Logiciel Libre
Cours 1 — Introduction

Stefano Zacchiroli
zack@irif.fr

Laboratoire IRIF, Université Paris Diderot

2019–2020

URL https://upsilon.cc/zack/teaching/1920/loglib/

Outline

Outline

What is free software?

Free Software guarantees to its users 4 fundamental freedoms:

- Freedom #0, to run the program, for any purpose
- Freedom #1, to study how the program works, and change it
- Freedom #2, to redistribute copies
- Freedom #3, to improve the program, and release improvements

`http://www.gnu.org/philosophy/free-sw.html`

In other words, *if* you get a piece of Free Software, you can:

- use it
- study and adapt it [1]
- redistribute it as is
- improve it and release improvements

1. access to source code is a requirement for this

Free Software v. Open Source

This definition (AKA the "Free Software Definition") is by the Free Software Foundation and, in its original version by Richard Stallman (1986), introduced the notion of Free Software

- it is grounded in ethical/moral values
- it focuses on user freedoms, that are obtained as a result of adopting Free Software principles

A related notion is that of Open Source, which is embodied in the "Open Source Definition" (OSD)[2] by the Open Source Initiative (1998)

- it focuses on practical aspects (e.g., software quality, efficiency of the development process)

2. `http://opensource.org/docs/osd`, discussed next

Open Source Definition (OSD) — Foreword

Open source doesn't just mean access to the source code. The distribution terms of open-source software must comply with the following criteria:

The text of these slides is a verbatim copy of the Open Source Definition (annotated version), as published by the Open Source Initiative [3]

3. https://opensource.org/osd-annotated

OSD — 1. Free Redistribution

The license shall not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license shall not require a royalty or other fee for such sale.

Rationale: By constraining the license to require free redistribution, we eliminate the temptation to throw away many long-term gains in order to make a few short-term sales dollars. If we didn't do this, there would be lots of pressure for cooperators to defect.

OSD — 2. Source Code

The program must include source code, and must allow distribution in source code as well as compiled form. Where some form of a product is not distributed with source code, there must be a well-publicized means of obtaining the source code for no more than a reasonable reproduction cost preferably, downloading via the Internet without charge. The source code must be the preferred form in which a programmer would modify the program. Deliberately obfuscated source code is not allowed. Intermediate forms such as the output of a preprocessor or translator are not allowed.

Rationale: We require access to un-obfuscated source code because you can't evolve programs without modifying them. Since our purpose is to make evolution easy, we require that modification be made easy.

OSD — 3. Derived Works

The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.

Rationale: The mere ability to read source isn't enough to support independent peer review and rapid evolutionary selection. For rapid evolution to happen, people need to be able to experiment with and redistribute modifications.

OSD — 4. Integrity of The Author's Source Code

The license may restrict source-code from being distributed in modified form only if the license allows the distribution of "patch files" with the source code for the purpose of modifying the program at build time. The license must explicitly permit distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software.

Rationale: Encouraging lots of improvement is a good thing, but users have a right to know who is responsible for the software they are using. Authors and maintainers have reciprocal right to know what they're being asked to support and protect their reputations. Accordingly, an open-source license must guarantee that source be readily available, but may require that it be distributed as pristine base sources plus patches. In this way, "unofficial" changes can be made available but readily distinguished from the base source.

OSD — 5. No Discrimination Against Persons or Groups

The license must not discriminate against any person or group of persons.

Rationale: In order to get the maximum benefit from the process, the maximum diversity of persons and groups should be equally eligible to contribute to open sources. Therefore we forbid any open-source license from locking anybody out of the process. Some countries, including the United States, have export restrictions for certain types of software. An OSD-conformant license may warn licensees of applicable restrictions and remind them that they are obliged to obey the law; however, it may not incorporate such restrictions itself.

OSD — 6. No Discrimination Against Fields of Endeavor

The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business, or from being used for genetic research.

Rationale: The major intention of this clause is to prohibit license traps that prevent open source from being used commercially. We want commercial users to join our community, not feel excluded from it.

OSD — 7. Distribution of License

The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.

Rationale: This clause is intended to forbid closing up software by indirect means such as requiring a non-disclosure agreement.

OSD — 8. License Must Not Be Specific to a Product

The rights attached to the program must not depend on the program's being part of a particular software distribution. If the program is extracted from that distribution and used or distributed within the terms of the program's license, all parties to whom the program is redistributed should have the same rights as those that are granted in conjunction with the original software distribution.

Rationale: This clause forecloses yet another class of license traps.

OSD — 9. License Must Not Restrict Other Software

The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programs distributed on the same medium must be open-source software.

