Free/Open Source Software Licensing Primer

Stefano Zacchiroli
zack@pps.univ-paris-diderot.fr

Laboratoire PPS, Université Paris Diderot

23/11/2015

Outline

Disclaimer

- IANAL
- IANYL
- TINLA

Outline

Free Software

A program is free software if the program's users have the four essential 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

— Richard Stallman, 1986 (original version)

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

Open Source

- term coined in 1998 as a

  > *single label that identified this [open development] approach and distinguished it from the philosophically- and politically-focused label "free software."*
  >
  > — Open Source Initiative
  >
  > https://opensource.org/history

- Open Source Definition (OSD, 1998)
  - https://opensource.org/osd-annotated
  - criteria to determine whether some software (actually: its license) is open source or not
  - generalization of the Debian Free Software Guidelines (1997)

Free/Open Source Software (FOSS) gatekeepers

The term "open source" software is used by some people to mean more or less the same category as free software. It is not exactly the same class of software: they accept some licenses that we consider too restrictive, and there are free software licenses they have not accepted. However, the differences in extension of the category are small: nearly all free software is open source, and nearly all open source software is free. *https://www.gnu.org/philosophy/categories.html*

1. See: Richard Fontana, *The tragedy of the commons gatekeepers* https://lwn.net/Articles/516896/

Free/Open Source Software (FOSS) gatekeepers

> *The term "open source" software is used by some people to mean more or less the same category as free software. It is not exactly the same class of software: they accept some licenses that we consider too restrictive, and there are free software licenses they have not accepted. However, the differences in extension of the category are small: nearly all free software is open source, and nearly all open source software is free.* https://www.gnu.org/philosophy/categories.html

FOSS gatekeepers,[1] i.e., bodies that determine which software licenses are "good" or not:

- Open Source Initiative http://opensource.org/licenses
- Free Software Foundation http://www.gnu.org/licenses/license-list.en.html
- Debian https://www.debian.org/legal/licenses/

1. See: Richard Fontana, *The tragedy of the commons gatekeepers* https://lwn.net/Articles/516896/

Outline

Intellectual Property (IP)

WIPO: the World International Property Organization, UN agency

Definition (Intellectual Property according to WIPO)

Intellectual property (IP) refers to creations of the mind, such as inventions; literary and artistic works; designs; and symbols, names and images used in commerce.

IP is protected in law by, *for example*, patents, copyright and trademarks, which enable people to earn recognition or financial benefit from what they invent or create. By striking the right balance between the interests of innovators and the wider public interest, the IP system aims to foster an environment in which creativity and innovation can flourish.

http://www.wipo.int/about-ip/en/

Types of IP

There is no single “IP law”. Rather there are several bodies of law that, collectively, form what is improperly called “IP”.

IP hence covers several types of legally recognized rights, which vary across countries. The most popular branches of “IP”, found in most countries, are:

- trade secrets
- trademarks
- patents
- copyrights

Trade secrets

- The oldest branch of intellectual property
- Information that:
  - has business value
  - is not generally known to the public
  - is actively maintained secret
- A trade secret is a way to protect investments in industrial area, through industrial property laws.
- Examples: the recipe for Coca-cola; proprietary software
- Trade secret protection is obtained by declaring that the details of a subject are secret
- Trade secrets last as long as their secret status is actively protected. Disclosure, reverse-engineering, or independent invention may destroy it.

Trademarks

- Trademarks protect the association of a word, phrase, symbol, or design used with the provider of goods or services
- Trademarks are meant to protect consumers from confusion as to whom they are buying a product or a service from
  - hence: using a mark to truthfully identify a product/service is permitted by trademarks (this is known as nominative [fair] use)
- Trademarks can be obtained *de facto* by mere use, but formal registration to a trademark registration system (e.g., the USPTO in the US) gives additional rights
- trademarks can last forever, as long as they are used and are strong on the market
  - e.g., how long will the McDonald's, Coca-Cola, Apple, and Debian trademarks be around?
- trademark protection is segmented by class (e.g., food vs computers) and territory (e.g., USA vs Japan)

Open Source Definition §4

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.

