Software licenses

A software license is a legal document that accompanies a program. Without a software license, according to the provisions laid down within the Berne copyright convention, *a program cannot be distributed or modified without the explicit permission of the authors*.

A free software license, on the other hand, allows you to study, redistribute, modify and release the modified program. This is possible because you, the copyright holder, give up some of your rights, by licensing others to copy, modify and distribute your code, at times under some conditions.

Proprietary licenses

A license that is neither free nor semi-free is said to be *proprietary*. Proprietary licenses come in a variety of flavours, and do not generally allow access to the source code or modifications. A brief description follows of some of the terms most commonly used when distributing software with a proprietary license.

No license

Even if this is not a license, the final effect is that of a proprietary license. Every program that is not accompanied by a copyright license is subject to the Berne international copyright convention, and can not be distributed or modified without the explicit consent of the copyright holders. This means that the program is not free without a free copyright license, even when the source is available, with or without charge.

Freeware

Without any other specifications, a program is said to be *freeware* if it can be copied freely at no charge, for any use. The program does not usually come with source, or has some other restriction that makes it not free. A notable example is Adobe Acrobat reader.

Shareware

A *shareware* program can be freely copied and distributed, but cannot be rightfully used without paying for it after an initial period of test usage. Some of these programs stop working after the test period, or allow limited functionality until you purchase a key that unlocks them.

No source available

Most proprietary programs are distributed in binary-only form. The binary may have or not limits on its usage. See, for an example, the license of the Macromedia Flash Plugin.

Non disclosure agreement

Sometimes programs are distributed only after the recipient has signed a contract (the *agreement*) where the signer agrees not to divulge some specific information about the program. This is common when the source is distributed but cannot be further redistributed. For example, the signer can agree not to divulge information about the details of how the program works.

Restricted use

A common constraint that makes a license non-free is the *non-commercial* clause. Many programs are distributed as *free for non-commercial use*, i.e. they can be used without charge provided that no money is gained from their use. Similar clauses are *for personal use*, *for academic use*, *for educational use*. Other restrictions include the *non transferable* clause, which prevents free redistribution. Such programs may come with or without source, usually without.

Semi-free software

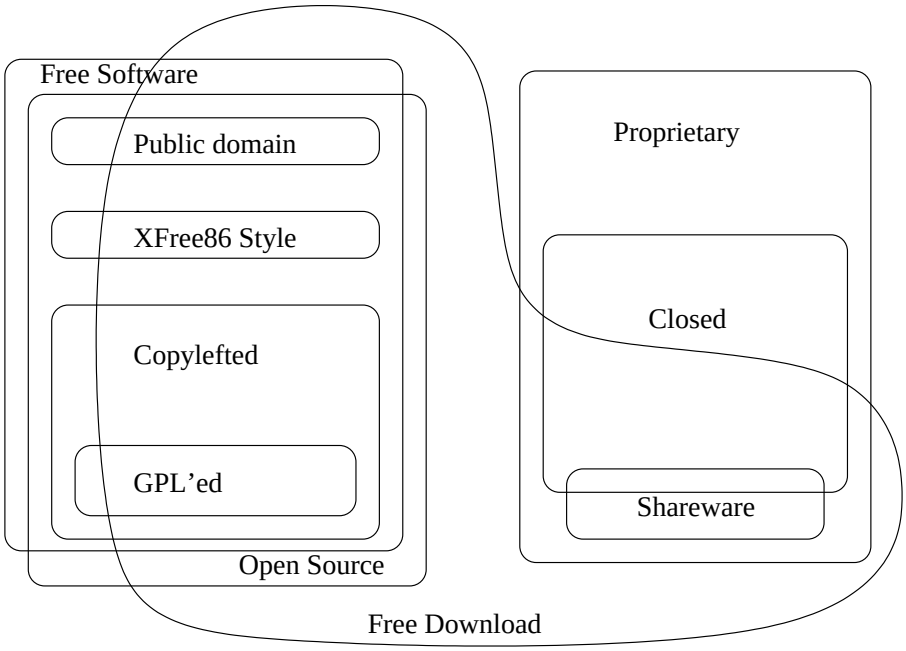
Software that comes with source code, with permission to use, copy, modify and distribute for limited usage is called *semi-free*. A well written semifree license for non commercial use is the shared license used by Microsoft. Another example is the ns-2 network simulator, parts of which are licensed only for non-commercial use, while other parts can only be modified for evaluation and research purposes.

This means that commercial Linux distributions oriented to the science and research market cannot include the ns-2 network simulator, unless the non-commercial parts are removed, which is a difficult and perhaps impossible task. Moreover, since the non-commercial clause also includes documentation, no book about ns-2 can be sold if it describes the parts covered by that license.

