free SOFTWARE FOUNDATION

Detailed Study and Analysis of the GPL and LGPL

Stanford University, Stanford, CA, USA

Tuesday, 24 August 2004

Bradley M. Kuhn
Executive Director
Free Software Foundation

Daniel Ravicher
Senior Counsel
Free Software Foundation
President and Executive Director
Public Patent Foundation

Detailed Study and Analysis of the GPL and LGPL

Stanford University, Stanford, CA, 24 August 2004

09:00 - 09:25	Registration / Check-in / Continental Breakfast
09:25 - 09:30	Welcome
09:30 - 10:00	Free Software Principles and the Free Software Definition *Bradley M. Kuhn*
10:00 - 10:10	Preamble of the GNU General Public License (GPL) *Bradley M. Kuhn*
10:10 - 10:35	GPL, §0: Definitions, etc. *Bradley M. Kuhn*
10:35 - 10:50	GPL, §1: Grant for Verbatim Source Copying *Bradley M. Kuhn*
10:50 - 11:00	Q & A
11:00 - 11:10	Break
11:10 - 11:55	Derivative Works: Statute and Case Law *Daniel Ravicher*

11:55 - 12:20	GPL, §2: Grants for Source Derivative Works *Bradley M. Kuhn*
12:20 - 12:30	Q & A
12:30 - 14:00	Lunch with Lecture "Patents and Free Software" *Prof. Eben Moglen*
14:00 - 14:20	GPL, §3 Grants for Creating Binary Derivative Works *Bradley M. Kuhn*
14:20 - 14:40	The Implied Patent Grant in the GPL *Daniel Ravicher*
14:40 - 15:25	GPL, §4: Termination of License GPL, §5: Acceptance of License GPL, §6: Prohibition on Further Restrictions GPL, §7: Conflicts with other Agreements or Orders GPL, §8: International Licensing Issues GPL, §10: Copyright Holder's Exceptions to the GPL *Bradley M. Kuhn*
15:25 - 15:35	GPL, §11: Disclaimer of Warranties GPL, §12: Limitation of Liability *Daniel Ravicher*
15:35 - 15:45	Q & A
15:45 - 16:00	Break
16:00 - 17:30	GNU Lesser General Public License (LGPL) *Bradley M. Kuhn*
17:30 - 18:00	Q & A

Preface

This one-day course gives a section-by-section explanation of the most popular Free Software copyright license, the GNU General Public License (GNU GPL), and teaches lawyers, software developers, managers and business people how to use the GPL (and GPL'd software) successfully in a new Free Software business and in existing, successful enterprises.

Attendees should have a general familiarity with software development processes. A vague understanding of how copyright law applies to software is also helpful. The tutorial is of most interest to lawyers, software developers and managers who run software businesses that modify and/or redistribute software under terms of the GNU GPL (or who wish to do so in the future), and those who wish to make use of existing GPL'd software in their enterprise.

Upon completion of the tutorial, successful attendees can expect to have learned the following:

- The freedom-defending purpose of each term of the GNU GPL
- The redistribution options under the GPL
- The obligations when modifying GPL'd software
- How to build a plan for proper and successful compliance with the GPL
- The business advantages that the GPL provides
- The most common business models used in conjunction with the GPL
- How existing GPL'd software can be used in existing enterprises
- The basics of the LGPL and how it differs from the GPL
- How best to understand the complexities regarding derivative works of software

These course materials are merely a summary of the highlights of the course presented. Please be aware that during the actual GPL course, class discussion supplements this printed curriculum. Simply reading it is not equivalent to attending the course.

Contents

Chapter 1

What Is Free Software?

Consideration of the GNU General Public License (herein, abbreviated as *GNU GPL* or just *GPL*) must begin by first considering the broader world of Free Software. The GPL was not created from a void, rather, it was created to embody and defend a set of principles that were set forth at the founding of the GNU project and the Free Software Foundation (FSF)—the organization that upholds, defends and promotes the philosophy of software freedom. A prerequisite for understanding the GPL and its terms and conditions is a basic understanding of the principles behind it. The GPL is unlike almost all other software licenses in that it is designed to defend and uphold these principles.

1.1 The Free Software Definition

The Free Software Definition is set forth in full on FSF's Web site at `http://www.fsf.org/ philosophy/free-sw.html`. This section presents an abbreviated version that will focus on the parts that are most pertinent to the terms of the GPL.

A particular program is Free Software if it grants a particular user of that program, the following freedoms:

- The freedom to run the program for any purpose
- The freedom to change and modify the program
- The freedom to copy and share the program
- The freedom to share improved versions of the program

The focus on "a particular user" is very pertinent here. It is not uncommon for the same version of a specific program to grant these freedoms to some subset of its user base, while others have none or only some of these freedoms. Section 9.2 talks in detail about how this can happen even if a program is released under the GPL.

Some people refer to software that gives these freedoms as "Open Source." Besides having a different political focus than those who call it Free Software,[1] those who call the software "Open Source" are focused on a side issue. User access to the source code of a program is a prerequisite to make use of the freedom to modify. However, the important issue is what freedoms are granted in the license of that source code. Microsoft's "Shared Source" program, for example, gives various types of access to source code, but almost none of the freedoms described in this section.

One key issue central to these freedoms is that there are no restrictions on how these freedoms can be exercised. Specifically, users and programmers can exercise these freedoms noncommercially or commercially. Licenses that grant these freedoms for noncommercial activities but prohibit them for commercial activities are considered non-Free.

In general, software for which most or all of these freedoms are restricted in any way is called "non-Free Software." Typically, the term "proprietary software" is used more or less interchangeably with "non-Free Software." Personally, I tend to use the term "non-Free Software" to refer to noncommercial software that restricts freedom (such as "shareware") and "proprietary software" to refer to commercial software that restricts freedom (such as nearly all of Microsoft's and Oracle's offerings).

The remainder of this section considers each of the four freedoms in detail.

1.1.1 The Freedom to Run

For a program to be Free Software, the freedom to run the program must be completely unrestricted. This means any use for software the user can come up with must be permitted. Perhaps, for example, the user has discovered an innovative use for a particular program, one that the programmer never could have predicted. Such a use must not be restricted.

It was once rare that this freedom was restricted by even proprietary software; today it is not so rare. Most End User Licensing Agreements

[1]The political differences between the Free Software Movement and the Open Source Movement are documented on FSF's Web site at `http://www.fsf.org/philosophy/free-software-for-freedom.html`.

(EULAs) that cover most proprietary software restrict some types of use. For example, some versions of Microsoft's FrontPage software prohibit use of the software to create Web sites that generate negative publicity for Microsoft. Free Software has no such restrictions; everyone is free to use Free Software for any purpose whatsoever.

1.1.2 The Freedom to Change and Modify

Free Software programs allow users to change, modify and adapt the software to suit their needs. Access to the source code and related build scripts are an essential part of this freedom. Without the source code and the ability to build the binary applications from that source, the freedom cannot be properly exercised.

Programmers can take direct benefit from this freedom, and often do. However, this freedom is also important to users who are not programmers. Users must have the right to exercise this freedom indirectly in both commercial and noncommercial settings. For example, users often seek noncommercial help with the software on email lists and in users groups. When they find such help, they must have the freedom to recruit programmers who might altruistically assist them to modify their software.

The commercial exercise of this freedom is also essential for users. Each user, or group of users, must have the right to hire anyone they wish in a competitive free market to modify and change the software. This means that companies have a right to hire anyone they wish to modify their Free Software. Additionally, such companies may contract with other companies to commission software modification.

1.1.3 The Freedom to Copy and Share

Users may share Free Software in a variety of ways. Free Software advocates work to eliminate a fundamental ethical dilemma of the software age: choosing between obeying a software license, and friendship (by giving away a copy of a program to your friend who likes the software you are using). Free Software licenses, therefore, must permit this sort of altruistic sharing of software among friends.

The commercial environment must also have the benefits of this freedom. Commercial sharing typically takes the form of selling copies of Free Software. Free Software can be sold at any price to anyone. Those who redistribute Free Software commercially have the freedom to selectively distribute (you can pick your customers) and to set prices at any level the redistributor sees fit.

It is true that many people get copies of Free Software very cheaply (and sometimes without charge). The competitive free market of Free Software tends to keep prices low and reasonable. However, if someone is willing to pay a billion dollars for one copy of the GNU Compiler Collection, such a sale is completely permitted.

Another common instance of commercial sharing is service-oriented distribution. For example, a distribution vendor may provide immediate security and upgrade distribution via a special network service. Such distribution is completely permitted for Free Software.

(Section 9.2 of this tutorial talks in detail about various Free Software business models that take advantage of the freedom to share commercially.)

1.1.4 The Freedom to Share Improvements

The freedom to modify and improve is somewhat empty without the freedom to share those improvements. The Free Software community is built on the pillar of altruistic sharing of improved Free Software. Inevitably, a Free Software project sprouts a mailing list where improvements are shared freely among members of the development community. Such noncommercial sharing must be permitted for Free Software to thrive.

Commercial sharing of modified Free Software is equally important. For commercial support to exist in a competitive free market, all developers — from single-person contractors to large software companies — must have the freedom to market their services as improvers of Free Software. All forms of such service marketing must be equally available to all.

For example, selling support services for Free Software is fully permitted. Companies and individuals can offer themselves as “the place to call” when software fails or does not function properly. For such a service to be meaningful, the entity offering that service must have the right to modify and improve the software for the customer to correct any problems that are beyond mere user error.

Entities must also be permitted to make available modified versions of Free Software. Most Free Software programs have a “standard version” that is made available from the primary developers of the software. However, all who have the software have the “freedom to fork” — that is, make available nontrivial modified versions of the software on a permanent or semi-permanent basis. Such freedom is central to vibrant developer and user interaction.

Companies and individuals have the right to make true value-added versions of Free Software. They may use freedom to share improvements to distribute distinct versions of Free Software with different functionality and features. Furthermore, this freedom can be exercised to serve a disenfranchised subset of the user community. If the developers of the standard version refuse to serve the needs of some of the software's users, other entities have the right to create a long- or short-lived fork to serve that sub-community.

1.2 How Does Software Become Free?

The last section set forth the freedoms and rights respected by Free Software. It presupposed, however, that such software exists. This section discusses how Free Software comes into existence. But first, it addresses how software can be non-Free in the first place.

Software can be made proprietary only because it is governed by copyright law.[2] Copyright law, with respect to software, governs copying, modifying, and redistributing that software.[3] By law, the copyright holder (a.k.a. the author) of the work controls how others may copy, modify and/or distribute the work. For proprietary software, these controls are used to prohibit these activities. In addition, proprietary software distributors further impede modification in a practical sense by distributing only binary code and keeping the source code of the software secret.

Copyright law is a construction. In the USA, the Constitution permits, but does not require, the creation of copyright law as federal legislation. Software, since it is an idea fixed in a tangible medium, is thus covered by the statues, and is copyrighted by default.

However, this legal construction is not necessarily natural. Software, in its natural state without copyright, is Free Software. In an imaginary world with no copyright, the rules would be different. In this world, when you received a copy of a program's source code, there would be no default legal system to restrict you from sharing it with others, making modifications, or

[2]This statement is a bit of an oversimplification. Patents and trade secrets can cover software and make it effectively non-Free, one can contract away their rights and freedoms regarding software, or source code can be practically obscured in binary-only distribution without reliance on any legal system. However, the primary control mechanism for software is copyright.

