Open Source Licences Wars

Shlomi Fish, Shlomi Fish <shlomif@shlomifish.org>

Revision History

Revision 2659	21 August 2009	shlomif
Fixed some small typos and misphrasings.		
Revision 2651	21 August 2009	shlomif
Added descriptions of strong copyleft licences, some links to opinions about which licence to choose. Spell checked and prepared for release.		
Revision 2453	30 June 2009	shlomif
Added descriptions of weak copyleft licences.		
Revision 2452	30 June 2009	shlomif
Fixed some typos and misphrasings.		
Revision 2434	18 June 2009	shlomif
Added the Freecell though experiment.		
Revision 2433	18 June 2009	shlomif
Added the Exceed thought-experiment.		
Revision 2431	18 June 2009	shlomif
Added the part about the GPL and the hacker attitude.		
Revision 2428	15 June 2009	shlomif
Finished up to the first E-mail in the Hackers-IL thread. Now to cover the rest of the E-mails.		
Revision 1000	19 July 2008	shlomif
Forked the template from a previous work and working on it.		

Abstract

Table of Contents

Introduction

When Joel Spolsky ("Joel on Software") wrote his notorious blog post "Language Wars", many people asked whether he has "jumped the shark" and that his blog will go downhill from there. I too have read the post, and agreed, that while it had a few good points, it was too based on "feeling rather than knowing". Joel later on posted many good articles and shorter entries on his blog, but many people still recalled it as a very low-point in the blog.

Like Joel, I have a home-site and several blogs, where I post articles and essays about my thoughts, and this time I've decided to risk something similar to what Joel has done on an equally flamatory topic: licences of open-source software. I'm going to introduce the various options, explain a little about their motivation and then give some advice according to my own personal opinion.

I should note that I am not a lawyer, and only write this according to my best knowledge, understanding and experience. [1] I think what I'm writing here is correct, but would still appreciate any correction for future versions. If you're unsure about my advice, then you should consult a legal expert (if you can afford one) or your local or global Free and Open Source Software organisation. I cannot guarantee my advice is valid.

Spoiler

In this spoiler I'll explain what I'm using right away, which is also what I recommend other people to use. When I started releasing my code as open-source software, I used the Public Domain. So, for example, Freecell Solver which is one of my earliest projects and still one of the most successful, used to carry a `COPYING` file with the following contents:

> Relax, this is not GPL software, but rather it is distributed under the public domain. It means it can be linked against anything, converted to any different license, freely used and distributed, and anything else without any restrictions whatsoever. No Strings Attached!™

(This was also a joke of sorts.)

Later on, to avoid the problematic status of public domain code in many countries, and to avoid other potential legal problems, I started distributing my original code under the MIT X11 Licence, a BSD-style licence which is the closest one to the Public Domain. If the project I'm contributing to has a different licence, then I disclaim any explicit or implicit ownership on the code I wrote, and allow the existing copyright holders to do with my code as they see fit.

Note that I'm using different terms for other types of digital artworks. Most of my essays (including this one) are under the Creative Commons Attribution Licence, my original (i.e: non-derived-work or fan-art) fictitious stories are under the Creative Commons Attribution-Share Alike Licence (which is more GPL-like), and most of the photos I took and published were placed under the Public Domain.

[1] Some people believe that discussing legal matters should be reserved to people who are Law experts. However, this is a sub-case of the "only specialists should deal with a specialised field" fallacy.

Like it or not, the Law affects us all and the Government expects common citizens to be aware of it and not violate it, even if they are not lawyers. As software developers, we do not limit discussions on software and computing to people with a Bachelor's degree in Computer Science (much less a Ph.D. or whatever) and often not-too-qualified people can be very knowledgeable and/or productive.

Similarly, I expect discussions on the governmental law to have a lower barrier for entry than people who are qualified lawyers, or Law Professors.

Types of Licences

What is a Free Software Licence?

According to the Free Software Definition free software must fulfil 4 freedoms:

1. The freedom to run the program, for any purpose

2. The freedom to study how the program works, and adapt it to your needs. Access to the source code is a precondition for this.