Patents

- Patents are IP rights on inventions or technological developments, i.e., on devices or processes that perform a "useful" function.
- A patent grants the inventor a time-limited monopoly (e.g., 20 years) on the manufacture, use, sale, or import of the invention.
- The monopoly applies to others, even in case of independent discovery.
- In exchange of the grant of a patent, the inventor fully disclose the idea to the public.
- For exploiting the invention, interested parties must obtain (and pay for, usually) a license from the patent owner.

Copyrights

- usage restrictions on specific expressions of an idea
- e.g., copyrights can be used to limit actions like:
    - produce copies of a work and sell them
    - create derivative works
    - perform or display a work publicly
    - sell or cede these rights to others
- apply to anything that shows individual creative expression
- attach automatically to anything you create, as soon as it is "fixed in a tangible medium of expression"
    - copyright default $\approx$ monopolistic control
- limitations:
    - duration: usually in the range of 90–150 years
    - fair use: freedom of speech, freedom of citation, etc.
- somewhat uniform world-wide, since Berne Convention (1886)

Outline

Why do I need a license anyway?

- copyright applies to software
- copyright default is "all rights reserved"

User point of view

- without a license, you can't do (almost) anything with a software

Author point of view

- without a license your (potential) users can't use your software
- you need to offer at least *some rights*

FOSS licenses are judo moves on copyright

- FOSS licenses are legal hacks: they behave as other copyright licenses, but instead of restricting user rights, they grant more (and very specific) rights
- in particular: FOSS licenses grant enough rights to ensure users enjoy the 4 freedoms (run, study, copy, modify)
- that does *not* mean "free for all"; FOSS licenses can (and do) impose specific conditions
  - if you do not respect them, the license does not apply to you and you fallback to copyright default: "all rights reserved"

Note: FOSS is not against "IP". In fact, FOSS licenses *use* copyright law to realize software freedom.

Licenses are constitutions for FOSS communities

- Software licenses are social contracts just as much as they are legal documents.
- When you choose a license, you are charting a course for the future
- You are often establishing a relationship to a larger community
- Not purely about mechanical and legal choices
- It is very difficult to change later: it is worthwhile investing time to understand licensing

FOSS license categories

FOSS licenses can be classified according to the conditions they impose in exchange of software freedom.

- note: “more strict” licenses are not “less free” than others
- even the most strict FOSS license are incomparably more permissive than proprietary software licenses
  - exercise: read the EULA (End User License Agreement) of Microsoft Windows

We identify the following macro-classes of FOSS licenses:

- Lax permissive (AKA “permissive”)
- Scope-limited reciprocal (AKA “weak copyleft”)
- Reciprocal (AKA “strong copyleft”)

Academic licenses

- Relevant subset of popular permissive licenses
- The simplest licenses: very few restrictions
- Reserving only attribution (keep names and copyright notice)
- Available for all uses, including use in proprietary products
- Originally written for and popularized by universities

Examples: MIT, BSD, ISC

Permissive licenses

- Superset of academic licenses
- Include explicit grant of patent license (in modern variants)
- Available for almost all uses, including use in proprietary products

Examples: Apache License

Reciprocal licenses

- Requires that derivative work maintains the same license
- In most case reciprocal licenses require binary distribution to also include full source code
- Also known as “strong copyleft” or just “copyleft”
- Sometimes called “viral licenses”, as a denigration tactic.
  - If reciprocally licensed code is incorporated, then the application is “infected” and must be released as a whole under the same license

Examples: GPL, AGPL
Examples: CC BY-SA (for non-software works)

Scope-limited reciprocal licenses

- Like reciprocal licenses, but with limits on the scope of which parts of a derived work fall under the license terms
  - changes to the main work falls under the license terms
  - additional works that happen to be used with/added to/embedded with the main work do not
- They vary in the way the scope of the main work is limited
- According to the denigratory analogy: “virality” is limited to the main work
- Also known as: “weak copyleft”

Examples: MPL, CDDL, LGPL

What is copyleft?

Copyleft is a strategy of utilizing copyright law to pursue the policy goal of fostering and encouraging the equal and inalienable right to copy, share, modify and improve creative works of authorship.

