.0)

GNU also maintains the GNU Free Documentation License, which will be covered under specialized licenses later. GPL 2.0 is considered a legacy license by GNU, and is discouraged for new projects, but it remains relevant thanks to the Linux kernel. These licenses are distinguished by differences in the qualifications for a "derivative work". Generally speaking, this distinction orders the licenses from LGPL to GPL to AGPL, such that the LGPL "virally" infects the fewest derivative works, and AGPL the most.

These licenses are rather complicated and raise a lot of interesting questions. If you are using GPL-covered works to create more GPL-covered works, you are generally safe with a surface-level understanding of these licenses. However, if you want to mix GPL-covered works with non-covered works, things can get complicated. I would advise anyone in this situation to review the official GNU FAQ, or consult a lawyer.

https://www.gnu.org/licenses/gpl-faq.html

Derivative works of GPL software

Put simply, the usage of these licenses generally falls along these lines:

- **GPL** is used for computer programs.
- **LGPL** is used for programming libraries.
- **AGPL** is used for internet services.

The GPL and LGPL define a derivative work in terms of linking (i.e. the job facilitated by the linker, **ld**). The GPL "infects" any code with which it is linked, period. This includes copying and pasting GPL-covered source code into your program, or preparing it as a static or shared library and linking to it.

The LGPL is less strict in this requirement, allowing you to, under certain conditions, link a LGPL-covered work with another work without triggering the "viral" clause. The caveats are mainly concerned with providing for the user's ability to modify the LGPL-covered work and incorporate their modified version into your program.

The AGPL, however, is more strict. This license was introduced in the age of cloud services, which were able to circumvent the constraints of the GPL and LGPL by running covered works *on the user's behalf*, transmitting their requests to a private server, then transmitting the results back to the user. AGPL closes this "loophole" by entitling the user to the modified source code under these circumstances.

GPL 3.0 "or later" and FSF copyright assignments

Another interesting distinction of GPL licenses that you may want to be aware of is the use of the following phrase:

> 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 is interesting: it allows the user to choose to apply the GNU Public License version **4.0** to your project, instead of 3.0, should such a license ever be written.

The motivation for this choice comes from the revisions made from GPL 2.0 to 3.0 to close the so-called "Tivoization" loop-hole. The "Tivo" company made home entertainment systems using Linux, for which they dutifully provided their modifications to users who were entitled to them. However, they did *not* make it possible for users to install their own modified versions of Linux on Tivo devices. Linux uses GPL 2.0 to this day, and the loophole remains open with no clear means of closing it. To avoid this scenario playing out for any future loopholes that might be found in version 3.0, the use of this phrase has been encouraged by the Free Software Foundation so that any such loopholes can be closed retroactively by version 4.0.

For this to work correctly, users of the GPL must trust the Free Software Foundation to protect our works unconditionally in the future, and to forever uphold the rights they believe in.

Some projects trust the Free Software Foundation to an even greater degree with the use of a copyright assignment. The FSF encourages GPL users to assign their project's copyright to the FSF,

so that they can use their resources to defend against GPL violations in court.

There was a time when both of these recommendations were applied by most projects, but that time has passed, and many projects are no longer choosing to exercise these options. You may make your own personal determination. For my part, though I use the GPL family of licenses, I do not include the "or later" clause, and I do not assign my copyright to the FSF. The FSF of yesterday may not be the FSF of tomorrow, and I would prefer not to give the FSF of tomorrow the right to unconditionally relicense my works. Other approaches to GPL enforcement are also gaining in popularity, and the FSF has not always been there when enforcement was needed.

GPL compatibility

TODO

2.2.2 The Mozilla Public License

TODO

2.3 Special-purpose licenses

TODO

- GNU FDL
- Open Font License, Creative Commons
- CC-0, 0BSD
- WTFPL, Unlicense

2.4 How I choose the right license for my own projects

TODO