3. The freedom to redistribute copies so you can help your neighbour

4. The freedom to improve the program, and release your improvements to the public, so that the whole community benefits . Access to the source code is a precondition for this.

The Open Source definition is similar, but some licences can qualify as open-source and not as free software. This is usually not an issue, because the majority of open source software out there is free as well. (And few people would like to create an open-source licence that's not endorsed by the Free Software foundation.)

A software licence qualifies as a free software/open-source licence, if it allows those four freedoms. There's still a lot of room to manoeuvre and to impose various restrictions, but otherwise these freedoms make them usable enough for the users and co-developers.

Public Domain Licences

The first type of licences are public-domain-like licences, also known as BSD-style licences, Permissive free software licences, or "Copycenter licences". These licences allow you to do almost anything conceivable with the program and its source code, including distributing then, selling them, using the resultant software for any purpose, incorporating into other software, or even converting copies to different licences, including that of non-free (so-called "proprietary") software.

Probably the only two actions none of these licences allow is removing the original copyright notice, or suing the creator of the software for damages (the latter is due to the "no warranty" clause). Some of them pose some additional that do not reduce from their public domain nature.

Prominent examples of Public Domain licences include:

1. The various BSD licences, under which the "Berkeley Software Development" UNIX variant were made available. The original 4-clause BSD licence contained an additional advertising clause, that required publishing the names of the copyright holders on every advertising material. This proved to be a problem and was rendered GPL-incompatible and so using this licence is no longer recommended.

   Later on, the advertising clause was removed to formulate 3-clause and 2-clause BSD licences, which are both less problematic and GPL-compatible.

2. The MIT X11 Licence was created by MIT for their X Window System software. It explicitly allows to do many activities with the licensed code, including sub-licensing, which means converting derived works of the code to different licences.

3. the Apache Licence was formulated for the Apache project and has seen several revisions. It includes some language to deal with patent-related issues. The latest version - Apache 2.0 was partially created in

order to be compatible with version 2 of the GPL, but the FSF ruled the Apache Licence as incompatible with it. The Apache Licence, however, was declared as compatible with version 3 of the GPL.

4. The ISC licence is functionally equivalent to the 2-clause BSD-licence, with some “language made unnecessary by the Berne convention removed”. This licence is also favoured by the OpenBSD project.

Weak Copyleft Licences

Weak Copyleft licences are free software licences that mandate that source code that descended from software licensed under them, will remain under the same, weak copyleft, licence. However, one can link to weak copyleft code from code under a different licence (including non-open-source code), or otherwise incorporate it in a larger software.

Otherwise, weak copyleft licences allow free distribution, use , selling copies of the code or the binaries (as long as the binaries are accompanied by the (unobfuscated) source code), etc.

Examples of weak copyleft licences include:

1. The GNU Lesser General Public Licence (or LGPL for short) - this licence was formulated by the Free Software Foundation based on the GNU General Public Licence (or GPL). Its text is longer than the GPL, and it is reportedly a more complicated licence. Other than the general “you-can-link-it-against-everything-as-long-as-you-keep-inclusive-code-LGPLed”, it allows an explicit conversion to the GPL, has patent-related clauses, and contains many additional restrictions.

   Version 2 of the LGPL was succeeded by version 2.1, and now there’s also version 3 which is based on a short adaptation of the version 3 of the GPL (see below).

2. Version 2.0 of the Artistic Licence, which was formulated by the Perl foundation is used by the Parrot Virtual Machine, some Perl 6 implementations, and other projects. The original Artistic Licence has problematic phrasing, and is not recommended for general use, unless possibly when dually licensed with a different licence (as is the case for perl 5 and many Perl modules on the CPAN).

   The Artistic Licence version 2 is relatively short and easy to digest. One note about it is that it allows distributing binaries that were derived from a modified source, without the obligation to release the modifications of the source. However, if one wishes to distribute a modified source, he must make his modifications under the same licence.

   An anecdote about this licence is that it was chosen by R.E.M for licensing some video-clips from one of their albums.

3. The Mozilla Public Licence (or MPL) was phrased for the public source release of the Mozilla project. It was characterised as a hybridisation of the modified BSD licence and the GPL. This licence was ruled as GPL-incompatible, which eventually caused the Mozilla project to re-license its code under a triple MPL, GPL and LGPL licence. This is still the licensing terms of Mozilla projects such as the Firefox web-browser up to today.