[3]Copyright law in general also governs "public performance" of copyrighted works. There is no generally agreed definition for public performance of software and version 2 of the GPL does not govern public performance.

redistributing those modified versions.[4]

Software in the real world is copyrighted by default and is automatically covered by that legal system. However, it is possible to move software out of the domain of the copyright system. A copyright holder is always permitted to *disclaim* their copyright. If copyright is disclaimed, the software is not governed by copyright law. Software not governed by copyright is in the "public domain."

1.2.1 Public Domain Software

An author can create public domain software by disclaiming all copyright interest on the work. In the USA and other countries that have signed the Berne convention on copyright, software is copyrighted automatically by the author when she "fixes the software into a tangible medium." In the software world, this usually means typing the source code of the software into a file.

However, an author can disclaim that default control given to her by the copyright laws. Once this is done, the software is in the public domain — it is no longer covered by copyright. Since it is copyright law that allows for various controls on software (i.e., prohibition of copying, modification, and redistribution), removing the software from the copyright system and placing it into the public domain does yield Free Software.

Carefully note that software in the public domain is *not* licensed in any way. It is nonsensical to say software is "licensed for the public domain," or any phrase that implies the copyright holder gave expressed permission to take actions governed by copyright law.

By contrast, what the copyright holder has done is renounce her copyright controls on the work. The law gave her controls over the work, and she has chosen to waive those controls. Software in the public domain is absent copyright and absent a license. The software freedoms discussed in Section 1.1 are all granted because there is no legal system in play to take them away.

1.2.2 Why Copyright Free Software?

If simply disclaiming copyright on software yields Free Software, then it stands to reason that putting software into the public domain is the easiest and most straightforward way to produce Free Software. Indeed, some major Free Software projects have chosen this method for making their software

[4]There could still exist legal systems, like our modern patent system, which could restrict the software in other ways.

Free. However, most of the Free Software in existence *is* copyrighted. In most cases (particularly in those of FSF and the GNU Project), this was done due to very careful planning.

Software released into the public domain does grant freedom to those users who receive the standard versions on which the original author disclaimed copyright. However, since the work is not copyrighted, any nontrivial modification made to the work is fully copyrightable.

Free Software released into the public domain initially is Free, and perhaps some who modify the software choose to place their work into the public domain as well. However, over time, some entities will choose to proprietarize their modified versions. The public domain body of software feeds the proprietary software. The public commons disappears, because fewer and fewer entities have an incentive to contribute back to the commons. They know that any of their competitors can proprietarize their enhancements. Over time, almost no interesting work is left in the public domain, because nearly all new work is done by proprietarization.

A legal mechanism is needed to redress this problem. FSF was in fact originally created primarily as a legal entity to defend software freedom, and that work of defending software freedom is a substantial part of its work today. Specifically because of this "embrace, proprietarize and extend" cycle, FSF made a conscious choice to copyright its Free Software, and then license it under "copyleft" terms. Many, including the developers of the kernel named Linux, have chosen to follow this paradigm.

Copyleft is a legal strategy to defend, uphold and propagate software freedom. The basic technique of copyleft is as follows: copyright the software, license it under terms that give all the software freedoms, but use the copyright law controls to ensure that all who receive a copy of the software have equal rights and freedom. In essence, copyleft grants freedom, but forbids others to forbid that freedom to anyone else along the distribution and modification chains.

Copyleft is a general concept. Much like ideas for what a computer might do must be *implemented* by a program that actually does the job, so too must copyleft be implemented in some concrete legal structure. "Share and share alike" is a phrase that is used often enough to explain the concept behind copyleft, but to actually make it work in the real world, a true implementation in legal text must exist. The GPL is the primary implementation of copyleft in copyright licensing language.

1.3 An Ecosystem of Equality

The GPL uses copyright law to defend freedom and equally ensure users' rights. This ultimately creates an ecosystem of equality for both business and noncommercial users.

1.3.1 The Noncommercial Ecosystem

A GPL'd code base becomes a center of a vibrant development and user community. Traditionally, volunteers, operating noncommercially out of keen interest or "scratch an itch" motivations, produce initial versions of a GPL'd system. Because of the efficient distribution channels of the Internet, any useful GPL'd system is adopted quickly by noncommercial users.

Fundamentally, the early release and quick distribution of the software gives birth to a thriving noncommercial community. Users and developers begin sharing bug reports and bug fixes across a shared intellectual commons. Users can trust the developers, because they know that if the developers fail to address their needs or abandon the project, the GPL ensures that someone else has the right to pick up development. Developers know that the users cannot redistribute their software without passing along the rights granted by GPL, so they are assured that every one of their users is treated equally.

Because of the symmetry and fairness inherent in GPL'd distribution, nearly every GPL'd package in existence has a vibrant noncommercial user and developer base.

1.3.2 The Commercial Ecosystem

By the same token, nearly all established GPL'd software systems have a vibrant commercial community. Nearly every GPL'd system that has gained wide adoption from noncommercial users and developers eventually begins to fuel a commercial system around that software.

For example, consider the Samba file server system that allows Unix-like systems (including GNU/Linux) to serve files to Microsoft Windows systems. Two graduate students originally developed Samba in their spare time and it was deployed noncommercially in academic environments. However, very soon for-profit companies discovered that the software could work for them as well, and their system administrators began to use it in place of Microsoft Windows NT file-servers. This served to lower the cost of running such servers by orders of magnitude. There was suddenly room in Windows file-server budgets to hire contractors to improve Samba. Some of the first people hired to do such work were those same two graduate students who originally developed the software.

The noncommercial users, however, were not concerned when these two fellows began collecting paychecks off of their GPL'd work. They knew that because of the nature of the GPL that improvements that were distributed in the commercial environment could easily be folded back into the standard version. Companies are not permitted to proprietarize Samba, so the noncommercial users, and even other commercial users are safe in the knowledge that the software freedom ensured by GPL will remain protected.

Commercial developers also work in concert with noncommercial developers. Those two now-long-since graduated students continue to contribute to Samba altruistically, but also get paid work doing it. Priorities change when a client is in the mix, but all the code is contributed back to the standard version. Meanwhile, many other individuals have gotten involved noncommercially as developers, because they want to "cut their teeth on Free Software," or because the problems interest them. When they get good at it, perhaps they will move on to another project, or perhaps they will become commercial developers of the software themselves.

No party is a threat to another in the GPL software scenario because everyone is on equal ground. The GPL protects rights of the commercial and noncommercial contributors and users equally. The GPL creates trust, because it is a level playing field for all.

1.3.3 Law Analogy

In his introduction to Stallman's *Free Software, Free Society*, Lawrence Lessig draws an interesting analogy between the law and Free Software. He argues that the laws of a free society must be protected much like the GPL protects software. So that I might do true justice to Lessig's argument, I quote it verbatim:

> A "free society" is regulated by law. But there are limits that any free society places on this regulation through law: No society that kept its laws secret could ever be called free. No government that hid its regulations from the regulated could ever stand in our tradition. Law controls. But it does so justly only when visibly. And law is visible only when its terms are knowable and controllable by those it regulates, or by the agents of those it regulates (lawyers, legislatures).

This condition on law extends beyond the work of a legislature. Think about the practice of law in American courts. Lawyers are hired by their clients to advance their clients' interests. Sometimes that interest is advanced through litigation. In the course of this litigation, lawyers write briefs. These briefs in turn affect opinions written by judges. These opinions decide who wins a particular case, or whether a certain law can stand consistently with a constitution.

All the material in this process is free in the sense that Stallman means. Legal briefs are open and free for others to use. The arguments are transparent (which is different from saying they are good), and the reasoning can be taken without the permission of the original lawyers. The opinions they produce can be quoted in later briefs. They can be copied and integrated into another brief or opinion. The "source code" for American law is by design, and by principle, open and free for anyone to take. And take lawyers do—for it is a measure of a great brief that it achieves its creativity through the reuse of what happened before. The source is free; creativity and an economy is built upon it.

This economy of free code (and here I mean free legal code) doesn't starve lawyers. Law firms have enough incentive to produce great briefs even though the stuff they build can be taken and copied by anyone else. The lawyer is a craftsman; his or her product is public. Yet the crafting is not charity. Lawyers get paid; the public doesn't demand such work without price. Instead this economy flourishes, with later work added to the earlier.

We could imagine a legal practice that was different—briefs and arguments that were kept secret; rulings that announced a result but not the reasoning. Laws that were kept by the police but published to no one else. Regulation that operated without explaining its rule.

We could imagine this society, but we could not imagine calling it "free." Whether or not the incentives in such a society would be better or more efficiently allocated, such a society could not be known as free. The ideals of freedom, of life within a free society, demand more than efficient application. Instead, openness and transparency are the constraints within which a legal system gets built, not options to be added if convenient to the

> leaders. Life governed by software code should be no less.
>
> Code writing is not litigation. It is better, richer, more productive. But the law is an obvious instance of how creativity and incentives do not depend upon perfect control over the products created. Like jazz, or novels, or architecture, the law gets built upon the work that went before. This adding and changing is what creativity always is. And a free society is one that assures that its most important resources remain free in just this sense.[5]

In essence, lawyers are paid to service the shared commons of legal infrastructure. Few citizens defend themselves in court or write their own briefs (even though they are legally permitted to do so) because everyone would prefer to have an expert do that job.

The Free Software economy is a market ripe for experts. It functions similarly to other well established professional fields like the law. The GPL, in turn, serves as the legal scaffolding that permits the creation of this vibrant commercial and noncommercial Free Software economy.

[5]This quotation is Copyright © 2002, Lawrence Lessig. It is licensed under the terms of `http://creativecommons.org/licenses/by/1.0/`the "Attribution License" version 1.0 or any later version as published by Creative Commons.

Chapter 2

Running Software and Verbatim Copying

This chapter begins the deep discussion of the details of the terms of GPL. In this chapter, we consider the first two sections: GPL §§0–2. These are the straightforward sections of the GPL that define the simplest rights that the user receives.

2.1 GPL §0: Freedom to Run

§0, the opening section of GPL, sets forth that the work is governed by copyright law. It specifically points out that it is the "copyright holder" who decides if a work is licensed under its terms and explains how the copyright holder might indicate this fact.

A bit more subtly, §0 makes an inference that copyright law is the only system under which it is governed. Specifically, it states:

> Activities other than copying, distribution and modification are not covered by this License; they are outside its scope.

In essence, the license governs *only* those activities, and all other activities are unrestricted, provided that no other agreements trump GPL (which they cannot; see Sections 6.3 and 6.4). This is very important, because the Free Software community heavily supports users' rights to "fair use" and "unregulated use" of copyrighted material. GPL asserts through this clause that it supports users' rights to fair and unregulated uses.

Fair use of copyrighted material is an established legal doctrine that permits certain activities. Discussion of the various types of fair use activity are beyond the scope of this tutorial. However, one important example of fair use is the right to quote a very few lines (less than seven or so) and reuse them as you would with or without licensing restrictions.

Fair use is a doctrine established by the courts or by statute. By contrast, unregulated uses are those that are not covered by the statue nor determined by a court to be covered, but are common and enjoyed by many users. An example of unregulated use is reading a printout of the program's source code like an instruction book for the purpose of learning how to be a better programmer.

Thus, the GPL protects users fair and unregulated use rights precisely by not attempting to cover them. Furthermore, the GPL ensures the freedom to run specifically by stating the following:

> "The act of running the Program is not restricted."

