Basic information on licenses and collaborative software development

Edlira Nano

December 2, 2016

1 What is copyright and why use a license?

The law implicitly protects your work with a copyright (“droit d’auteur" in french), which is valid for 70 years after death in Europe.

This protection benefits to the author automatically and does not allow any usage from other parties without the author’s consent. This is why, when you want to share software, it has to be an explicit and willing act from the author.

This is expressed in the form of a license added to the work, where the author indicates the rights given to users and eventually the obligations.

By putting a license on your work you protect this work, even if this is the most open and permissive license there is. But you also permit its distribution : without a license, the user has to get an explicit authorization from the author in order to use the work.

See this article in french for a detailed explanation of copyright.

2 License

2.1 Proprietary license

Limits in proprietary or closed licenses are most of the time :

- interdiction to copy (no distribution)
- usage limitations (number of users, machines, time etc)
- interdiction to find out how it works (it is called reverse engineering or to decompile, it means more or less to hack, to reproduce the code). As a consequence, one cannot verify the precise working algorithms of the software, not only because it is difficult (the source code is hidden) but because it is prohibited. This can be a big problem in research, where one has to precisely know, trace and be able to reproduce software computational steps. Therefore, it is almost impossible to use proprietary software for computations in a research work, unless the proprietary of the software gives you the exact information that such scientific work demands.

2.2 Free and open-source licenses

Free licensed software is not the same as :

- software in the public domain : software on which its author has renounced to all copyright
- freeware : free of cost software that is not necessarily open-source
- shareware : software that asks for a payment after a free of cost trial period

2.2.1 Free software definition

Richard Stallman and the FSF (Free Software Foundation), define free software (free as in freedom not as in free beer) according to four freedoms:

1. freedom to run the program without restrictions,
2. freedom to study how it works and adapt it to your needs (free access to source-code is hence a required condition here),
3. freedom to redistribute copies,
4. freedom to modify the code and redistribute modifications (free access to source-code is again a required condition here).

Free software is not antinomic with commercial software.

2.2.2 Open-source license definition

Derived from the Debian guideline, the OSI (Open Source Initiative) has defined open-source software as following these 10 criteria.

2.2.3 FOSS licenses comparison

FOSS stands for : Free and Open Source Software.

The FSF's free software definition focuses on the user's unrestricted rights to use a program, to study and modify it, to copy it, and redistribute it for any purpose, which are considered by the FSF the four essential freedoms. The OSI's open-source criteria focuses on the availability of the source code and the advantages of an unrestricted and community driven development model. Yet, many FOSS licenses, like the Apache license, and all Free Software licenses allow commercial use of FOSS components.

Most of the open-source licenses are free licenses.

For a detailed comparison of some well known open-source and free licenses see here.

2.2.4 Derived work and copyleft licenses

The notion of copyleft is what makes the big difference between two FOSS licenses.

If one has a free licensed software A:

- one can always make a modified version B of A if the license of A allows him
- but the license of product B depends on the license of A.

In fact there are two sorts of free licenses :

- the permissive ones: no constraints on the license of B (Apache, BSD ...). These licenses are very close to public domain.
- the copyleft ones : the license of B has to be the same as A or equivalent.

GNU's GPL (Gnu Public License) is the most known and widespread copyleft license. Its aim it to ensure that free software stays free.

See here for a list of copyleft and non-copyleft FOSS licenses.

There is a further categorization of copyleft licenses in strong and weak copyleft `https://en.wikipedia.org/wiki/Copyleft`.

2.2.5 FOSS licenses list

See here for a complete list of FOSS licenses.

Lionel told me that there is an EUPL (European Union Public License) that he has been using, This license is very similar to the GPL (both are copyleft) but takes better in account the European laws.

The CNRS, CEA and INRIA are using a french version of the GPL that they have defined, called CeCILL. They recommend the use of this license designed to better fit the European laws, in particular for research software.