4. **GPL with an Exemption Clause** - the GPL is a strong copy-left licence, but sometimes an additional clause is added that allows it to be linked against code of non-GPL-compatible licences (including non-open-source code). This strategy was chosen by Sun Microsystems when they decided to release the source of the Java programming language under an open-source licence.

Strong Copyleft Licences

Strong copyleft licences go a step further from weak copyleft licences and mandate that any distributed software that links or otherwise incorporates such code be licensed under compatible licences, which are a subset of the available open-source licences. As a result, these licences have been called “viral”.

Examples of strong copyleft licences include:

1. the GNU General Public Licence (or GPL for short), which has two common primary versions - version 2 and version 3. The GPL has a complicated language (I read the GPL version 2 once and did not fully understand it) and has some additional restrictions beyond just making sure that code incorporating its code would be under compatible, open-source licences.

   Note that there is a difference between the GPL version 2, and the GPL version 3 and that they are mutually incompatible. Some people use the language “Licensed under the GNU General Public Licence, version 2, or at your option any later version” which avoids this problem. However, many projects including the Linux kernel are GPL version 2 only.

2. The Sleepycat licence. This licence was formulated by Sleepycat Software (now part of Oracle) for use in their Berkeley DB database product. The reason this licence was chosen instead of the GPL was because the source for the Berkeley DB was originally written under one of the BSD licences, and it was not clear whether it was possible to sub-license it under the GPL.

   The Sleepycat licence is short and has a simple language, and is easy to understand. From my reading of it, it seems that it is both recursive (requires code that originated from it to be licensed under the same licence) and that it mandates releasing the source code of software applications that linked against it, but under a non-specified (and possibly non-free) licence.

   Sleepycat has also provided a commercial licence for using Berkeley DB which allows incorporating it into a larger work without the need to release the source. This is a valid business model with originators of GPL-licensed software and other strong copyleft licences.

Uncertainty in Categorising Licences

The first thing I’d like to note is that it is not entirely agreed upon which licences are free/open-source and which are not.

The original Artistic Licence (authored by Larry Wall for dual-licensing perl) is:

- Considered non-free (and non-GPL-compatible) by the Free Software Foundation
- Considered free by the Debian Linux distribution.
- Considered non-free by the Red Hat Linux distributions of Fedora and Red Hat Enterprise Linux.
- Considered free by Mandriva (another Linux distributor).
- Considered open-source by the Open Source Institute - see their licence page of it on their site.
- Was considered a contract in a certain judgement.

(Who are you going to believe?)

The Artistic Licence was later clarified into the Clarified Artistic Licence which incorporated the minimum changes to make it “free” and GPL-compatible, and now has a new and improved version - Artistic 2.0, which is considered free and GPL-compatible by everybody, and is generally preferable for newer projects.

Moreover, the Apache 2.0 Licence was specifically phrased to be GPL-compatible, and is considered GPL compatible by the Apache Software Foundation, but the FSF concluded that, while it was a free-software licence, it was not compatible with the GPLv2 (although it is with the GPLv3 that was published afterwards). Their legal teams still disagree on that matter.

The Creative Commons Attribution (CC-by) and Attribution-ShareAlike Licences (CC-by-sa) are considered free (but not GPL-compatible) by the FSF (see their licences page), while the Debian legal team concluded that they were not free up to version 3.0, which was. [DFSG for Non-software]

Another curious licence is the Affero GPL - - which aims to close the "Application Service Provider Loophole". What it means is that if I install an AGPLed program on a public web-server and modify it then I must make my modifications public. However, the Free Software Definition , says that one must have "The freedom to study how the program works, and adapt it to his needs.". As a result, I personally don't consider the AGPL as free (because I may wish to run it on my publicly accessible web-server and modify it), but the FSF thinks otherwise. We have enough problems with the suitability of the GPL for embedded systems, that we don't need to kill the prospering web-apps market too.