Thus, users are explicitly given the freedom to run by §0.

The bulk of §0 not yet discussed gives definitions for other terms used throughout. The only one worth discussing in detail is "work based on the Program." The reason this definition is particularly interesting is not for the definition itself, which is rather straightforward, but because it clears up a common misconception about the GPL.

The GPL is often mistakenly criticized because it fails to give a definition of "derivative work." In fact, it would be incorrect and problematic if the GPL attempted to define this. A copyright license, in fact, has no control over what may or may not be a derivative work. This matter is left up to copyright law, not the licenses that utilize it.

It is certainly true that copyright law as a whole does not propose clear and straightforward guidelines for what is and is not a derivative software work under copyright law. However, no copyright license — not even the GNU GPL — can be blamed for this. Legislators and court opinions must give us guidance to decide the border cases.

2.2 GPL §1: Verbatim Copying

GPL §1 covers the matter of redistributing the source code of a program exactly as it was received. This section is quite straightforward. However, there are a few details worth noting here.

The phrase "in any medium" is important. This, for example, gives the freedom to publish a book that is the printed copy of the program's source code. It also allows for changes in the medium of distribution. Some vendors may ship Free Software on a CD, but others may place it right on the hard drive of a pre-installed computer. Any such redistribution media is allowed.

Preservation of copyright notice and license notifications are mentioned specifically in §1. These are in some ways the most important part of the redistribution, which is why they are mentioned by name. The GPL always strives to make it abundantly clear to anyone who receives the software what its license is. The goal is to make sure users know their rights and freedoms under GPL, and to leave no reason that someone would be surprised the software she got was licensed under GPL. Thus throughout the GPL, there are specific references to the importance of notifying others down the distribution chain that they have rights under GPL.

Also mentioned by name is the warranty disclaimer. Most people today do not believe that software comes with any warranty. Notwithstanding the proposed state-level UCITA bills (which have never obtained widespread adoption), there are few or no implied warranties with software. However, just to be on the safe side, GPL clearly disclaims them, and the GPL requires redistributors to keep the disclaimer very visible. (See Sections 7.3 and 7.4 of this tutorial for more on GPL's warranty disclaimers.)

Note finally that §1 begins to set forth the important defense of commercial freedom. §1 clearly states that in the case of verbatim copies, one may make money. Redistributors are fully permitted to charge for the redistribution of copies of Free Software. In addition, they may provide the warranty protection that the GPL disclaims as an additional service for a fee. (See Section 9.2 for more discussion on making a profit from Free Software redistribution.)

Chapter 3

Derivative Works: Statute and Case Law

We digress for this chapter from our discussion of GPL's exact text to consider the matter of derivative works — a concept that we must understand fully before considering §§2–3 of GPL. GPL, and Free Software licensing in general, relies critically on the concept of "derivative work" since software that is "independent," (i.e., not "derivative") of Free Software need not abide by the terms of the applicable Free Software license. As much is required by §106 of the Copyright Act, 17 U.S.C. §106 (2002), and admitted by Free Software licenses, such as the GPL, which (as we have seen) states in §0 that "a 'work based on the Program' means either the Program or any derivative work under copyright law." It is being a derivative work of Free Software that triggers the necessity to comply with the terms of the Free Software license under which the original work is distributed. Therefore, one is left to ask, just what is a "derivative work"? The answer to that question differs depending on which court is being asked.

The analysis in this chapter sets forth the differing definitions of derivative work by the circuit courts. The broadest and most established definition of derivative work for software is the abstraction, filtration, and comparison test ("the AFC test") as created and developed by the Second Circuit. Some circuits, including the Ninth Circuit and the First Circuit, have either adopted narrower versions of the AFC test or have expressly rejected the AFC test in favor of a narrower standard. Further, several other circuits have yet to adopt any definition of derivative work for software.

As an introductory matter, it is important to note that literal copying of a significant portion of source code is not always sufficient to establish that a second work is a derivative work of an original program. Conversely, a second work can be a derivative work of an original program even though absolutely no copying of the literal source code of the original program has been made. This is the case because copyright protection does not always extend to all portions of a program's code, while, at the same time, it can extend beyond the literal code of a program to its non-literal aspects, such as its architecture, structure, sequence, organization, operational modules, and computer-user interface.

3.1 The Copyright Act

The copyright act is of little, if any, help in determining the definition of a derivative work of software. However, the applicable provisions do provide some, albeit quite cursory, guidance. Section 101 of the Copyright Act sets forth the following definitions:

> A "computer program" is a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result.
>
> A "derivative work" is a work based upon one or more preexisting works, such as a translation, musical arrangement, dramatization, fictionalization, motion picture version, sound recording, art reproduction, abridgment, condensation, or any other form in which a work may be recast, transformed, or adapted. A work consisting of editorial revisions, annotations, elaborations, or other modifications which, as a whole, represent an original work of authorship, is a "derivative work."

These are the only provisions in the Copyright Act relevant to the determination of what constitutes a derivative work of a computer program. Another provision of the Copyright Act that is also relevant to the definition of derivative work is §102(b), which reads as follows:

> In no case does copyright protection for an original work of authorship extend to any idea, procedure, process, system, method of operation, concept, principle, or discovery, regardless of the form in which it is described, explained, illustrated, or embodied in such work.

Therefore, before a court can ask whether one program is a derivative work of another program, it must be careful not to extend copyright protection to any ideas, procedures, processes, systems, methods of operation, concepts, principles, or discoveries contained in the original program. It is the implementation of this requirement to "strip out" unprotectable elements that serves as the most frequent issue over which courts disagree.

3.2 Abstraction, Filtration, Comparison Test

As mentioned above, the AFC test for determining whether a computer program is a derivative work of an earlier program was created by the Second Circuit and has since been adopted in the Fifth, Tenth, and Eleventh Circuits. Computer Associates Intl., Inc. v. Altai, Inc., 982 F.2d 693 (2nd Cir. 1992); Engineering Dynamics, Inc. v. Structural Software, Inc., 26 F.3d 1335 (5th Cir. 1994); Kepner-Tregoe, Inc. v. Leadership Software, Inc., 12 F.3d 527 (5th Cir. 1994); Gates Rubber Co. v. Bando Chem. Indust., Ltd., 9 F.3d 823 (10th Cir. 1993); Mitel, Inc. v. Iqtel, Inc., 124 F.3d 1366 (10th Cir. 1997); 5 Bateman v. Mnemonics, Inc., 79 F.3d 1532 (11th Cir. 1996); and, Mitek Holdings, Inc. v. Arce Engineering Co., Inc., 89 F.3d 1548 (11th Cir. 1996).

Under the AFC test, a court first abstracts from the original program its constituent structural parts. Then, the court filters from those structural parts all unprotectable portions, including incorporated ideas, expression that is necessarily incidental to those ideas, and elements that are taken from the public domain. Finally, the court compares any and all remaining kernels of creative expression to the structure of the second program to determine whether the software programs at issue are substantially similar so as to warrant a finding that one is the derivative work of the other.

Often, the courts that apply the AFC test will perform a quick initial comparison between the entirety of the two programs at issue in order to help determine whether one is a derivative work of the other. Such a holistic comparison, although not a substitute for the full application of the AFC test, sometimes reveals a pattern of copying that is not otherwise obvious from the application of the AFC test when, as discussed below, only certain components of the original program are compared to the second program. If such a pattern is revealed by the quick initial comparison, the court is more likely to conclude that the second work is indeed a derivative of the original.

3.2.1 Abstraction

The first step courts perform under the AFC test is separation of the work's ideas from its expression. In a process akin to reverse engineering, the courts dissect the original program to isolate each level of abstraction contained within it. Courts have stated that the abstractions step is particularly well suited for computer programs because it breaks down software in a way that mirrors the way it is typically created. However, the courts have also indicated that this step of the AFC test requires substantial guidance from experts, because it is extremely fact and situation specific.

By way of example, one set of abstraction levels is, in descending order of generality, as follows: the main purpose, system architecture, abstract data types, algorithms and data structures, source code, and object code. As this set of abstraction levels shows, during the abstraction step of the AFC test, the literal elements of the computer program, namely the source and object code, are defined as particular levels of abstraction. Further, the source and object code elements of a program are not the only elements capable of forming the basis for a finding that a second work is a derivative of the program. In some cases, in order to avoid a lengthy factual inquiry by the court, the owner of the copyright in the original work will submit its own list of what it believes to be the protected elements of the original program. In those situations, the court will forgo performing its own abstraction, and proceed to the second step of the AFC test.

3.2.2 Filtration

The most difficult and controversial part of the AFC test is the second step, which entails the filtration of protectable expression contained in the original program from any unprotectable elements nestled therein. In determining which elements of a program are unprotectable, courts employ a myriad of rules and procedures to sift from a program all the portions that are not eligible for copyright protection.

First, as set forth in §102(b) of the Copyright Act, any and all ideas embodied in the program are to be denied copyright protection. However, implementing this rule is not as easy as it first appears. The courts readily recognize the intrinsic difficulty in distinguishing between ideas and expression and that, given the varying nature of computer programs, doing so will be done on an ad hoc basis. The first step of the AFC test, the abstraction, exists precisely to assist in this endeavor by helping the court separate out all the individual elements of the program so that they can be independently analyzed for their expressive nature.

A second rule applied by the courts in performing the filtration step of the AFC test is the doctrine of merger, which denies copyright protection to expression necessarily incidental to the idea being expressed. The reasoning behind this doctrine is that when there is only one way to express an idea, the idea and the expression merge, meaning that the expression cannot receive copyright protection due to the bar on copyright protection extending to ideas. In applying this doctrine, a court will ask whether the program's use of particular code or structure is necessary for the efficient implementation of a certain function or process. If so, then that particular code or structure is not protected by copyright and, as a result, it is filtered away from the remaining protectable expression.

A third rule applied by the courts in performing the filtration step of the AFC test is the doctrine of scenes a faire, which denies copyright protection to elements of a computer program that are dictated by external factors. Such external factors can include:

- The mechanical specifications of the computer on which a particular program is intended to operate
- Compatibility requirements of other programs with which a program is designed to operate in conjunction
- Computer manufacturers' design standards
- Demands of the industry being serviced, and

  widely accepted programming practices within the computer industry

Any code or structure of a program that was shaped predominantly in response to these factors is filtered out and not protected by copyright. Lastly, elements of a computer program are also to be filtered out if they were taken from the public domain or fail to have sufficient originality to merit copyright protection.

Portions of the source or object code of a computer program are rarely filtered out as unprotectable elements. However, some distinct parts of source and object code have been found unprotectable. For example, constant s, the invariable integers comprising part of formulas used to perform calculations in a program, are unprotectable. Further, although common errors found in two programs can provide strong evidence of copying, they are not afforded any copyright protection over and above the protection given to the expression containing them.

3.2.3 Comparison

The third and final step of the AFC test entails a comparison of the original program's remaining protectable expression to a second program. The issue will be whether any of the protected expression is copied in the second program and, if so, what relative importance the copied portion has with respect to the original program overall. The ultimate inquiry is whether there is "substantial" similarity between the protected elements of the original program and the potentially derivative work. The courts admit that this process is primarily qualitative rather than quantitative and is performed on a case-by-case basis. In essence, the comparison is an ad hoc determination of whether the protectable elements of the original program that are contained in the second work are significant or important parts of the original program. If so, then the second work is a derivative work of the first. If, however, the amount of protectable elements copied in the second work are so small as to be de minimis, then the second work is not a derivative work of the original.