2.3 Multi-licensing

One has the right to distribute its software under multiple different licenses. Examples :

- Perl is distributed under two free software licenses : GPL and Artistic;
- there is not necessarily license compatibility : Qt used to have a double licensing : one free license (GPL) for free software and one non-free license for non-free software;

- one commercial license for users willing to buy/sell your software, one open-source license for the user/developer community : in my previous research lab (an INRIA lab) we developed a software licensed under GPL. A private company wanted to buy it, we made a special proprietary contract with them for a precise given version of the software that they bought. Our GPL licensed version of the software is still available and is the official maintained version.

2.4 Non-software license (documentation, slides, videos, art...)

- Creative Commons (CC) are a set of licenses which aim is to give you various liberty degrees while distributing your work. The idea is that you have a list of liberties : for example NC (Non Commercial), ND (No Derivative), SA (Share-ALike = copyleft), etc. Then you go shopping and choose the options that you want for your work (and that are allowed together) among a total of 8 possibilities (for example CC−BY−SA or CC−ND.
- GNU Free Documentation License, GFDL
- Licence de Libre Diffusion des Documents and its english version Free Document Dissemination License
- For art work : `http://artlibre.org/`, `http://gnuart.org/`

2.5 For databases

In Europe, an EC directive (96/9/CE of March 11, 1996) gives the proprietary of the databases some special possibilities and protection.

2.6 Licensing of collaborative works

The author of a software is the only one that can decide the license of its software. He can also choose different licenses for different clients. But the situation gets complicated if you accept external contributors to your software:

- if the contributor accepts to give all of his patrimonial rights to the original author, then the author can continue doing whatever he wants to do and change licenses as he wants to (MySQL for example).

- if not, the original author can be forced to use the license under which the contributions are placed.

3 Source code repositories (forges)

A source code repository or forge is a file archive and web hosting of source code that often offers a source code version control (git, svn, ...), bug tracking, release management, mailing-lists, wiki-based documentation etc. They are very convenient and widely used to host FOSS projects.

The most popular forge is GitHub (38 million projects), but also GitLab (550 thousand projects), Assembla (526 thousands projects) and SourceForge (430 thousand projects) (see here for an extended comparison list).

3.1 SourceSup

In France, only for public research software, SourceSup is a very complete forge that offers : code versioning (git or svn you choose), release management, wiki-based documentation, bug tracking system, mailing-lists.

The projects hosted on SourceSup can only come from research labs in France, but the users and the community can be everyone (source-code can be downloaded/cloned/checked-out by everyone, the project administrator chooses these settings in his project administration panel).

SourceSup is based on the open-source FusionForge, which explains the not so modern and not so user friendly interface, but the decision has been made to migrate SourceSup to GitLab, an open-source and modern forge.

3.2 GitLab

It is a very serious open-source alternative to GitHub.

3.3 GitHub

GitHub is by far the most popular web hosted forge with its 38 million projects. GitHub is proprietary.

3.4 Proprietary and open-source forges : the GitHub case

Not every source-forge provider is based on free or open-source code its self. For example the most popular one GitHub, is not.

So, if you are going to host an open-source project on a forge, you have to be careful to where and why you host it. You could be loosing some privileges that open-source software provides. GitHub being the most popular proprietary forge, we are taking it as an example to discuss these problems as follows.

3.4.1 Problems

Centralisation: The more obvious danger of proprietary forges is the centralization of data. For example in the case of GitHub, GitHub Inc, the society that owns GitHub, is the only owner of the GitHub platform. So if GitHub encounters technical problems or worse closes, you have nothing left (this has happened before). Your project's future depends on the only will of GitHub Inc's society. In an open-source forge this is not the case, you and the community have full access to the code and memory of the forge, if the hosting facility closes, everything is there ready to be forked or hosted elsewhere.

Bug Tracking System (BTS) problem : the Bug Tracking System (BTS) that GitHub offers for projects is a proprietary tool.

The bug reports are crucial to any free open-source software, they are the entry point of new contributors, they are useful to its users community, they are its memory, they are its past and future history.

Being proprietary, the BTS of GitHub offers no migration possibility to other open-source BTS standards, so if GitHub fails or closes, you loose this precious memory. (This happened before, see Astrid and Yahoo). And if you simply want to migrate your BTS from GitHub to another BTS system, you cannot.