Some Opinions about the Different Licences

The so-called "GPL vs. BSD" war probably started from as early as the time when the GPL was phrased in stark contrast to the BSD or generic public-domain licences used up to that point by the free software community. Here are some links that I can recall:

1. Ken Coar ,famous for his involvement with the Apache project, published an essay titled "Ken's Musings about Free Software and Open Source" where he explained why he prefers BSD-style licences over copyleft licences. Reading from the essay:

   > I consider myself an open sourcerer; I won't work on (or release) software covered by the GPL or any other copyleft licence except in unusual circumstances. I feel that the infective nature of copyleft licences does a disservice to the developer community and, more importantly, to the industry as a whole.
   >
   > Regardless of whether there are viable alternatives, a lot of commercial organisations are extremely reluctant to release the source code to their products, regarding the source as a trade secret. And many copyleft licences include terms that prevent any part of software covered by them from being included as part of a package that is not itself completely source-available.

2. Eric S. Raymond , an open-source developer and advocate famous for his writing of the "Cathedral and the Bazaar" series was interviewed by O'ReillyNet, and in the Interview he claimed that BSD-style licences are preferable over the GPL and the LGPL. Raymond later continued this theme in a different interview.

3. Zed Shaw published an essay titled "Why I (A/L)GPL" which was also featured on Slashdot. In my opinion, Shaw seems to speak out of a lot of cynicism there, which heavily detracts from his arguments.

4. The Free Software Foundation had published an article explaining why, in their opinion, a free software developer should use the GPL instead of the LGPL, even for libraries.

5. Elad Efrat, an Israeli developer of NetBSD, wrote a message to the Hackers-IL mailing list (in English) (Hackers in this context are any software and software development enthusiasts, not necessarily computer intruders) arguing that using the GPL for one's software is standing against the Hacker Attitude

DFSG for Non-software I should note that I have issues with the entire Debian policy of including only free-as-in-speech material in their distribution, regardless of its type. I don't feel that non-software-content (e.g: documents in human languages, graphics, audio, video, animations, non-code binary data, etc.) should be treated by the same rules as software, and even Richard Stallman said that computer games are morally allowed to have non-free graphics, sound, and plots as long as their engines are free.

So Debian are trying to be holier than the pope here and try to coerce everybody into abiding by irrational rules.

But this is beside the point of whether they are free or not.

(as phrased by Eric S. Raymond in his "How to become a Hacker" document), due to its "No problem should ever be solved twice" clause. I replied saying that copyleft licences may sometime aid in making sure that no problem should be solved twice, because they prevent proprietary solutions that later on need to be implemented as open-source software.

One can also find further emails in support of this view by Mr. Efrat in the "Hamakor discussions" mailing list archive. His opinions are not unique among proponents of permissively licensed software.

6. This IRC (=Internet Relay Chat) discussion is illustrative of various people's opinions on the GPL and BSDL licences.

Of course, this is just the tip of the iceberg. Searching the web for "gpl bsd" and similar searches will yield many other resources. So let's move on to my recommendations.

Which Licence To Choose

Now that we covered the controversy of whether some licences are open-source/free software or not, let's move on to which licence you should choose. I'll start with some bad ideas of what not to do.

Bad Idea No. 1: Choose a non-Open-Source Licence

Well, I'm not sure it is a bad idea, but choosing a licence for a non-FOSS program is out of the scope for this document. I'm not going to stop you from trying, but neither can I help you with the licence choice. My best advice is to consult a lawyer.

Bad Idea No. 2: Choose a non-GPL-Compatible Licence

David A. Wheeler wrote about why it is important to pick a GPL-compatible licence for one's software project.

Now, since version 2 of the GPL is incompatible with version 3, and vice-versa (due to the additional restrictions imposed by version 3) you shouldn't choose GPLv2-only or GPLv3-only or GPLv3-and-above, because some projects may be licensed only under one of them. Also, the version 3 of the LGPL is incompatible with the GPLv2, so you shouldn't choose it either. GPL-version-2-or-above or LGPL-version-2.1-or-above should be OK in this respect.

Bad Idea No. 3: Choose a Custom Licence

You shouldn't phrase your own licence and hope it will be considered open-source and GPL-compatible as there are plenty of open-source licences to choose from. Creating your own licence may put your users and you under risk of abuse and will contribute to the problem of licence proliferation, and may confuse your users who will need to read your unusual licence in order to study it.