Free software licenses

An elegant definition of a free license was given for the first time by Richard Stallman in the Eighties. An equivalent, more detailed definition, was given in the Nineties by the Open Source Initiative.

Free software licenses allow any recipient of a program to use it for any purpose, to copy, modify, and redistribute it. Whether money is charged for distribution has no relevance. The source code must be included or requested at no cost other than shipping. If the recipient can only redistribute the software under the same licensing terms, the term *copyleft* license is used; otherwise we speak about a *non-copyleft* license. The following picture by Chao-Kuei tries to draw a relationship among the different types of software licenses described so far.



Choosing a license when writing new free software

In order to make a program free, you should accompany the source code with a file (usually named `LICENSE.TXT`) containing the text of the license you choose. Moreover, it is highly recommended to add a copyright line at the top of each source file, similar to the one found at the end of this document. When you modify the file, make sure that you add the current year to the list of years in the copyright line. If entry of the "©" character is a problem, use the three characters "(C)" instead. After the copyright line, enter one more line stating that the copyright licensing terms are contained in the accompanying file `LICENSE.TXT`.

Do not try to write your own license. It is very difficult to come up with something well written, even if you are a lawyer. Also, the use of an established license has many advantages: people are familiar with it and know the implications, they are written by lawyers, they have long been tested on the field, and they make it easier to share code between free software projects.

What license should you choose for your program? It depends on your aims. As also mentioned in the Debian Free Software Guidelines FAQ draft and in an article by David A. Wheeler, we suggest using one of the three following licenses.

The GNU General Public License

If you want to be sure that all copies of your program, even modified ones, are accompanied by the corresponding source, or by an easy way of getting the source, you should use the *GNU GPL*.

This is the most successful license of all, used for the programs developed by the GNU Project and many others. Examples include the Linux kernel, the GNU Emacs editor, the portable GCC compiler, the KDE and Gnome desktops for X.

The GPL is the first and foremost example of a copyleft license. Whoever modifies copylefted software and distributes the modifications must do so according to the same terms as the modified program. In other words, they must give others the same freedom that they had themselves. More precisely, the distribution of any work derived from a GPL-licensed program must be under a GPL license or a license that grants the same freedoms. This includes linking your program with others, because the result is considered a derived work. However, the license does not cover calling your program from others, or using it as part of a collections of programs, or merely aggregating it with other programs on the same support: all these uses of the program are permitted without constraints.

The GPL protects the reputation of authors by requiring that each modified version that is distributed is clearly labelled as modified. However, no obligations exist to distribute modified versions. For example, an organisation may modify a GPL program for internal use only, and would not be required to release the modification to anyone.

It normally advisable to link your program with libraries released under a GPL-compatible license, such as the GNU LGPL or the new BSD license. However, if you are forced to use libraries that are not released under a GPL-compatible license, then the issue needs careful examination. Start by looking at the GPL FAQ, which contains a great number of questions about GNU GPL. A particularly interesting answer regards binding to modules, such as those provided by many modern languages (Perl, Python, Java). Start from that question, then read about adding exceptions to the GPL for your program. There is also an email from Linus Torvalds, dated 19 Oct 2001, which offers precious insight into this kind of problems.

The GNU Lesser General Public License

If you want to be sure that your software remains free even after modification, and you want to leave people free to link it against any program, be it free or proprietary, you should use the *GNU LGPL*.

The LGPL is a GPL compatible license, used mostly for libraries, plugins and components. Notable software available under the LGPL includes the i386 emulator Bochs, many of the Gnome desktop libraries, the C library Glibc, and the C++ library libg++.

The LGPL is a copyleft license, just like the GPL. However, software released under the LGPL may be linked against non GPL-compatible software, even proprietary software, provided that the source of the LGPL part is made available, and that the LGPL part is upgradeable independently from the rest of the program linked against it. This is usually feasible if the LGPL part is a shared library, a DLL, a loadable module or a component.

Using the LGPL is a mixed blessing. The Free Software Foundation explicitly advises against using the LGPL, in order to further the scope of GPL-covered software. If your intent is similar to that of the FSF, your best bet is to read and understand why they prefer the GPL instead. If, on the other hand, you are concerned about making your software free and not so much about the software of people who will link against your libraries or modules, the LGPL may be okay for you.

The LGPL is compatible with the GPL; in other words you can take a piece of LGPL code, incorporate it into your GPL program, and release everything under the GPL. This is an important property of the license.