For a full description of the problems in using only GitHub : see here.

3.4.2 Advantages

But, on the other hand, GitHub nowadays offers the visibility, and the potential community that you will maybe never have elsewhere. So what to do?

Many people, use GitHub as a secondary source-forge, not the primary one, just in case the primary one fails. And they do not use GitHub's Bug Tracking System. For example : Linus Torvalds (the creator of git among other stuff ;-) has put the Linux kernel code on GitHub in 2011 as a secondary access point, when the servers at kernel.org were encountering some technical problems. He says that he is using GitHub as a secondary free hosting possibility. But the Linux kernel's whole project (including BTS) is not hosted on GitHub (it is hosted in the Linux Foundation facilities).

4 Going further

For this document I mainly used these resources :

- the lectures of *Logiciels libres* from Roberto Di Cosmo, my Master teacher, director of IRILL;
- an AFUL document on free licenses;
- a CNRS document in french which gives a very good explanation of copyright and *droit d'auteur*;
- Libérez vos oeuvres, a document in french from l'April.

Depending on your requests and needs, this document can be augmented and detailed further on.

5 Some common questions/problems

My code is under no license, and i do not care. Why don't you take it and do whatever you want with it? I cannot, if you read section 1 above about copyright, you see that the law protects you even if you don't want to. Not licensed code means that nobody has the right to use it or modify it. I can privately use your code but then I have no right to give it to anybody else. So I will not take your code.

I have so much work that I have no time to deal with this kind of less important things. Once you decide which license to use for your code, you just integrate a text line saying "this code is licensed under the X license terms. See website for details". Sometimes you have to add the text of the license in a separated file each time you distribute your code. But that's it.

There is generally no need to invent a new license, people have thought and thought about that and have invented everything, if you think you have to invent a new license, most of the time it is because you have not opened your eyes enough around you. But luckily (or not:), the TAGC hired me so I can help you.

I am afraid that if I put my code under free or open-source license, somebody is going to take my code and never say nothing about it, he will become rich and famous and I will never be granted. This will not happen. The reality is that most of very good real free licensed software are lacking contributors and dying somewhere in the GitHub dimension, hoping that one day they will have the recognition they certainly deserve.

But let's pretend it happens in an imaginary world. Law is with you, the fact of putting your software under a free solid copyleft license like GPL for example protects your software even more. The Free Software Foundation will rescue you, the FOSS community will be on your side, you will become a hero. Instead, if you put your code in the new closed license that you just invented on your own and that makes every single user of the software ask for permission, you can be sure that nobody is never going to help you with its code (unless you force them) and that a large part of the users will rather not use your software then ask for permission.

Just think about big successful free software as Linux kernel, Debian, Ubuntu, Firefox etc. They all benefited from free licenses. And they all have a person or a foundation that coordinates the project, that makes the big decisions, they never loose their leadership on the source code, but their community is huge. So being afraid and closing-up will not help you, on the contrary open-up, be nice, be welcoming, protect your code with a good well-known FOSS license, and if the community comes, give them all the possibilities, don't impose yourself, they will likely need and ask for your leadership and expertise. Even Microsoft is doing FOSS now, because they have too, their proprietary economical model is failing.

I see. But I still do not care at all. Ok no problem, have a nice day, see you around :-)

If my program or the program I am using is under a free or open-source license, do I have to publish every "in house" code modifications that I am doing over it (disclosure obligation)? No, the GPL and most of the free / open-source licenses do not have a disclosure obligation : you do not have to share your "in-house" code that you can thus modify and use it privately as you wish. The only licenses that I know of that force you to do so are the QPL (trolltech) and the Affero GPL.

SourceSup does not allow people that do not have an account to clone my repository This is not true, it is not easy to find out but there is a way of telling SourceSup to let everybody clone your software and to parametrise every project settings you want to. I can help you with that.

Is the FSF, OSI, SourceSup or any other entity you cite paying you? I wish they were. No I just work here with you :-)