Using your own custom licence deprives you of the benefit of using a well-reviewed, well-understood and debugged licence. This is similar to writing your own program from scratch instead of using an existing, well-tested software solution. (Thanks to Omer Zak for this analogy.)

So, don't do that.

Bad Idea No. 4: "Same Terms as Perl"

I originally wrote about the problem of saying your code is "licensed under the same terms as Perl itself" (which is prevalent for CPAN modules and other Perl code) on my Perl blog. It was also covered on

Perlbuzz and covered there again. I'm only going to summarise these arguments here, and conclude by saying that using the Artistic Licence Version 2.0 or later, or a GPL-compatible permissive licence such as the MIT/X11 Licence would be preferable.

Essentially, the licence of perl (the Perl implementation) has changed several times since its inception - from a non-free licence, to the GPL, to a dual GPL and Artistic licensing. Furthermore, the Artistic license is not considered free and GPL compatible by the Free Software Foundation, so it's not a wise choice, and the GPL has its share of problems too (see below) so it's not a wise choice either.

Bad Idea No. 5: Under the Public Domain

It is not wise to licence one's code under the Public Domain because the Public Domain is not widely understood or accepted internationally, and because licensing code under the Public Domain may not be possible. Rick Moen gives more reasons under "Public Domain" in the Licensing and Law section of his knowledge base.

If you're interested in making your software as-close-to-PD-as-possible, you should choose the MIT/X11 Licence or a similar licence such as the ISC licence.

Bad Idea No. 6: Using the GPL or the LGPL

The GNU General Public Licence (GPL) and GNU Lesser General Public Licence (LGPL) contain many additional restrictions to the concept of copyleft, and are very misunderstood, over-hyped, and don't maintain compatibility with newer versions. Even the LGPL is reportedly problematic

The GPL and LGPL are of more political nature than other similar FOSS licences, and as such should be avoided. I recommend using the Sleepycat licence, which is a strong-copyleft licence, that is compatible with GPLv2 and above, instead of the GPL and the Artistic 2.0 (or above) licence instead of the LGPL. And naturally permissive, non-copyleft, open-source licences such as the MIT X11 Licence are an option. What the Sleepycat licence says is that code distributed under it, must remain under it, and that one must release the source code for publicly-distributed binaries that link against it. However, from my understanding, as opposed to the GPL, this source code doesn't have to be completely free. It doesn't have the additional restrictions on licences compatibility that the GPL has, nor does it have the other obscure and not-well-understood restrictions that the GPL has.

I read the GPLv2 originally once and couldn't understand it. The LGPLv2 or the GPLv3 would likely prove to be more problematic. I find it harder to trust lengthy documents that I am unable to understand. I had no problem understanding the Sleepycat licence, the Artistic 2.0 licence, or the MIT X11 and BSD licences, so I can better recommend them instead.

One fact I should note is that the GPL often stands against the Hacker's attitude as presented by Eric Raymond in the document "How to become a Hacker". It states that:

- The world is full of fascinating problems waiting to be solved.
- No problem should ever have to be solved twice.

For example, the Free Software Foundation now started the GNU PDF project that is licensed under the GPL version 3, because all the other Free software PDF projects are GPL version 2 only. So because the GPL was used, the same problem need to be solved twice.

Another case where it happened was this story of an the Inkscape set operations patch:

> Once before, someone had contributed a patch to add boolean operations, but that patch relied on a polygon clipping library provided under an incompatible license. There's

> little more frustrating than having a solution in hand, only to be hamstrung by legal problems. Even though it was an important feature for us, we regretfully postponed development of it into the distant future on our roadmap and proceeded with other work.

Furthermore, the OpenBSD project are now re-implementing a lot of software that is only available as GPL or similar licences, under BSD-style licences, due to OpenBSD's more pedantic licensing policy.

And finally, often one can build upon non-GPL-like-code and not release the derived work as free and open source software, without it causing any harm, and actually causing a lot of good. Which is otherwise prevented if the code is GPLed.