Copyleft (as a general term) describes any method that utilizes the copyright system to achieve the aforementioned goal. Copyleft as a concept is usually implemented in the details of a specific copyright license, such as the GNU General Public License (GPL) and the Creative Commons Attribution Share Alike License.

Copyright holders of creative work can unilaterally implement these licenses for their own works to build communities that collaboratively share and improve those copylefted creative works.

— *http://copyleft.org/*

What is copyleft? (cont.)

- Granting the four freedoms is enough to guarantee users will get them only for a specific copy of the work
  - how about further downstream redistribution?
  - how about derived works?
  - how about future versions?
- Copyleft makes sure that all users receiving a copy of the program, no matter how modified, also enjoy the four freedoms.
- The copyleft clause might have diverse implementations but all of them (at least for software licenses) share the same concept: distribution of any version of this program must preserve user freedoms.
- On the other hand copyleft does preclude some business models, and for that reason it gets backlash (e.g., from corporations)

Restrictions and FOSS

Are there permissible restrictions in FOSS licenses?

Restrictions and FOSS

Are there permissible restrictions in FOSS licenses?

Yes: everything that does not get in the way of software freedom is acceptable.
In practice, deciding what is OK and what is not is not always clear cut, and the decisions may very across gatekeepers (FSF/OSI/Debian/etc).

Commonly accepted restrictions are:

- attribution of authors (as long as attribution does not impede normal use of the work)
- transmission of freedoms (e.g., copyleft)
- detailed protection of user freedoms (access to source code or prohibition of "technical measures", e.g., DRM)

License compatibility

- Two licenses are compatible if a joint derivative work (i.e., a work containing code released under each license) could be legally distributed
  - ideally as FOSS, although the notion of compatibility is general
- Compatibility is determined by comparing restrictions imposed by all involved licenses
- A dependent variable, that does not affect compatibility *per se*, is the resulting license under which the joint derivative work will be redistributed
  - e.g., GPL and BSD licenses are compatible, but the resulting joint work will be under the terms of GPL only
  - e.g., GPL and MPL version 1.1 are incompatible (i.e., it is impossible to integrate code released under the two licenses without violating the terms of at least one of them)

Dual- (or multi-) licensing

Distribute software under two (or more) different sets of licenses.

The expression is used to express two different notions:

- license segregation: different licenses apply to different copies of the same program (e.g., for proprietary relicensing business models)
- user choice: different, alternative (OR-ed) licenses apply to the same copy of the software; the user choose the license
    - degenerate case: “version N or above” clauses. The user can choose *which version* of the license apply to them

Motivations:

- License compatibility (e.g., Perl, Firefox)
- Business models based on market segregation (e.g., MySQL, OCaml)
- Future-proof license-based strategies

Outline

Popular and noteworthy licenses

Grandma's licensing tips

- **Don't write your own license,** you'll do it wrong
- Choose your strategy, than pick a popular licenses

- Lax permissive (AKA "permissive")
  - BSD 3-Clause "New" or "Revised" license
  - BSD 2-Clause "Simplified" or "FreeBSD" license
  - Apache License 2.0
  - MIT license
  - ISC License
- Scope-limited reciprocal (AKA "weak copyleft")
  - GNU Lesser General Public License (LGPL), versions 2.1 and 3
  - Mozilla Public License (MPL), version 2.0
  - Common Development and Distribution License (CDDL)
- Reciprocal (AKA "strong copyleft")
  - GNU General Public License (GPL), versions 2 and 3
  - GNU Affero General Public License (AGPL), version 3

Outline

3-clause BSD (1999)

```
Copyright (c) <year>, <copyright holder>
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:
    * Redistributions of source code must retain the above copyright
      notice, this list of conditions and the following disclaimer.
    * 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.
    * Neither the name of the <organization> nor the
      names of its contributors may be used to endorse or promote products
      derived from this software without specific prior written permission.
[ ... ]
```

- popular permissive license
- you may redistribute the work, in any form (source or binary), with or without modifications, as long as you preserve copyright notices
- non endorsement requirement
- approved by: FSF, OSI, Debian
- GPL compatible

The MIT License (1988)

```
The MIT License (MIT)
Copyright (c) <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.
[ ... ]
```