3.3 Analytic Dissection Test

The Ninth Circuit has adopted the analytic dissection test to determine whether one program is a derivative work of another. Apple Computer, Inc. v. Microsoft Corp., 35 F.3d 1435 (9th Cir. 1994). The analytic dissection test first considers whether there are substantial similarities in both the ideas and expressions of the two works at issue. Once the similar features are identified, analytic dissection is used to determine whether any of those similar features are protected by copyright. This step is the same as the filtration step in the AFC test. After identifying the copyrightable similar features of the works, the court then decides whether those features are entitled to "broad" or "thin" protection. "Thin" protection is given to non-copyrightable facts or ideas that are combined in a way that affords copyright protection only from their alignment and presentation, while "broad" protection is given to copyrightable expression itself. Depending on the degree of protection afforded, the court then sets the appropriate standard for a subjective comparison of the works to determine whether, as a whole, they are sufficiently similar to support a finding that one is a derivative work of the other. "Thin" protection requires the second work be virtually identical in order to be held a derivative work of an original, while "broad" protection requires only a "substantial similarity."

3.4 No Protection for "Methods of Operation"

The First Circuit expressly rejected the AFC test and, instead, takes a much narrower view of the meaning of derivative work for software. The First Circuit holds that "method of operation," as used in §102(b) of the Copyright Act, refers to the means by which users operate computers. Lotus Development Corp. v. Borland Intl., Inc., 49 F.3d 807 (1st Cir. 1995). More specifically, the court held that a menu command hierarchy for a computer program was uncopyrightable because it did not merely explain and present the programs functional capabilities to the user, but also served as a method by which the program was operated and controlled. As a result, under the First Circuits test, literal copying of a menu command hierarchy, or any other "method of operation," cannot form the basis for a determination that one work is a derivative of another. It is also reasonable to expect that the First Circuit will read the unprotectable elements set forth in §102(b) broadly, and, as such, promulgate a definition of derivative work that is much narrower than that which exists under the AFC test.

3.5 No Test Yet Adopted

Several circuits, most notably the Fourth and Seventh, have yet to declare their definition of derivative work and whether or not the AFC, Analytic Dissection, or some other test best fits their interpretation of copyright law. Therefore, uncertainty exists with respect to determining the extent to which a software program is a derivative work of another in those circuits. However, one may presume that they would give deference to the AFC test since it is by far the majority rule amongst those circuits that have a standard for defining a software derivative work.

3.6 Cases Applying Software Derivative Work Analysis

In the preeminent case regarding the definition of a derivative work for software, Computer Associates v. Altai, the plaintiff alleged that its program, Adapter, which was used to handle the differences in operating system calls and services, was infringed by the defendant's competitive program, Oscar. About 30% of Oscar was literally the same code as that in Adapter. After the suit began, the defendant rewrote those portions of Oscar that contained

Adapter code in order to produce a new version of Oscar that was functionally competitive with Adapter, without have any literal copies of its code. Feeling slighted still, the plaintiff alleged that even the second version of Oscar, despite having no literally copied code, also infringed its copyrights. In addressing that question, the Second Circuit promulgated the AFC test.

In abstracting the various levels of the program, the court noted a similarity between the two programs' parameter lists and macros. However, following the filtration step of the AFC test, only a handful of the lists and macros were protectable under copyright law because they were either in the public domain or required by functional demands on the program. With respect to the handful of parameter lists and macros that did qualify for copyright protection, after performing the comparison step of the AFC test, it was reasonable for the district court to conclude that they did not warrant a finding of infringement given their relatively minor contribution to the program as a whole. Likewise, the similarity between the organizational charts of the two programs was not substantial enough to support a finding of infringement because they were too simple and obvious to contain any original expression.

Perhaps not surprisingly, there have been few cases involving a highly detailed software derivative work analysis. Most often, cases involve clearer basis for decision, including frequent bad faith on the part of the defendant or overaggressiveness on the part of the plaintiff. However, no cases involving Free Software licensing have ever gone to court. As Free Software becomes an ever-increasingly important part of the economy, it remains to be seen if battle lines will be drawn over whether particular programs infringe the rights of Free Software developers or whether the entire community, including industry, adopts norms avoiding such risk.

Chapter 4

Modified Source and Binary Distribution

In this chapter, we discuss the two core sections that define the rights and obligations for those who modify, improve, and/or redistribute GPL'd software. These sections, §§2–3, define the central core rights and requirements of GPL.

4.1 GPL §2: Share and Share Alike

For many, this is where the "magic" happens that defends software freedom along the distribution chain. §2 is the only place in the GPL that governs the modification controls of copyright law. If someone modifies a GPL'd program, she is bound in the making those changes by §2. The goal here is to ensure that the body of GPL'd software, as it continues and develops, remains Free as in freedom.

To achieve that goal, §2 first sets forth that the rights of redistribution of modified versions are the same as those for verbatim copying, as presented in §1. Therefore, the details of charging, keeping copyright notices intact, and other §1 provisions are in tact here as well. However, there are three additional requirements.

The first (§2(a)) requires that modified files carry "prominent notices" explaining what changes were made and the date of such changes. The goal here is not to put forward some specific way of marking changes nor controlling the process of how changes get made. Primarily, §2(a) seeks to ensure that those receiving modified versions know the history of changes to the software. For some users, it is important to know that they are using the standard version of program, because while there are many advantages to using a fork, there are a few disadvantages. Users should be informed about the historical context of the software version they use, so that they can make proper support choices. Finally, §2(a) serves an academic purpose — ensuring that future developers can use a diachronic approach to understand the software.

The second requirement (§2(b)) contains the four short lines that embody the legal details of "share and share alike." These 46 words are considered by some to be the most worthy of careful scrutiny because §2(b) can be a source of great confusion when not properly understood.

In considering §2(b), first note the qualifier: it only applies to derivative works that "you distribute or publish." Despite years of education efforts by FSF on this matter, many still believe that modifiers of GPL'd software are required by the license to publish or otherwise share their changes. On the contrary, §2(b) **does not apply if** the changes are never distributed. Indeed, the freedom to make private, personal, unshared changes to software for personal use only should be protected and defended.[1]

Next, we again encounter the same matter that appears in §0, in the following text:

> "...that in whole or part contains or is derived from the Program or any part thereof."

Again, the GPL relies here on what the copyright law says is a derivative work. If, under copyright law, the modified version "contains or is derived from" the GPL'd software, then the requirements of §2(b) apply. The GPL invokes its control as a copyright license over the modification of the work in combination with its control over distribution of the work.

The final clause of §2(b) describes what the licensee must do if she is distributing or publishing a work that is deemed a derivative work under copyright law — namely, the following:

> [The work must] be licensed as a whole at no charge to all third parties under the terms of this License.

That is probably the most tightly-packed phrase in all of the GPL. Consider each subpart carefully.

[1]FSF does maintain that there is an **ethical** obligation to redistribute changes that are generally useful, and often encourages companies and individuals to do so. However, there is a clear distinction between what one **ought** to do and what one **must** do.

The work "as a whole" is what is to be licensed. This is an important point that §2 spends an entire paragraph explaining; thus this phrase is worthy of a lengthy discussion here. As a programmer modifies a software program, she generates new copyrighted material — fixing expressions of ideas into the tangible medium of electronic file storage. That programmer is indeed the copyright holder of those new changes. However, those changes are part and parcel to the original work distributed to the programmer under GPL. Thus, the license of the original work affects the license of the new whole derivative work.

It is certainly possible to take an existing independent work (called $\mathcal{I}$) and combine it with a GPL'd program (called $\mathcal{G}$). The license of $\mathcal{I}$, when it is distributed as a separate and independent work, remains the prerogative of the copyright holder of $\mathcal{I}$. However, when $\mathcal{I}$ is combined with $\mathcal{G}$, it produces a new work that is the combination of the two (called $\mathcal{G}$+$\mathcal{I}$). The copyright of this combined work, $\mathcal{G}$+$\mathcal{I}$, is held by the original copyright holder of each of the two works.

In this case, §2 lays out the terms by which $\mathcal{G}$+$\mathcal{I}$ may be distributed and copied. By default, under copyright law, the copyright holder of $\mathcal{I}$ would not have been permitted to distribute $\mathcal{G}$+$\mathcal{I}$; copyright law forbids it without the expressed permission of the copyright holder of $\mathcal{G}$. (Imagine, for a moment, if $\mathcal{G}$ were a Microsoft product — would they give you permission to create and distribute $\mathcal{G}$+$\mathcal{I}$ without paying them a hefty sum?) The license of $\mathcal{G}$, the GPL, sets forth ahead of time options for the copyright holder of $\mathcal{I}$ who may want to create and distribute $\mathcal{G}$+$\mathcal{I}$. This pregranted permission to create and distribute derivative works, provided the terms of GPL are upheld, goes far above and beyond the permissions that one would get with a typical work not covered by a copyleft license. Thus, to say that this restriction is any way unreasonable is simply ludicrous.

The next phrase of note in §2(b) is "licensed...at no charge." This is a source of great confusion to many. Not a month goes by that FSF does not receive an email that claims to point out "a contradiction in GPL" because §2 says that redistributors cannot charge for modified versions of GPL'd software, but §1 says that they can. The "at no charge" does not prohibit redistributors from charging when performing the acts governed by copyright law,[2] but rather that they cannot charge a fee for the *license itself.* In other words, redistributors of (modified and unmodified) GPL'd works may charge any amount they choose for performing the modifications on

[2]Recall that you could by default charge for any acts not governed by copyright law, because the license controls are confined by copyright.

contract or the act of transferring the copy to the customer, but they may not charge a separate licensing fee for the software.

§2(b) further states that the software must "be licensed...to all third parties." This too has led to some confusions, and feeds the misconception mentioned earlier — that all modified versions must made available to the public at large. However, the text here does not say that. Instead, it says that the licensing under terms of the GPL must extend to anyone who might, through the distribution chain, receive a copy of the software. Distribution to all third parties is not mandated here, but §2(b) does require redistributors to license the derivative works in a way that extends to all third parties who may ultimately receive a copy of the software.

In summary, §2(b) says what terms under which the third parties must receive this no-charge license. Namely, they receive it "under the terms of this License," the GPL. When an entity *chooses* to redistribute a derivative work of GPL'd software, the license of that whole derivative work must be GPL and only GPL. In this manner, §2(b) dovetails nicely with §6 (as discussed in Section 6.3 of this tutorial).

The final paragraph of §2 is worth special mention. It is possible and quite common to aggregate various software programs together on one distribution medium. Computer manufacturers do this when they ship a pre-installed hard drive, and GNU/Linux distribution vendors do this to give a one-stop CD or URL for a complete operating system with necessary applications. The GPL very clearly permits such "mere aggregation" with programs under any license. Despite what you hear from its critics, the GPL is nothing like a virus, not only because the GPL is good for you and a virus is bad for you, but also because simple contact with a GPL'd code-base does not impact the license of other programs. Actual effort must be expended by a programmer to cause a work to fall under the terms of the GPL. Redistributors are always welcome to simply ship GPL'd software alongside proprietary software or other unrelated Free Software, as long as the terms of GPL are adhered to for those packages that are truly GPL'd.