Why I Prefer the MIT X11 Licence

I personally accept the fact that some developers would like to use weak copyleft or strong copyleft licences for their works. However, I still prefer to use the MIT X11 Licence for works that I originate and now I'd like to explain why:

- The MIT X11 Licence is short, simple, and easy to understand.
- It is practically public domain which is what people innocently expect when they hear terms like "open-source" or "free-as-in-speech software".
- By using such permissive licences, one has much fewer worries on how people can violate one's licence. While I commonly hear about GPL violations, I don't ever recall hearing about a violation of a permissive, BSD-style licence.
- The MIT X11 licence still gives one protection against litigation with its "NO WARRANTY" clause.
- The MIT X11 licence is GPL-compatible, so it can be made useful to people who are using the GPL.
- By using such a licence one can boast that one's program is Public Domain / BSD-style-licensed.

Strong Copyleft and Preventing Threats to FOSS

When I posted an early draft of this essay to the Hackers-IL mailing list, I received a comment about the fact that GPL (or other strong copyleft licences) prevent threats to free and open source software. I'd like to reply to it here.

Quoting from the comment:

> In the second half of the 90s, X-Windows was quite popular - a department (in universities, companies, etc.) would have strong Unix workstations (from Sun, DEC, SGI, HP and other vendors) and people would have cheaper machines showing the output from the strong machine using X-Windows. As MS-Windows grew popular, people wanted to use their PC running Windows to display X-Windows sessions. But unfortunately, the only X server available for Windows was commercial software (Exceed), which could happen because X was BSD-licensed and not GPL. Users (at the time, most had corporate or university funds - they weren't home users) bit the bullet and paid. It took literally years before a free X server for Windows became available.

My response is that this was indeed a problem and inconvenienced the users of Exceed. However, if X-Windows were GPLed, then the people who made Exceed and wanted to sell it, would not have made

it in the first place, because they had to make it GPLed and keep it as FOSS. So either they would have implemented it from scratch or not at all. It is possible that a different group would have created a free X server for Windows, but it is possible that no free X server would have been available at all. And if a different group could have created a free X server for Windows without Exceed, they could have certainly created it with it.

Furthermore, If X11 had been initiated under a non-BSD-style-licence, then it is possible it would not have become as ubiquitous as it is in the UNIX world, thus making it irrelevant to port it to Windows in the first place. We can't tell that for sure, but I think it is a possibility.

Quoting again:

> > For example, someone integrated my Freecell Solver library in a shareware (without source) game he wrote titled Freecell 3D.
> >
> > He sells the game online. It did not do any damage, because there are plenty of Freecell implementations around. I on my part am pretty happy with the open-source PySolFC.
>
> Let's imagine that you wrote a great Freecell solver library, but you suck at UI and failed to create a graphical game based on that library, and wish that someone created a UI on top of your library. If your library is BSD, someone could create this UI but make it commercial and not even you can use it. If your library is GPL, this person is forced to either make his code public (so you and your friends can use it) or pay you for a new license. Which of those scenarios will make you happier? I can't see how the GPL isn't better in this scenario.

Well, first of all I should note that that is a lot of "if"s. When I started working on Freecell Solver, there were already plenty of graphical Freecell games, possibly with support for other variants of Solitaire. And there were also several open-source ones, including some very polished ones.

As a result, I could have easily integrated my solver into one of them. So his thought experiment is not very realistic, although it may be valid in the general case.

Back to the thought experiment, a person who would be interested in creating a commercial Freecell UI and still integrate a solving mechanism to it could:

1. Develop his own Freecell solving library from scratch. That will probably be less work than developing the rest of the graphical Freecell game.

2. Write a command line executable that uses the library, invoke it, and process its output. This strategy was taken by the developers of the open-source PySolFC out of convenience using my existing "fc-solve" executable.

3. Write a GPLed server for solving Freecell that the graphical game would communicate with using TCP/IP or a different IPC mechanism.

4. Decide that developing a graphical game only to later release it as free software won't be worth his time, and as a result not develop it at all.

In all of these cases, we won't have a graphical game as FOSS.

Finally I'd rather have a proprietary derived work than no derived program at all, or that instead someone will duplicate my effort in creating a BSD-style or a proprietary replacement for my work.