- functionally similar to modern BSD licenses
- used by X11/X.org, Symfony, RoR, Lua, Putty, Mono, . . .
- no explicit non endorsement clause
- it states more explicitly the rights given to the end-user
- approved by: FSF, OSI, Debian
- GPL compatible

Apache License 2.0 (2004)

- popular license
  - ≈150 projects hosted by the Apache Software Foundation (2015)
  - over 8’000 non-ASF projects located at SourceForge are available under Apache License (2012)
  - 25% of Google Code projects, including Android user space (2008)
- approved by: FSF, OSI, Debian
- compatible with GPLv3
- incompatible with GPLv2

Apache 2.0: patent license

§3. Grant of Patent License

Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable, patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, [. . .] to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s)

[. . .]

If You institute patent litigation against any entity alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

Outline

Mozilla Public License (MPL) 2.0 (2012)

- descendant of the Netscape Public License
- scope-limited reciprocal license, with file-based boundaries on the reach of copyleft requirements
- “covered software” vs “larger work”

Covered Software

> *means Source Code Form to which the initial Contributor has attached the notice in Exhibit A, the Executable Form of such Source Code Form, and Modifications of such Source Code Form, in each case including portions thereof*

Larger Work

> *means a work that combines Covered Software with other material, in a separate file or files, that is not Covered Software*

Mozilla Public License (MPL) 2.0 (cont.)

Covered software

1. copyleft-like clause

   All distribution of Covered Software in Source Code Form, including any Modifications that You create or to which You contribute, must be under the terms of this License

2. source code requirement

   If You distribute Covered Software in Executable Form then [. . .] such Covered Software must also be made available in Source Code Form

Larger work: its own license

If the Larger Work is a combination of Covered Software with a work governed by one or more Secondary Licenses, [. . .], this License permits You to additionally distribute such Covered Software under the terms of such Secondary License(s), so that the recipient of the Larger Work may, at their option, further distribute the Covered Software under the terms of either this License or such Secondary License(s).

Mozilla Public License (MPL) 2.0 (cont.)

- Explicitly grants patent rights where necessary to operate the software.
- approved by: FSF, OSI, Debian
- version 2.0 of the license is compatible with the GPL
- version 1.1 is *in*compatible with the GPL
    - A module covered by the GPL and a module covered by the MPL version 1.1 cannot be linked together.
    - For this reason, Firefox has been relicensed under multiple licenses (MPL, GPL, LGPL).
    - MPL 1.1 can be specifically amended to allow combining with GPL and others (sect. 13, “Multiple-licensed code”).

GNU LGPL

1991 GNU *Library* General Public License, version 2 (for uniformity with GPL version)

1999 GNU *Lesser* General Public License, version 2.1

- name change to emphasize that it is inferior (from a copyleft POV) to the GPL, rather then the recommended variant of the GPL for software libraries

2007 GNU LGPL, version 3

- reimplemented as GPLv3 + additional permissions

- very popular license for libraries (and more)
- approved by: FSF, OSI, Debian
- GPL compatible

GNU LGPL 2.1

- *§4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, [. . .]*

GNU LGPL 2.1

- *§4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, [. . .]*
- *§5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.*

Note the lack of explicit file boundaries (contrary to, e.g., MPL)

GNU LGPL 2.1

- *§4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, [. . .]*
- *§5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.*

Note the lack of explicit file boundaries (contrary to, e.g., MPL)

- *§3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library [. . .] This option is useful when you wish to copy part of the code of the Library into a program that is not a library.*

Outline

GNU General Public License (GPL)

- approved by: FSF, Debian, OSI

1989	version 1 (by RMS), as a generalization (hence the name) of licenses already used by the GNU project for: Emacs, GDB, GCC
1991	version 2 (by RMS) • "liberty or death"; early ex. of defense against patents and similar threats to user freedoms
2007	version 3 (by RMS with counsel from E. Moglen/SFLC) • public review process • software patents clauses • DRM clauses (anti "tivoization") • license compatibility provision • internationalization • self-defense against further restrictions

GPL — relevance

What makes the GPL so special?