Rationale: Distributors of open-source software have the right to make their own choices about their own software.
Yes, the GPL is conformant with this requirement. Software linked with GPLed libraries only inherits the GPL if it forms a single work, not any software with which they are merely distributed.

OSD — 10. License Must Be Technology-Neutral

No provision of the license may be predicated on any individual technology or style of interface.

Rationale: This provision is aimed specifically at licenses which require an explicit gesture of assent in order to establish a contract between licensor and licensee. Provisions mandating so-called "click-wrap" may conflict with important methods of software distribution such as FTP download, CD-ROM anthologies, and web mirroring; such provisions may also hinder code re-use. Conformant licenses must allow for the possibility that (a) redistribution of the software will take place over non-Web channels that do not support click-wrapping of the download, and that (b) the covered code (or re-used portions of covered code) may run in a non-GUI environment that cannot support popup dialogues.

Free Software v. Open Source (cont.)

What makes a software free and/or open source is its license.

- to be considered Free Software, a software should be released under a license that guarantees the 4 freedoms to its users
  - to help determining that, the FSF maintains a list of Free Software licenses [4]
- to be considered Open Source, a software should be released under a license that respects the Open Source Definition
  - OSI is the body in charge of determining license adherence to the OSD and maintains a list of OSI-approved open source licenses [5]

Many licenses exist: GPL, LGPL, MIT, BSD, Apache, MPL, ... [6]

4. https://www.gnu.org/licenses/license-list.en.html
5. https://opensource.org/licenses
6. we will discuss many of them extensively in a dedicated class

Free Software v. Open Source (cont.)

The license *lists* by FSF and OSI are almost identical and coincides on all popular licenses.

- in practical terms it doesn't matter if you use "free software" or "open source", user freedoms are guaranteed anyhow
- the difference is relevant though if you want to highlight the moral/ethical aspects about free/open source software

Terminology

Some terminological difficulties:

- "free software" v. "open source"
- "*free* software" is not "*gratis* software"
  "free" as in "free speech" not as in "free beer"

As a result:

- "libre" often used instead of (or in addition to) "free", also in English-speaking contexts
- it is common to use FOSS (Free/Open Source Software), or FLOSS (Free/Libre/Open Source Software) as comprehensive acronyms

In this course:

- we use "Free Software" and "Open Source" as synonyms (default)
- . . . with exceptions when discussing moral aspects and history
- in French, we use "logiciel libre"

Outline

There is a new player in town

- Android, GNU/Linux, Apache, Wordpress, OpenStack, etc. are very important software products, but...
- The actual novelty is in the free software model:
  - Unprecedented combination of collaboration and competition.
  - Shift in emphasis from marketing to support and quality.
  - Classical assumptions about intellectual propriety are questioned.
  - End-users recover control (from software providers)
  - A new model for a new (global, networked) world?
- Recent years have shown the feasibility of the model.

Moral and economic challenges

For the first time in human history, we face an economy in which the most important goods have zero marginal cost. And the transformation to digital methods of production and distribution therefore poses to the twenty-first century a fundamental moral problem.

If I can provide to everyone all goods of intellectual value or beauty, for the same price that I can provide the first copy of those works to anyone, why is it ever moral to exclude anyone from anything?

If you could feed everyone on earth at the cost of baking one loaf and pressing a button, what would be the moral case for charging more for bread than some people could afford to pay? This represents the difficulty at which we find ourselves straining at the opening of the twenty-first century.

— *Eben Moglen*

When free software enters a new niche...

- It can become one of the best technical choices
  - Android in smartphones, Apache in web servers, Wordpress in blog engines, Jupyter in data science, TensorFlow in machine learning, Linux in IoT and super computers, etc.

When free software enters a new niche...

- It can become one of the best technical choices
  - Android in smartphones, Apache in web servers, Wordpress in blog engines, Jupyter in data science, TensorFlow in machine learning, Linux in IoT and super computers, etc.
- It benefits from many synergies
  - reuse of code, reuse of knowledge, reuse of distribution channels, etc.
- Users gain competitive advantage:
  - Availability of source code makes improvements and customization possible at large scale (by in-house or subcontracted teams).
  - Standardization, but maintaining competition between providers.
  - No more per-use licenses.
  - More/better support opportunities (thanks to competition).
- Competition is the name of the game.

Consequences for the software industry

The software business is turning upside down (still slowly, but gaining momentum):

- Traditional software "manufacturers" will have to reinvent themselves completely (no more per-copy incomes).
- A whole new industry (based on support and FOSS development) will be needed as free software gains market acceptance.
- It allows for (and encourages) competition in support, and even in the evolution of a piece of software.
- Users benefit in several ways. Therefore, big pressure from end-users (including big companies) to switch to free software.