4.2 GPL §3: Producing Binaries

Software is a strange beast when compared to other copyrightable works. It is currently impossible to make a film or a book that can be truly obscured. Ultimately, the full text of a novel, even one written by William Faulkner, must presented to the reader as words in some human-readable language so that they can enjoy the work. A film, even one directed by David Lynch, must be perceptible by human eyes and ears to have any value.

Software is not so. While the source code, the human-readable representation of software is of keen interest to programmers, users and programmers alike cannot make the proper use of software in that human-readable form. Binary code — the ones and zeros that the computer can understand — must be predicable and attainable for the software to be fully useful. Without the binaries, be they in object or executable form, the software serves only the didactic purposes of computer science.

Under copyright law, binary representations of the software are simply derivative works of the source code. Applying a systematic process (i.e., "compilation") to a work of source code yields binary code. The binary code is now a new work of expression fixed in the tangible medium of electronic file storage.

Therefore, for GPL'd software to be useful, the GPL, since it governs the rules for creation of derivative works, must grant permission for the generation of binaries. Furthermore, notwithstanding the relative popularity of source-based GNU/Linux distributions like Gentoo, users find it extremely convenient to receive distribution of binary software. Such distribution is the redistribution of derivative works of the software's source code. §3 addresses the matter of creation and distribution of binary versions.

Under §3, binary versions may be created and distributed under the terms of §§1–2, so all the material previously discussed applies here. However, §3 must go a bit further. Access to the software's source code is an incontestable prerequisite for the exercise of the fundamental freedoms to modify and improve the software. Making even the most trivial changes to a software program at the binary level is effectively impossible. §3 must ensure that the binaries are never distributed without the source code, so that these freedoms are passed through the distribution chain.

§3 permits distribution of binaries, and then offers three options for distribution of source code along with binaries. The most common and the least complicated is the option given under §3(a).

§3(a) offers the option to directly accompany the source code alongside the distribution of the binaries. This is by far the most convenient option for most distributors, because it means that the source-code provision obligations are fully completed at the time of binary distribution (more on that later).

Under §3(a), the source code provided must be the "corresponding source code." Here "corresponding" primarily means that the source code provided must be that code used to produce the binaries being distributed. That source code must also be "complete." A later paragraph of §3 explains in detail what is meant by "complete." In essence, it is all the material that a programmer of average skill would need to actually use the source code to produce the binaries she has received. Complete source is required so that, if the licensee chooses, she should be able to exercise her freedoms to modify and redistribute changes. Without the complete source, it would not be possible to make changes that were actually directly derived from the version received.

Furthermore, §3 is defending against a tactic that has in fact been seen in FSF's GPL enforcement. Under GPL, if you pay a high price for a copy of GPL'd binaries (which comes with corresponding source, of course), you have the freedom to redistribute that work at any fee you choose, or not at all. Sometimes, companies attempt a GPL-violating cozenage whereby they produce very specialized binaries (perhaps for an obscure architecture). They then give source code that does correspond, but withhold the "incantations" and build plans they used to make that source compile into the specialized binaries. Therefore, §3 requires that the source code include "meta-material" like scripts, interface definitions, and other material that is used to "control compilation and installation" of the binaries. In this manner, those further down the distribution chain are assured that they have the unabated freedom to build their own derivative works from the sources provided.

FSF (as authors of GPL) realizes that software distribution comes in many forms. Embedded manufacturers, for example, have the freedom to put GPL'd software into their PDAs with very tight memory and space constraints. In such cases, putting the source right alongside the binaries on the machine itself might not be an option. While it is recommended that this be the default way that people comply with GPL, the GPL does provide options when such distribution is infeasible.

§3, therefore, allows source code to be provided on any physical "medium customarily used for software interchange." By design, this phrase covers a broad spectrum. At best, FSF can viably release a new GPL every ten years or so. Thus, phrases like this must be adaptive to changes in the technology. When GPL version 2 was first published in June 1991, distribution on magnetic tape was still common, and CD was relatively new. Today, CD is the default, and for larger systems DVD-R is gaining adoption. This language must adapt with changing technology.

Meanwhile, the binding created by the word "customarily" is key. Many incorrectly believe that distributing binary on CD and source on the Internet is acceptable. In the corporate world, it is indeed customary to simply download CDs worth of data over a T1 or email large file attachments.

However, even today in the USA, many computer users with CD-ROM drives are not connected to the Internet, and most people connected to the Internet are connected via a 56K dial-up connection. Downloading CDs full of data is not customary for them in the least. In some cities in Africa, computers are becoming more common, but Internet connectivity is still available only at a few centralized locations. Thus, the "customs" here must be normalized for a worldwide userbase. Simply providing source on the Internet — while it is a kind, friendly and useful thing to do — is not usually sufficient.

Note, however, a major exception to this rule, given by the last paragraph of §3. *If* distribution of the binary files is made only on the Internet (i.e., "from a designated place"), *then* simply providing the source code right alongside the binaries in the same place is sufficient to comply with §3.

As is shown above, Under §3(a), embedded manufacturers can put the binaries on the device and ship the source code along on a CD. However, sometimes this turns out to be too costly. Including a CD with every device could prove too costly, and may practically (although not legally) prohibit using GPL'd software. For this situation and others like it, §3(b) is available.

§3(b) allows a distributor of binaries to instead provide a written offer for source code alongside those binaries. This is useful in two specific ways. First, it may turn out that most users do not request the source, and thus the cost of producing the CDs is saved — a financial and environmental windfall. In addition, along with a §3(b) compliant offer for source, a binary distributor might choose to *also* give a URL for source code. Many who would otherwise need a CD with source might turn out to have those coveted high bandwidth connections, and are able to download the source instead — again yielding environmental and financial windfalls.

However, note that regardless of how many users prefer to get the source online, §3(b) does place lasting long-term obligations on the binary distributor. The binary distributor must be prepared to honor that offer for source for three years and ship it out (just as they would have had to do under §3(a)) at a moment's notice when they receive such a request. There is real organizational cost here: support engineers must be trained how to route source requests, and source CD images for every release version for the last three years must be kept on hand to burn such CDs quickly. The requests might not even come from actual customers; the offer for source must be valid for "any third party."

That phrase is another place where some get confused — thinking again that full public distribution of source is required. The offer for source must be valid for "any third party" because of the freedoms of redistribution granted by §§1–2. A company may ship a binary image and an offer for source to only one customer. However, under GPL, that customer has the right to redistribute that software to the world if she likes. When she does, that customer has an obligation to make sure that those who receive the software from her can exercise their freedoms under GPL — including the freedom to modify, rebuild, and redistribute the source code.

§3(c) is created to save her some trouble, because by itself §3(b) would unfairly favor large companies. §3(b) allows the separation of the binary software from the key tool that people can use to exercise their freedom. The GPL permits this separation because it is good for redistributors, and those users who turn out not to need the source. However, to ensure equal rights for all software users, anyone along the distribution chain must have the right to get the source and exercise those freedoms that require it.

Meanwhile, §3(b)'s compromise primarily benefits companies who distribute binary software commercially. Without §3(c), that benefit would be at the detriment of the companies' customers; the burden of source code provision would be unfairly shifted to the companies' customers. A customer, who had received binaries with a §3(b)-compliant offer, would be required under GPL (sans §3(c)) to acquire the source, merely to give a copy of the software to a friend who needed it. §3(c) reshifts this burden to entity who benefits from §3(b).

§3(c) allows those who undertake *noncommercial* distribution to simply pass along a §3(b)-compliant source code offer. The customer who wishes to give a copy to her friend can now do so without provisioning the source, as long as she gives that offer to her friend. By contrast, if she wanted to go into business for herself selling CDs of that software, she would have to acquire the source and either comply via §3(a), or write her own §3(b)-compliant source offer.

This process is precisely the reason why a §3(b) source offer must be valid for all third parties. At the time the offer is made, there is no way of knowing who might end up noncommercially receiving a copy of the software. Companies who choose to comply via §3(b) must thus be prepared to honor all incoming source code requests. For this and the many other additional necessary complications under §§3(b–c), it is only rarely a better option than complying via §3(a).

Chapter 5

The Implied Patent Grant in GPL

We digress again briefly from our section-by-section consideration of GPL to consider the interaction between the terms of GPL and patent law. The GPL, despite being silent with respect to patents, actually confers on its licensees more rights to a licensor's patents than those licenses that purport to address the issue. This is the case because patent law, under the doctrine of implied license, gives to each distributee of a patented article a license from the distributor to practice any patent claims owned or held by the distributor that cover the distributed article. The implied license also extends to any patent claims owned or held by the distributor that cover "reasonably contemplated uses" of the patented article. To quote the Federal Circuit Court of Appeals, the highest court for patent cases other than the Supreme Court:

> Generally, when a seller sells a product without restriction, it in effect promises the purchaser that in exchange for the price paid, it will not interfere with the purchaser's full enjoyment of the product purchased. The buyer has an implied license under any patents of the seller that dominate the product or any uses of the product to which the parties might reasonably contemplate the product will be put.

Hewlett-Packard Co. v. Repeat-O-Type Stencil Mfg. Corp., Inc., 123 F.3d 1445 (Fed. Cir. 1997).

Of course, Free Software is licensed, not sold, and there are indeed restrictions placed on the licensee, but those differences are not likely to prevent the application of the implied license doctrine to Free Software, because software licensed under the GPL grants the licensee the right to make, use, and sell the software, each of which are exclusive rights of a patent holder. Therefore, although the GPL does not expressly grant the licensee the right to do those things under any patents the licensor may have that cover the software or its reasonably contemplated uses, by licensing the software under the GPL, the distributor impliedly licenses those patents to the GPL licensee with respect to the GPL licensed software.

An interesting issue regarding this implied patent license of GPL'd software is what would be considered "uses of the [software] to which the parties might reasonably contemplate the product will be put." A clever advocate may argue that the implied license granted by GPL is larger in scope than the express license in other Free Software licenses with express patent grants, in that, the patent license clause of many of those licenses are specifically limited to the patent claims covered by the code as licensed by the patentee.

To the contrary, GPL's implied patent license grants the GPL licensee a patent license to do much more than just that because the GPL licensee, under the doctrine of implied patent license, is free to practice any patent claims held by the licensor that cover "reasonably contemplated uses" of the GPL'd code, which may very well include creation and distribution of derivative works since the GPL's terms, under which the patented code is distributed, expressly permits such activity.

Further supporting this result is the Federal Circuit's pronouncement that the recipient of a patented article has, not only an implied license to make, use, and sell the article, but also an implied patent license to repair the article to enable it to function properly, Bottom Line Mgmt., Inc. v. Pan Man, Inc., 228 F.3d 1352 (Fed. Cir. 2000). Additionally, the Federal Circuit extended that rule to include any future recipients of the patented article, not just the direct recipient from the distributor. This theory comports well with the idea of Free Software, whereby software is distributed amongst many entities within the community for the purpose of constant evolution and improvement. In this way, the law of implied patent license used by the GPL ensures that the community mutually benefits from the licensing of patents to any single community member.

Note that simply because GPL'd software has an implied patent license does not mean that any patents held by a distributor of GPL'd code become worthless. To the contrary, the patents are still valid and enforceable against either:

(a) any software other than that licensed under the GPL by the patent holder, and

(b) any party that does not comply with the GPL with respect to the licensed software.