The new BSD and the MIT X license

If you aim at maximum diffusion for your code by allowing its use in any program, be it free or proprietary, you should use the “new BSD license” (also called the “BSD license without the advertising clause”) or the “MIT X license”.

Successful programs released under one of these licenses include the kernel and core utilities of the FreeBSD operating system, the XFree86 X server, many networking utilities, and the Apache web server.

The new BSD license gives the user the most freedom of all. Only public domain software gives even more freedom, but public domain does not exist in Europe, as an author cannot give up authorship. The users can do whatever they want with the software, including releasing it as proprietary software, whether modified or not. The drawback is that people can take it, improve it, and distribute it in binary-only form, keeping the modifications secret, without giving back the freedom they got. However, if the original authors represent a group that is the most competent on that specific topic, and is committed to keep releasing the improvements with the same license, this risk does not exist, and the software is given the widest possible audience. An example of this is the way the TCP/IP stack evolved from the original BSD source: even though most software producers took code from the BSD stack and put it into their proprietary programs, all the new developments went into the original code base, and the best of the BSD stack is still free to date.

The new BSD and the MIT X licenses are compatible with the GPL, i.e. you can take a piece of MIT X code, incorporate it into your GPL program, and release everything under the GPL. The original BSD license, however, is not compatible because of the advertising clause, which imposes that a note is displayed in all advertising material distributed with the program. Since the GPL prohibits any additional obligations on the distribution of software, the original BSD license is not compatible with

the GPL.

Contributing free software

If you are contributing to a free program, rather than writing it from scratch, your best bet is to use the same license that the program adopts. While it is often possible to do things differently, usually this is not advisable.

One reason to avoid mixing different licenses for the same program is that their compatibility is not always obvious. It has happened in the past that incompatibilities and legal license loopholes have been discovered only after long time, putting the rights of the authors and the users at risk. Another reason is that, if the program you are modifying has a community developed around it, that community may be hostile with respect to changing the license of the program, so your changes may get a bad reputation based not on their technical merits, but on the license with which they are distributed.

For example, if you want to contribute a patch to the *Linux kernel*, you will have to release it under the GNU GPL, as this is the license used for the rest of the kernel, and it would not be possible to do otherwise. On the other hand, if you want to contribute a new module to the *ns-2 network simulator*, it is preferable to release it under the new BSD license, even if it could be released under the LGPL, or even the GPL with an exception clause. However, since the rest of the ns-2 code (which is huge) is mostly covered by the new BSD license, a different license would risk meeting with hostility in the community that manages the program.

Some free software entities may ask you for a copyright assignment or a copyright disclaimer. The Apache foundation and the Free Software Foundation may ask you to assign your copyright to them , or at least to disclaim any copyright interest in the work that you submit for inclusion in their free software projects. The reason is that, if ever a copyright litigation goes to court, it will be much easier for the FSF or the Apache foundation to defend their rights as the only copyright owner than to convince a myriad of developers to do so, many of whom may be uninterested in the issue or simply impossible to trace.

In any case, contributing to an already established body of software entails not only technical merits and the right choice of software license, but also requires a knowledge of the community that has built up around it, and the ways that the community uses to accept new contributions. Sometimes there are only one or two main developers, other times there are many in charge of different parts. In any case, playing by the rules is a precondition to be taken seriously.

Software documentation licenses

Software normally comes with documentation, which is generally an integral part of the distribution. Many free software authors like to use the same license for the software and the documentation. While this is arguably incorrect, because software licenses make explicit reference to software in their wording, it is an established usage, and has not given rise to legal problems. Consequently, using the same software license for the documentation accompanying your software is a sure way to go.

However, you may want to use a license that is specifically written for documentation. There are two interesting alternatives, the GFDL and the Creative Commons licenses.

GNU Free Documentation License

If you mean to write a complete book to be printed and sold, contents of which are freely reproducible and modifiable, the GFDL is a good option. It is a complex copyleft license, with many optional parts, including book covers, history of changes, invariant sections, which make it suitable for a complex product such as a book. It is used a great deal by the FSF for the documentation of the GNU project. However, it has received much criticism, mostly because abuse of invariant sections can cause problems when creating small derivative works, and can make them accumulate over time.

Creative Commons licenses

If you want a simple copyleft free license, you can go with the Attribution-ShareAlike creative commons license. Or, if you would prefer a free non-copyleft license, you can use the Attribution creative commons license. Both allow free use, commercial use and modification of the work, on the condition that the original author is given credit. Both allow redistribution, but the copyleft version requires that distribution be done under the same exact license. Both licenses are being translated and worked on by lawyers of different countries in order to adapt them more closely to different legal systems.

Updated: 2009-12-11