Some specific impacts

- Cost: cost model radically different from proprietary software
- Openness: can be modified, can be inspected, can be studied
- Distribution: new distribution channels, new methods
- Development: “surprising” development models
- Maintenance and support: true competition

Mixture of two powerful mechanisms:

- Competition (using the same source base)
- Cooperation (even non-voluntary)
- Competition + Cooperation = Coopetition

Outline

How free software affects...

- End users (individual or company)
- Developers (or software producer)
- Integrators
- Service and maintenance providers

End user

End users can forget about...

- ...company monopolies
  (real competition, best products and services)
- ...producer "reliability"
  (future path ensured by product acceptance, source code availability, community dynamics)
- ...decision making based on incomplete information
  (software can be tested in real environments, with near-zero cost)
- ...dependence on provider's strategies
  (many providers, community strategies, strategies follow clients)
- ...black boxes
  (no longer "blind confidence")

End user (2)

What if users could...

- . . . adapt/customize the product at will?
- . . . have the latest release with (very) low cost?
- . . . fix all the problems (or hire someone to fix them)?
- . . . decide on the future evolution of the product?
- . . . contract the (complete) integration of the best products in a given area?
- . . . buy complete auditing for each product by independent third parties?

End user (3)

A significant amount of control shifts from software producers to users

Free software for (large) end users

- Free software is not necessarily better or worse. It is just different
- In several niches, we have already excelent products and companies supporting them
- In many cases, the most cost-effective way of producing software
- Special advantages when there is interest in long-term life cycles, vendor independence, multiplatform support, adaption to evolving technologies
- If a powerful enough user (or group of users) needs to drive the technology, this is probably the best way to go
- Many things can be done to promote a competitive free software industry in a given niche. Many benefits are derived of such a promotion

Developer (or software producer)

Free software changes the rules of the game:

- Opportunities for competing while being small
- Easier (and cheaper) to acquire front-wave technology
- Can take advantage of the work of your competitors (but they can do the same!)
- External contributors can be found (in many cases, at a fraction of the usual cost, because of win-win relationships)
- Distribution channels are cheaper, and truly global
- Feasible to become reference application in a niche

Developer (or software producer) (2)

Where does the money come from? (sustainability)

- The producer enjoys the best knowledge about its product
- Producer can be the “most visible point”, if their image is properly curated
- Custom-made development, modifications, customizations
- “In depth” support (bug fixing, preference in access to new releases, new features)

Assuming there is a need for a software product and money to pay for related support, developers/service providers will benefit (assuming they can reach out to the relevant customers).

Integrator

Maybe the best-positioned actor:

- Free software products are largely available (without the constraints of proprietary licences!)
- If products “don’t fit” you can adapt them (source code is available, interoperability is always possible)
- Pieces of products, or full products, or anything in the middle, can be integrated
- No more black boxes: everything is transparent

Integrators can build on top of the work of others, with similar constraints and opportunities

Services and maintenance

- Similar conditions than the producer
- Competition in the maintenance business
- Added-value of services is better appreciated (the base cost of the program is low)
- Good knowledge of the state of the art is important (good idea to have good links with free software projects)
- New business models: advising on releases and combination of programs, information about new development, project management, etc.
- The most diverse and massive kind of business

Some conclusions

- Free software changes the rules of the game
- It is important to understand (and take advantage) of those rules
- We are still learning effects and mechanisms
- Many opportunities to discover new effects, and take advantage of them

Outline

"Free software cannot be used commercially"

"*Free software cannot be used commercially*"

See K. Fogel, *Dissecting The Myth That Open Source Software Is Not Commercial*, IEEE Software Blog, `http://blog.ieeesoftware.org/2016/04/dissecting-myth-that-open-source.html`

"Free software and proprietary (or non-free) software are incompatible"

"Free software is against copyright"

"Free software licensing makes software authors lose their rights"

"You can do whatever you want with free software"

"Free software licenses make it mandatory to publish modifications"

"Free software comes with no warranty"

"Free software may be used for genocide"

"Free software is communist"

"Free software is liberal/libertarian"

"Free software is communist"

"Free software is liberal/libertarian"

> *The Free Software movement is a combination of capitalist ideas, socialist ideas, and anarchist ideas.*
>
> *— Richard Stallman, Brussels, FOSDEM 2016*

"Free software starts with Richard M. Stallman"

"Free software is qualitatively better than proprietary software"