- Considered to be the most popular FOSS license
- It was the first license to define and implement copyleft
    - Without the GPL, copyleft would have been just an abstract idea
- Highly influential on all subsequent copyleft-Like licenses, including Creative Commons share-alike
- Designed to prevent proprietary relicensing

GPLv2 — source code requirement

§3. *You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:*

a) *Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or, [. . .]*

2. the so called "system library exception"

GPLv2 — source code requirement

§3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) *Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or, [. . .]*

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable.
However, as a special exception,[2] *the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.*

2. the so called "system library exception"

GPLv2 — copyleft

> *§2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:*
>
> a) *You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.*
>
> b) *You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.*

- derived works fall under the terms of the GPL themselves, hence *their* source code must be distributed as well
- (a) is a local requirement, whereas (b) only triggers upon distribution

GPLv2 — “or later”

Recommended way to apply the GPL to source code:

> *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 2 of the License, or (at your option) any later version.*

- part of the copyright/license notices, not of the license itself
- individual software authors *can* leave the “or later” clause out
- other licenses include implicit “or later” requirements in the license text itself (e.g., MPL)

For best practices on how to manage copyright/license notices see: Software Freedom Law Center, *Managing copyright information within a free software project* https://softwarefreedom.org/resources/2012/ManagingCopyrightInformation.html

The Application Service Provider (ASP) loophole

1. obtain a copy of some GPL'd program
2. modify it
3. offer remote access to your modified version over the Net (e.g., web app, remote API, etc.)

Does the GPL force you to redistribute the code of your modified version?

The Application Service Provider (ASP) loophole

1. obtain a copy of some GPL'd program
2. modify it
3. offer remote access to your modified version over the Net (e.g., web app, remote API, etc.)

Does the GPL force you to redistribute the code of your modified version? **No.**

- GPL (both v2 and v3) copyleft clauses trigger upon "distribution"/"convey" of the modified copy, in either source or non-source form
- if you do not do any of that, copyleft does not kick in
- from copyleft POV, this is very problematic for web/network apps

  "GPL is the BSD of Web applications" — *Bradley Kuhn*

- but in an increasingly more connected world, the problem is more general

GNU Affero GPL (AGPL)

- Based on the GPL
- Published by the Free Software Foundation (version 3: 2007).
- It contains the extra Affero clause that requires distribution of modified source code of applications to users interacting remotely over the network with the program
- The clause has initially been considered for inclusion in GPLv3, but then relegated into a separate license
- approved by: FSF, Debian, OSI
- GPL compatible (explicitly so)

AGPL — Affero clause

***§13. Remote Network Interaction**; Use with the GNU General Public License.*

[. . .] if you modify the Program, your modified version must prominently offer all users interacting with it remotely through a computer network (if your version supports such interaction) an opportunity to receive the Corresponding Source of your version by providing access to the Corresponding Source from a network server at no charge, through some standard or customary means of facilitating copying of software.

Conclusion

- the amount vary, but FOSS is ubiquitous in the software world
- software developers & strategists can not be ignorant on software licensing issues
- when you adopt software:
  1. find out about its license (and authors)
  2. review it for (legal- and strategical-) compatibility
- when you release software as FOSS:
  1. choose your strategy
  2. pick a matching popular FOSS license

  remember: the license is the Constitution of your community
- go read some licenses in full, they are full of nice hacks!

Outline

Outline

Derivative works and the GPL

GPL copyleft propagation applies to (GPLv2 language):

> *a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language.*

Quiz

Can you link a GPL program/library with a non-GPL program/library, without applying the GPL to the obtained binary?

Derivative works and the GPL

GPL copyleft propagation applies to (GPLv2 language):

> *a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language.*

Quiz

Can you link a GPL program/library with a non-GPL program/library, without applying the GPL to the obtained binary?

- FSF/FSF's lawyers (and popular) answer: no; the GPL applies
- some corporate lawyers' answer: yes; the GPL doesn't apply
- court cases/tribunal answer: none (yet)

(arguably, the resulting answer is thus "we don't know")

Derivative or collective works?