For example, if Company $\mathcal{A}$ has a patent on advanced Web browsing, but also licenses a Web browsing software program under the GPL, then it cannot assert the patent against any party that takes a license to its program under the GPL. However, if a party uses that program without complying with the GPL, then Company $\mathcal{A}$ can assert, not just copyright infringement claims against the non-GPL-compliant party, but also infringement of the patent, because the implied patent license only extends to use of the software in accordance with the GPL. Further, if Company $\mathcal{B}$ distributes a competitive advanced Web browsing program, Company $\mathcal{A}$ is free to assert its patent against any user or distributor of that product. It is irrelevant whether Company $\mathcal{B}$'s program is distributed under the GPL, as Company $\mathcal{B}$ can not grant implied licenses to Company $\mathcal{A}$'s patent.

This result also reassures companies that they need not fear losing their proprietary value in patents to competitors through the GPL implied patent license, as only those competitors who adopt and comply with the GPL's terms can benefit from the implied patent license. To continue the example above, Company $\mathcal{B}$ does not receive a free ride on Company $\mathcal{A}$'s patent, as Company $\mathcal{B}$ has not licensed-in and then redistributed Company A's advanced Web browser under the GPL. If Company $\mathcal{B}$ does do that, however, Company $\mathcal{A}$ still has not lost competitive advantage against Company $\mathcal{B}$, as Company $\mathcal{B}$ must then, when it re-distributes Company $\mathcal{A}$'s program, grant an implied license to any of its patents that cover the program. Further, if Company $\mathcal{B}$ relicenses an improved version of Company A's program, it must do so under the GPL, meaning that any patents it holds that cover the improved version are impliedly licensed to any licensee. As such, the only way Company $\mathcal{B}$ can benefit from Company $\mathcal{A}$'s implied patent license, is if it, itself, distributes Company $\mathcal{A}$'s software program and grants an implied patent license to any of its patents that cover that program.

Chapter 6

Defending Freedom on Many Fronts

Chapters 2 and 4 presented the core freedom-defending provisions of GPL, which are in §§0–3. §§4–7 of the GPL are designed to ensure that §§0–3 are not infringed, are enforceable, are kept to the confines of copyright law, and are not trumped by other copyright agreements or components of other entirely separate legal systems. In short, while §§0–3 are the parts of the license that defend the freedoms of users and programmers, §§4–7 are the parts of the license that keep the playing field clear so that §§0–3 can do their jobs.

6.1 GPL §4: Termination on Violation

§4 is GPL's termination clause. Upon first examination, it seems strange that a license with the goal of defending users' and programmers' freedoms for perpetuity in an irrevocable way would have such a clause. However, upon further examination, the difference between irrevocability and this termination clause becomes clear.

The GPL is irrevocable in the sense that once a copyright holder grants rights for someone to copy, modify and redistribute the software under terms of the GPL, they cannot later revoke that grant. Since the GPL has no provision allowing the copyright holder to take such a prerogative, the license is granted as long as the copyright remains in effect.[1] The copyright holder

[1]In the USA, due to unfortunate legislation, the length of copyright is nearly perpetual, even though the Constitution forbids perpetual copyright.

has the right to relicense the same work under different licenses (see Section 9.2 of this tutorial), or to stop distributing the GPL'd version (assuming §3(b) was never used), but she may not revoke the rights under GPL already granted.

In fact, when an entity looses their right to copy, modify and distribute GPL'd software, it is because of their *own actions*, not that of the copyright holder. The copyright holder does not decided when §4 termination occurs (if ever), the actions of the licensee does.

Under copyright law, the GPL has granted various rights and freedoms to the licensee to perform specific types of copying, modification, and redistribution. By default, all other types of copying, modification, and redistribution are prohibited. §4 says that if you undertake any of those other types (e.g., redistributing binary-only in violation of §3), then all rights under the license — even those otherwise permitted for those who have not violated — terminate automatically.

§4 gives GPL teeth. If licensees fail to adhere to the license, then they are stuck. They must completely cease and desist from all copying, modification and distribution of that GPL'd software.

At that point, violating licensees must gain the forgiveness of the copyright holder to have their rights restored. Alternatively, they could negotiate another agreement, separate from GPL, with the copyright holder. Both are common practice.

At FSF, it is part of the mission to spread software freedom. When FSF enforces GPL, the goal is to bring the violator back into compliance as quickly as possible, and redress the damage caused by the violation. That is FSF's steadfast position in a violation negotiation — comply with the license and respect freedom.

However, other entities who do not share the full ethos of software freedom as institutionalized by FSF pursue GPL violations differently. MySQL AB, a company that produces the GPL'd MySQL database, upon discovering GPL violations typically negotiates a proprietary software license separately for a fee. While this practice is not one that FSF would ever consider undertaking or even endorsing, it is a legal way for copyright holders to proceed.

6.2 GPL §5: Acceptance, Copyright Style

§5 brings us to perhaps the most fundamental misconception and common confusion about GPL. Because of the prevalence of proprietary software, most users, programmers, and lawyers alike tend to be more familiar with EULAs. EULAs are believed by their authors to be contracts, requiring formal agreement between the licensee and the software distributor to be valid. This has led to mechanisms like “shrink-wrap” and “click-wrap” as mechanisms to perform acceptance ceremonies with EULAs.

The GPL does not need contract law to “transfer rights.” No rights are transfered between parties. By contrast, the GPL is a permission slip to undertake activities that would otherwise have been prohibited by copyright law. As such, it needs no acceptance ceremony; the licensee is not even required to accept the license.

However, without the GPL, the activities of copying, modifying and distributing the software would have otherwise been prohibited. So, the GPL says that you only accepted the license by undertaking activities that you would have otherwise been prohibited without your license under GPL. This is a certainly subtle point, and requires a mindset quite different from the contractual approach taken by EULA authors.

An interesting side benefit to §5 is that the bulk of users of Free Software are not required to accept the license. Undertaking fair and unregulated use of the work, for example, does not bind you to the GPL, since you are not engaging in activity that is otherwise controlled by copyright law. Only when you engage in those activities that might have an impact on the freedom of others does license acceptance occur, and the terms begin to bind you to fair and equitable sharing of the software. In other words, the GPL only kicks in when it needs to for the sake of freedom.

6.3 GPL §6: GPL, My One and Only

A point that was glossed over in Section 6.1’s discussion of §4 was the irrevocable nature of the GPL. The GPL is indeed irrevocable, and it is made so formally by §6.

The first sentence in §6 ensures that as software propagates down the distribution chain, that each licensor can pass along the license to each new licensee. Under §6, the act of distributing automatically grants a license from the original licensor to the next recipient. This creates a chain of grants that ensure that everyone in the distribution has rights under the GPL. In a mathematical sense, this bounds the bottom — making sure that future licensees get no fewer rights than the licensee before.

The second sentence of §6 does the opposite; it bounds from the top. It prohibits any licensor along the distribution chain from placing additional restrictions on the user. In other words, no additional requirements may trump the rights and freedoms given by GPL.

The final sentence of §6 makes it abundantly clear that no individual entity in the distribution chain is responsible for the compliance of any other. This is particularly important for noncommercial users who have passed along a source offer under §3(c), as they cannot be assured that the issuer of the offer will honor their §3 obligations.

In short, §6 says that your license for the software is your one and only copyright license allowing you to copy, modify and distribute the software.

6.4 GPL §7: "Give Software Liberty or Give It Death!"

In essence, §7 is a verbosely worded way of saying for non-copyright systems what §6 says for copyright. If there exists any reason that a distributor knows of that would prohibit later licensees from exercising their full rights under GPL, then distribution is prohibited.

Originally, this was designed as the title of this section suggests — as a last ditch effort to make sure that freedom was upheld. However, in modern times, it has come to give much more. Now that the body of GPL'd software is so large, patent holders who would want to be distributors of GPL'd software have a tough choice. They must choose between avoiding distribution of GPL'd software that exercises the teachings of their patents, or grant a royalty-free, irrevocable, non-exclusive license to those patents. Many companies, including IBM, the largest patent holder in the world, have chosen the latter.

Thus, §7 rarely gives software death by stopping its distribution. Instead, it is inspiring patent holders to share their patents in the same freedom-defending way that they share their copyrighted works.

6.5 GPL §8: Excluding Unfreedonia

§8 is rarely used by copyright holders. Its intention is that if a particular country, say Unfreedonia, grants particular patents or allows copyrighted interfaces (no country to our knowledge even permits those yet), that the GPL'd software can continue in free and unabated distribution in the countries where such controls do not exist.

It is a partial "out" from §7. Without §8, if a copyright holder knew of a patent in a particular country licensed in a GPL-incompatible way, then she could not distribute under GPL, because the work could legitimately end up in the hands of citizens of Unfreedonia.

It is an inevitable but sad reality that some countries are freer than others. §8 exists to permit distribution in those countries that are free without otherwise negating parts of the license.

Chapter 7

Odds, Ends, and Absolutely No Warranty

§0–7 constitute the freedom-defending terms of the GPL. The remainder of the GPL handles administrivia and issues concerning warranties and liability.

7.1 GPL §9: FSF as Stewards of GPL

FSF reserves the exclusive right to publish future versions of the GPL; §9 expresses this. While the stewardship of the copyrights on the body of GPL'd software around the world is shared among thousands of individuals and organizations, the license itself needs a single steward. Forking of the code is often regrettable but basically innocuous. Forking of licensing is disastrous.

FSF has only released two versions of GPL — in 1989 and 1991. GPL version 3 is under current internal drafting. FSF's plan is to have a long and engaging comment period. The goal of GPL is to defend freedom, and a gigantic community depends on that freedom now. FSF hopes to take all stakeholders' opinions under advisement.

7.2 GPL §10: Relicensing Permitted

§10 reminds the licensee of what is already implied by the nature of copyright law. Namely, the copyright holder of a particular software program has the prerogative to grant alternative agreements under separate copyright licenses.

7.3 GPL §11: No Warranty

All warranty disclaimer language tends to be shouted in all capital letters. Apparently, there was once a case where the disclaimer language of an agreement was negated because it was not "conspicuous" to one of the parties. Therefore, to make such language "conspicuous," people started placing it in bold or capitalizing the entire text. It now seems to be voodoo tradition of warranty disclaimer writing.

Some have argued the GPL is unenforceable in some jurisdictions because its disclaimer of warranties is impermissibly broad. However, §11 contains a jurisdictional savings provision, which states that it is to be interpreted only as broadly as allowed by applicable law. Such a provision ensures that both it, and the entire GPL, is enforceable in any jurisdiction, regardless of any particular law regarding the permissibility of certain warranty disclaimers.

Finally, one important point to remember when reading §11 is that §1 permits the sale of warranty as an additional service, which §11 affirms.

7.4 GPL, §12: Limitation of Liability

There are many types of warranties, and in some jurisdictions some of them cannot be disclaimed. Therefore, usually agreements will have both a warranty disclaimer and a limitation of liability, as we have in §12. §11 thus gets rid of all implied warranties that can legally be disavowed. §12, in turn, limits the liability of the actor for any warranties that cannot legally be disclaimed in a particular jurisdiction.

Again, some have argued the GPL is unenforceable in some jurisdictions because its limitation of liability is impermissibly broad. However, §12, just like its sister, §11, contains a jurisdictional savings provision, which states that it is to be interpreted only as broadly as allowed by applicable law. As stated above, such a provision ensures that both §12, and the entire GPL, is enforceable in any jurisdiction, regardless of any particular law regarding the permissibility of limiting liability.

So end the terms and conditions of the GNU General Public License.

Chapter 8

The Lesser GPL

As we have seen in our consideration of the GPL, its text is specifically designed to cover all possible derivative works under copyright law. Our goal in designing GPL was to make sure that any derivative work of GPL'd software was itself released under GPL when distributed. Reaching as far as copyright law will allow is the most direct way to reach that goal.

However, while the strategic goal is to bring as much Free Software into the world as possible, particular tactical considerations regarding software freedom dictate different means. Extending the copyleft effect as far as copyright law allows is not always the most prudent course in reaching the goal. In particular situations, even those of us with the goal of building a world where all published software is Free Software realize that full copyleft does not best serve us. The GNU Lesser General Public License ("GNU LGPL") was designed as a solution for such situations.

8.1 The First LGPL'd Program

The first example that FSF encountered where such altered tactics were needed was when work began on the GNU C Library. The GNU C Library would become (and today, now is) a drop-in replacement for existing C libraries. On a Unix-like operating system, C is the lingua franca and the C library is an essential component for all programs. It is extremely difficult to construct a program that will run with ease on a Unix-like operating system without making use of services provided by the C library — even if the program is written in a language other than C. Effectively, all user application programs that run on any modern Unix-like system must make use of the C library.

By the time work began on the GNU implementation of the C libraries, there were already many C libraries in existence from a variety of vendors. Every proprietary Unix vendor had one, and many third parties produced smaller versions for special purpose use. However, our goal was to create a C library that would provide equivalent functionality to these other C libraries on a Free Software operating system (which in fact happens today on modern GNU/Linux systems, which all use the GNU C Library).

Unlike existing GNU application software, however, the licensing implications of releasing the GNU C Library ("glibc") under GPL were somewhat different. Applications released under GPL would never themselves become part of proprietary software. However, if glibc were released under GPL, it would require that any application distributed for the GNU/Linux platform be released under GPL.

Since all applications on a Unix-like system depend on the C library, it means that they must link with that library to function on the system. In other words, all applications running on a Unix-like system must be combined with the C library to form a new whole derivative work that is composed of the original application and the C library. Thus, if glibc were GPL'd, each and every application distributed for use on GNU/Linux would also need to be GPL'd, since to even function, such applications would need to be combined into larger derivative works by linking with glibc.

At first glance, such an outcome seems like a windfall for Free Software advocates, since it stops all proprietary software development on GNU/Linux systems. However, the outcome is a bit more subtle. In a world where many C libraries already exist, many of which could easily be ported to GNU/Linux, a GPL'd glibc would be unlikely to succeed. Proprietary vendors would see the excellent opportunity to license their C libraries to anyone who wished to write proprietary software for GNU/Linux systems. The defacto standard for the C library on GNU/Linux would likely be not glibc, but the most popular proprietary one.

Meanwhile, the actual goal of releasing glibc under GPL — to ensure no proprietary applications on GNU/Linux — would be unattainable in this scenario. Furthermore, users of those proprietary applications would also be users of a proprietary C library, not the Free glibc.

The Lesser GPL was initially conceived to handle this scenario. It was clear that the existence of proprietary applications for GNU/Linux was inevitable. Since there were so many C libraries already in existence, a new one under GPL would not stop that tide. However, if the new C library were released under a license that permitted proprietary applications to link with it, but made sure that the library itself remained Free, an ancillary goal could be met. Users of proprietary applications, while they would not have the freedom to copy, share, modify and redistribute the application itself, would have the freedom to do so with respect to the C library.

There was no way the license of glibc could stop or even slow the creation of proprietary applications on GNU/Linux. However, loosening the restrictions on the licensing of glibc ensured that nearly all proprietary applications at least used a Free C library rather than a proprietary one. This trade-off is central to the reasoning behind the LGPL.

Of course, many people who use the LGPL today are not thinking in these terms. In fact, they are often choosing the LGPL because they are looking for a "compromise" between the GPL and the X11-style liberal licensing. However, understanding FSF's reasoning behind the creation of the LGPL is helpful when studying the license.

8.2 What's the Same?

Much of the text of the LGPL is identical to the GPL. As we begin our discussion of the LGPL, we will first eliminate the sections that are identical, or that have the minor modification changing the word "Program" to "Library."

First, §1 of LGPL, the rules for verbatim copying of source, are equivalent to those in GPL's §1.

Second, §8 of LGPL is equivalent §4 of GPL. In both licenses, this section handles termination in precisely the same manner.

§9 in LGPL is equivalent to §5 in GPL. Both sections assert that the license is a copyright license, and handle the acceptance of those copyright terms.

LGPL's §10 is equivalent to GPL's §6. They both protect the distribution system of Free Software under these licenses, to ensure that up, down, and throughout the distribution chain, each recipient of the software receives identical rights under the license and no other restrictions are imposed.

LGPL's §11 is GPL's §7. As discussed, it is used to ensure that other claims and legal realities, such as patent licenses and court judgments, do not trump the rights and permissions granted by these licenses, and requires that distribution be halted if such a trump is known to exist.

LGPL's §12 adds the same features as GPL's §8. These sections are used to allow original copyright holders to forbid distribution in countries with draconian laws that would otherwise contradict these licenses.

LGPL's §13 sets up FSF as the steward of the LGPL, just as GPL's

§9 does for GPL. Meanwhile, LGPL's §14 reminds licensees that copyright holders can grant exceptions to the terms of LGPL, just as GPL's §10 reminds licensees of the same thing.

Finally, the assertions of no warranty and limitations of liability are identical; thus LGPL's §15 and §16 are the same as GPL's §11 and §12.

As we see, the entire latter half of the license is identical. The parts which set up the legal boundaries and meta-rules for the license are the same. It is our intent that the two licenses operate under the same legal mechanisms and are enforced precisely the same way.

We strike a difference only in the early portions of the license. Namely, in the LGPL we go into deeper detail of granting various permissions to create derivative works, so the redistributors can make some proprietary derivatives. Since we simply do not allow the license to stretch as far as copyright law does regarding what derivative works must be relicensed under the same terms, we must go further to explain which derivative works we will allow to be proprietary. Thus, we'll see that the front matter of the LGPL is a bit more wordy and detailed with regards to the permissions granted to those who modify or redistribute the software.

8.3 Additions to the Preamble

Most of LGPL's Preamble is identical, but the last seven paragraphs introduce the concepts and reasoning behind creation of the license, presenting a more generalized and briefer version of the story with which we began our consideration of LGPL.

In short, FSF designed LGPL for those edge cases where the freedom of the public can better be served by a more lax licensing system. FSF doesn't encourage use of LGPL automatically for any software that happens to be a library; rather, FSF suggests that it only be used in specific cases, such as the following:

- To encourage the widest possible use of a Free Software library, so it becomes a de-facto standard over similar, although not interface-identical, proprietary alternatives
- To encourage use of a Free Software library that already has interface-identical proprietary competitors that are more developed
- To allow a greater number of users to get freedom, by encouraging proprietary companies to pick a Free alternative for its otherwise proprietary products

LGPL's preamble sets forth the limits to which the license seeks to go in chasing these goals. LGPL is designed to ensure that users who happen to acquire software linked with such libraries have full freedoms with respect to that library. They should have the ability to upgrade to a newer or modified Free version or to make their own modifications, even if they cannot modify the primary software program that links to that library.

Finally, the preamble introduces two terms used throughout the license to clarify between the different types of derivative works: "works that use the library," and "works based on the library." Unlike GPL, LGPL must draw some lines regarding derivative works. We do this here in this license because we specifically seek to liberalize the rights afforded to those who make derivative works. In GPL, we reach as far as copyright law allows. In LGPL, we want to draw a line that allows some derivative works copyright law would otherwise prohibit if the copyright holder exercised his full permitted controls over the work.

8.4 A Work that Uses the Library

In the effort to allow certain proprietary derivative works and prohibit others, LGPL distinguishes between two classes of derivative works: "works based on the library," and "works that use the library." The distinction is drawn on the bright line of binary (or runtime) derivative works and source code derivatives. We will first consider the definition of a "work that uses the library," which is set forth in LGPL §5.

We noted in our discussion of GPL §3 (discussed in Section 4.2 of this document) that binary programs when compiled and linked with GPL'd software are derivative works of that GPL'd software. This includes both linking that happens at compile-time (when the binary is created) or at runtime (when the binary – including library and main program both – is loaded into memory by the user). In GPL, binary derivative works are controlled by the terms of the license (in GPL §3), and distributors of such binary derivatives must release full corresponding source.

In the case of LGPL, these are precisely the types of derivative works we wish to permit. This scenario, defined in LGPL as "a work that uses the library," works as follows:

- A new copyright holder creates a separate and independent work, $\mathcal{I}$, that makes interface calls (e.g., function calls) to the LGPL'd work, called $\mathcal{L}$, whose copyright is held by some other party. Note that since $\mathcal{I}$ and $\mathcal{L}$ are separate and independent works, there is no copyright obligation on this new copyright holder with regard to the licensing of $\mathcal{I}$, at least with regard to the source code.

- The new copyright holder, for her software to be useful, realizes that it cannot run without combining $\mathcal{I}$ and $\mathcal{L}$. Specifically, when she creates a running binary program, that running binary must be a derivative work, called $\mathcal{L}$+$\mathcal{I}$, that the user can run.

- Since $\mathcal{L}$+$\mathcal{I}$ is a derivative work of both $\mathcal{I}$ and $\mathcal{L}$, the license of $\mathcal{L}$ (the LGPL) can put restrictions on the license of $\mathcal{L}$+$\mathcal{I}$. In fact, this is what LGPL does.

We will talk about the specific restrictions LGPL places on "works that use the library" in detail in Section 8.7. For now, focus on the logic related to how the LGPL places requirements on the license of $\mathcal{L}$+$\mathcal{I}$. Note, first of all, the similarity between this explanation and that in Section 4.1, which discussed the combination of otherwise separate and independent works with GPL'd code. Effectively, what LGPL does is say that when a new work is otherwise separate and independent, but has interface calls out to an LGPL'd library, then it is considered a "work that uses the library."

In addition, the only reason that LGPL has any control over the licensing of a "work that uses the library" is for the same reason that GPL has some say over separate and independent works. Namely, such controls exist because the *binary combination* ($\mathcal{L}$+$\mathcal{I}$) that must be created to make the separate work ($\mathcal{I}$) at all useful is a derivative work of the LGPL'd software ($\mathcal{L}$).

Thus, a two-question test that will help indicate if a particular work is a "work that uses the library" under LGPL is as follows:

1. Is the source code of the new copyrighted work, $\mathcal{I}$, a completely independent work that stands by itself, and includes no source code from $\mathcal{L}$?

2. When the source code is compiled, does it create a derivative work by combining with $\mathcal{L}$, either by static (compile-time) or dynamic (run-time) linking, to create a new binary work, $\mathcal{L}$+$\mathcal{I}$?

If the answers to both questions are "yes," then $\mathcal{I}$ is most likely a "work that uses the library." If the answer to the first question "yes," but the answer to the second question is "no," then most likely $\mathcal{I}$ is neither a "work that uses the library" nor a "work based on the library." If the answer to the first question is “no,” but the answer to the second question is “yes,” then an investigation into whether or not $\mathcal{I}$ is in fact a “work based on the library” is warranted.