[US law language]

- a derivative work is a "work based upon one or more preexisting works", which requires some transformation or adaption of the original
- a collective work is created when a person brings together "preexisting materials. . . in such a way that the resulting work as a whole constitutes an original work of authorship"
  - individual parts remain under their individual licenses
  - a separate license apply to the collection

Does linking create a derivative or a collective work (or both)?

Linking and the GPL — FSF position

License text (redux):

> *a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language.*

From the GPL FAQ[3]

> *Q: Does the GPL have different requirements for statically vs dynamically linked modules with a covered work?*
> *A: No. Linking a GPL covered work statically or dynamically with other modules is making a combined work based on the GPL covered work. Thus, the terms and conditions of the GNU General Public License cover the whole combination. [. . .]*
> *Q: Can I release a non-free program that's designed to load a GPL-covered plug-in?*
> *A: [. . .] Using shared memory to communicate with complex data structures is pretty much equivalent to dynamic linking*

3. `http://www.gnu.org/licenses/gpl-faq.html`

Linking and the GPL — arguments

- dynamically linked executables contains "annotations and elaborations" on a base binary
  - does a Linux kernel module contains annotations and elaborations on the base expression of the kernel?
  - if yes, then it might be a derived work of the kernel (GPLv2)

4. `https://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/tree/COPYING`

Linking and the GPL — arguments

- dynamically linked executables contains “annotations and elaborations” on a base binary
  - does a Linux kernel module contains annotations and elaborations on the base expression of the kernel?
  - if yes, then it might be a derived work of the kernel (GPLv2)
  - how about user programs that run on Linux?
  - according to Linus:[4]

    > *NOTE! This copyright does *not* cover user programs that use kernel services by normal system calls - this is merely considered normal use of the kernel, and does *not* fall under the heading of "derived work". Also note that the GPL below is copyrighted by the Free Software Foundation, but the instance of code that it refers to (the Linux kernel) is copyrighted by me and others who actually wrote it.*

  . . . but is he right?
- the legal principle of usage of trade might play a role too

4. `https://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/tree/COPYING`

Linking and the GPL — arguments (cont.)

- arguments to the contrary (often by corporate lawyers) claim that linking only creates collective works—not subject to the GPL as a whole—because there is no substantial difference between two executables on disk and two in memory
- header files might also play a role
  - during compilation (before linking) you might use header files to prepare your executable for dynamic linking
  - if the headers used at compile time are GPL'd, then your dynamically linked executable might be a derived work of the headers

Linking and the GPL — arguments (cont.)

- there are also other types of “linking”: RPC, RMI, REST API, etc. When do they constitute “linking” in a sense that would trigger strong copyleft requirements?
  - no consensus yet
  - folklore suggests that:
    - loosely coupled and/or popular and/or standardized APIs with several alternative implementations should not trigger the GPL
    - tightly coupled and/or ad-hoc and/or single-implementation APIs should trigger the GPL
- on the other hand, it seems consensual that static linking will produce a derivative work of the GPL part

Linking and the GPL — arguments (cont.)

- there are also other types of “linking”: RPC, RMI, REST API, etc. When do they constitute “linking” in a sense that would trigger strong copyleft requirements?
  - no consensus yet
  - folklore suggests that:
    - loosely coupled and/or popular and/or standardized APIs with several alternative implementations should not trigger the GPL
    - tightly coupled and/or ad-hoc and/or single-implementation APIs should trigger the GPL
- on the other hand, it seems consensual that static linking will produce a derivative work of the GPL part

Ultimately, this GPL linking dilemma seems to be problematic only for those who want to somehow circumvent the main principle of the GPL which, *per se*, is very clear.

Outline

GPLv2 — looking back

- Written by Richard Stallman and the FSF, published in 1991.
- The most popular Free Software license: estimated to cover 50-70 % of all Free Software projects (at the time)
- It’s more than a software license: it is a social contract, imposing that all players have the same rights and obligations

Why update it?

GPLv2 — looking back

- Written by Richard Stallman and the FSF, published in 1991.
- The most popular Free Software license: estimated to cover 50-70 % of all Free Software projects (at the time)
- It's more than a software license: it is a social contract, imposing that all players have the same rights and obligations

Why update it?
After 15 years, needed updating in order to remain effective against new threats to user freedoms.

Intuition: the GPL is a mean to an end. It is an implementation that might have bugs (or grow them over time), which need to be fixed in further releases of the license.

GPLv3 — public consultation

Public consultation process:

- very relevant and the social responsible thing to do: given the abundance of "or later" software, the effects of the release of GPLv3 might be huge
- It lasted eighteen months: from January 16, 2006 (first draft) to June 29, 2007 (final version).
- Selected/invited participants from high-profile Free Software projects.
- 4 drafts.
- 5 International Conferences (Boston, Porto Alegre, Barcelona, Tokyo and Brussels)

GPLv3 — DRM

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

[. . .]

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

- does not *forbid* to implement DRM & co. in software
- but allows to write interoperable software and bypass restrictions
- neutralize laws that get in the way of user freedoms (e.g., DMCA, EUCD)

GPLv3 — DRM (cont.)

- Together with

  > *6. Conveying Non-Source Forms.*
  >
  > *[. . .]*
  >
  > *"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source.*

  it also neutralizes "tivoization", i.e., the circumvention of the GPL by using cryptography to disallow the installation/execution of modified versions of a GPL'd program

GPLv2 — patents

Protection against patent threats is implemented by GPLv2 only in the "Liberty or Death" clause:

> *7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all.*
>
> *[. . .]*
>
> *It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. [. . .]*

GPLv3 — patents

GPLv3 *adds* (i.e., “liberty or death” remains) stronger protection against patent threats through legal-engineering:

> *11. Patents*
>
> *[. . .] Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor’s essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.*

> *10. Automatic Licensing of Downstream Recipients.*
>
> *[. . .] you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.*

GPLv3 — self-defense

7. Additional Terms.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. [require copyright ownership]

[. . .]

All other non-permissive additional terms are considered "further restrictions" [. . .] If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term.

GPLv3 — variants and compatibility

- *7. Additional Terms.*
  [. . .] for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms: [. . .]
  e) *Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or*

  (and other similar permissions for adding warranties/“as is” disclaimers, limiting the use for publicity purposes, etc.)

- *Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work.*

Outline

CAA/CLA

Copyright Assignment Agreement (CAA)
: cession agreement where a copyright holder surrender all their copyright sanctioned rights on some work to another party

Contribution License Agreement (CLA)
: agreement where a copyright holder gives a license (usually non-revocable, possibly exclusive) to enforce specific copyright sanctioned rights to another party

- on paper, CAA are more powerful than CLA; but they only go as far as the legal system allows them
  - e.g., in most of Europe moral rights cannot be surrendered
- CLAs can be so broad to be *de facto* equivalent to CAAs
- key copyright right for policy reasons: the ability to relicense

If a vendor participating in a FOSS project has, alone, the ability to relicense, strategy considerations based solely on the chosen FOSS license are completely moot.

Not all CAAs/CLAs are born equal

- set of rights surrendered
  - e.g., enforcement-only agreements
- mandatory vs optional agreements
- to (public benefit) non profit vs for profit entities
- safe guards
  - e.g., we can relicense, but we will pick within this set of licenses
    - common choice: OSI-approved $\cap$ FSF-approved licenses
- alternatives (within limits)
  - “or later” clauses
    - possibly with license proxy (e.g., GPLv3, §14)
  - will

Outline

Creative Commons (CC) licenses

- good licenses for non-functional works (e.g., music, artwork, documentation, etc.)
  bad licenses for software
- CC0 is a notable exception, being the closest you can get to a world-wide public domain dedication
- several very different licenses under a common label
- *some* CC licenses are considered free by FOSS communities
  - free: CC0 (public domain), CC BY, CC BY-SA (copyleft)
  - non free: CC BY-ND, CC BY-NC, CC BY-NC-SA, CC BY-NC-ND

  i.e., any license with at least one of “Non Commercial” and “No Derivatives” is not free
- prefer recent versions (4.0+) if possible
- starting from version 4.0, CC BY-SA has an updatable list of explicitly, one-way compatible licenses
  - starting October 2015, GPLv3 has been added to the list: you can reuse CC BY-SA 4.0 licensed works under the terms of GPLv3