8.5 A Work Based on the Library

In short, a “work based on the library” could be defined as any derivative work of LGPL’d software that cannot otherwise fit the definition of a “work that uses the library.” A “work based on the library” extends the full width and depth of copyright derivative works, in the same sense that GPL does.

Most typically, one creates a “work based on the library” by directly modifying the source of the library. Such a work could also be created by tightly integrating new software with the library. The lines are no doubt fuzzy, just as they are with GPL’d works, since copyright law gives us no litmus test for derivative works of a software program.

Thus, the test to use when considering whether something is a “work based on the library” is as follows:

1. Is the new work, when in source form, a derivative work under copyright law of the LGPL’d work?

2. Is there no way in which the new work fits the definition of a “work that uses the library”?

If the answer is “yes” to both these questions, then you most likely have a “work based on the library.” If the answer is “no” to the first but “yes” to the second, you are in a gray area between “work based on the library” and a “work that uses the library.”

In our years of work with the LGPL, however, we have never seen a work of software that was not clearly one or the other; the line is quite bright. At times, though, we have seen cases where a derivative work appeared in some ways to be a work that used the library and in other ways a work based on the library. We overcame this problem by dividing the work into smaller subunits. It was soon discovered that what we actually had were three distinct components: the original LGPL’d work, a specific set of works that used that library, and a specific set of works that were based on the library. Once such distinctions are established, the licensing for each component can be considered independently and the LGPL applied to each work as prescribed.

8.6 Subtleties in Works that Use the Library

In our discussion of the definition of "works that use the library," we left out a few more complex details that relate to lower-level programming details. The fourth paragraph of LGPL's §5 covers these complexities, and it has been a source of great confusion. Part of the confusion comes because a deep understanding of how compiler programs work is nearly mandatory to grasp the subtle nature of what §5, ¶4 seeks to cover. It helps some to note that this is a border case that we cover in the license only so that when such a border case is hit, the implications of using LGPL continue in the expected way.

To understand this subtle point, we must recall the way that a compiler operates. The compiler first generates object code, which are the binary representations of various programming modules. Each of those modules is usually not useful by itself; it becomes useful to a user of a full program when those modules are *linked* into a full binary executable.

As we have discussed, the assembly of modules can happen at compile-time or at runtime. Legally, there is no distinction between the two — both create a derivative work by copying and combining portions of one work and mixing them with another. However, under LGPL, there is a case in the compilation process where the legal implications are different. Specifically, while we know that a "work that uses the library" is one whose final binary is a derivative work, but whose source is not, there are cases where the object code — that intermediate step between source and final binary — is a derivative work created by copying verbatim code from the LGPL'd software.

For efficiency, when a compiler turns source code into object code, it sometimes places literal portions of the copyrighted library code into the object code for an otherwise separate independent work. In the normal scenario, the derivative would not be created until final assembly and linking of the executable occurred. However, when the compiler does this efficiency optimization, at the intermediate object code step, a derivative work is created.

LGPL's §5, ¶4 is designed to handle this specific case. The intent of the license is clearly that simply compiling software to "make use" of the library does not in itself cause the compiled work to be a "work based on the library." However, since the compiler copies verbatim, copyrighted portions of the library into the object code for the otherwise separate and independent work, it would actually cause that object file to be a "work based on the library." It is not FSF's intent that a mere compilation idiosyncrasy would change the requirements on the users of the LGPL'd software. This paragraph removes that restriction, allowing the implications of the license to be the same regardless of the specific mechanisms the compiler uses underneath to create the “work that uses the library.”

As it turns out, we have only once had anyone worry about this specific idiosyncrasy, because that particular vendor wanted to ship object code (rather than final binaries) to their customers and was worried about this edge condition. The intent of clarifying this edge condition is primarily to quell the worries of software engineers who understand the level of verbatim code copying that a compiler often does, and to help them understand that the full implications of LGPL are the same regardless of the details of the compilation progress.

8.7 LGPL §6: Distributing Works that Use the Library

Now that we have established a good working definition of works that “use” and works that “are based on” the library, we will consider the rules for distributing these two different works.

The rules for distributing “works that use the library” are covered in §6 of LGPL. §6 is much like GPL's §3, as it requires the release of source when a binary version of the LGPL'd software is released. Of course, it only requires that source code for the library itself be made available. The work that “uses” the library need not be provided in source form. However, there are also conditions in LGPL §6 to make sure that a user who wishes to modify or update the library can do so.

LGPL §6 lists five choices with regard to supplying library source and granting the freedom to modify that library source to users. We will first consider the option given by §6(b), which describes the most common way currently used for LGPL compliance on a “work that uses the library.”

§6(b) allows the distributor of a “work that uses the library” to simply use a dynamically linked, shared library mechanism to link with the library. This is by far the easiest and most straightforward option for distribution. In this case, the executable of the work that uses the library will contain only the “stub code” that is put in place by the shared library mechanism, and at runtime the executable will combine with the shared version of the library already resident on the user's computer. If such a mechanism is used, it must allow the user to upgrade and replace the library with interface-compatible versions and still be able to use the “work that uses the library.” However, all modern shared library mechanisms function as such, and thus §6(b) is the simplest option, since it does not even require that the distributor of the "work based on the library" ship copies of the library itself.

§6(a) is the option to use when, for some reason, a shared library mechanism cannot be used. It requires that the source for the library be included, in the typical GPL fashion, but it also has a requirement beyond that. The user must be able to exercise her freedom to modify the library to its fullest extent, and that means recombining it with the "work based on the library." If the full binary is linked without a shared library mechanism, the user must have available the object code for the "work based on the library," so that the user can relink the application and build a new binary.

The remaining options in §6 are very similar to the other choices provided by GPL §3. There are some additional options, but time does not permit us in this course to go into those additional options. In almost all cases of distribution under LGPL, either §6(a) or §6(b) are exercised.

8.8 Distribution of Works Based on the Library

Essentially, "works based on the library" must be distributed under the same conditions as works under full GPL. In fact, we note that LGPL's §2 is nearly identical in its terms and requirements to GPL's §2. There are again subtle differences and additions, which time does not permit us to cover in this course.

8.9 And the Rest

The remaining variations between LGPL and GPL cover the following conditions:

- Allowing a licensing "upgrade" from LGPL to GPL (in LGPL §3)
- Binary distribution of the library only, covered in LGPL §4, which is effectively equivalent to LGPL §3
- Creating aggregates of libraries that are not derivative works of each other, and distributing them as a unit (in LGPL §7)

Due to time constraints, we cannot cover these additional terms in detail, but they are mostly straightforward. The key to understanding LGPL is understanding the difference between a "work based on the library" and a “work that uses the library.” Once that distinction is clear, the remainder of LGPL is close enough to GPL that the concepts discussed in our more extensive GPL unit can be directly applied.

Chapter 9

Integrating the GPL into Business Practices

Since GPL'd software is now extremely prevalent through the industry, it is useful to have some basic knowledge about using GPL'd software in business and how to build business models around GPL'd software.

9.1 Using GPL'd Software In-House

As discussed in Sections 2.1 and 6.2 of this tutorial, the GPL only governs the activities of copying, modifying and distributing software programs that are not governed by the license. Thus, in FSF's view, simply installing the software on a machine and using it is not controlled or limited in any way by GPL. Using Free Software in general requires substantially fewer agreements and less license compliance activity than any known proprietary software.

Even if a company engages heavily in copying the software throughout the enterprise, such copying is not only permitted by §§1 and 3, but it is encouraged! If the company simply deploys unmodified (or even modified) Free Software throughout the organization for its employees to use, the obligations under the license are very minimal. Using Free Software has a substantially lower cost of ownership — both in licensing fees and in licensing checking and handling – than the proprietary software equivalents.

9.2 Business Models

Using Free Software in house is certainly helpful, but a thriving market for Free Software-oriented business models also exists. There is the traditional model of selling copies of Free Software distributions. Many companies, including IBM and Red Hat, make substantial revenue from this model. IBM primarily chooses this model because they have found that for higher-end hardware, the cost of the profit made from proprietary software licensing fees is negligible. The real profit is in the hardware, but it is essential that software be stable, reliable and dependable, and the users be allowed to have unfettered access to it. Free Software, and GPL'd software in particular (because IBM can be assured that proprietary versions of the same software will not exists to compete on their hardware) is the right choice.

Red Hat has actually found that a "convenience fee" for Free Software, when set at a reasonable price (around $60 or so), can produce some profit. Even though Red Hat's system is fully downloadable on their Web site, people still go to local computer stores and buy copies of their box set, which is simply a printed version of the manual (available under a Free license as well) and the Free Software system it documents.

However, custom support, service, and software improvement contracts are the most widely used models for GPL'd software. The GPL is central to their success, because it ensures that the code base remains common, and that large and small companies are on equal footing for access to the technology. Consider, for example, the GNU Compiler Collection (GCC). Cygnus Solutions, a company started in the early 1990s, was able to grow steadily simply by providing services for GCC — mostly consisting of new ports of GCC to different or new, embedded targets. Eventually, Cygnus was so successful that it was purchased by Red Hat where it remains a profitable division.

However, there are very small companies like CodeSourcery, as well as other medium-sized companies like MontaVista and OpenTV that compete in this space. Because the code-base is protect by GPL, it creates and demands industry trust. Companies can cooperate on the software and improve it for everyone. Meanwhile, companies who rely on GCC for their work are happy to pay for improvements, and for ports to new target platforms. Nearly all the changes fold back into the standard versions, and those forks that exist remain freely available.

A final common business model that is perhaps the most controversial is proprietary relicensing of a GPL'd code base. This is only an option for software in which a particular entity is the sole copyright holder. As discussed earlier in this tutorial, a copyright holder is permitted under copyright law to license a software system under her copyright as many different ways as she likes to as many different parties as she wishes.

Some companies, such as MySQL AB and TrollTech, use this to their financial advantage with regard to a GPL'd code base. The standard version is available from the company under the terms of the GPL. However, parties can purchase separate proprietary software licensing for a fee.

This business model is problematic because it means that the GPL'd code base must be developed in a somewhat monolithic way, because volunteer Free Software developers may be reluctant to assign their copyrights to the company because it will not promise to always and forever license the software as Free Software. Indeed, the company will surely use such code contributions in proprietary versions licensed for fees.

9.3 Ongoing Compliance

GPL compliance is in fact a very simple matter – much simpler than typical proprietary software agreements and EULAs. Usually, the most difficult hurdle is changing from a proprietary software mindset to one that seeks to foster a community of sharing and mutual support. Certainly complying with the GPL from a users' perspective gives substantially fewer headaches than proprietary license compliance.

For those who go into the business of distributing *modified* versions of GPL'd software, the burden is a bit higher, but not by much. The glib answer is that by releasing the whole product as Free Software, it is always easy to comply with the GPL. However, admittedly to the dismay of FSF, many modern and complex software systems are built using both proprietary and GPL'd components that are not legally derivative works of each other. Sometimes, it is easier simply to improve existing GPL'd application than to start from scratch. In exchange for that benefit, the license requires that the modifier give back to the commons that made the work easier in the first place. It is a reasonable trade-off and a way to help build a better world while also making a profit.

Note that FSF does provide services to assist companies who need assistance in complying with the GPL. You can contact FSF's GPL Compliance Labs at <compliance@fsf.org>.

If you are particularly interested in matters of GPL compliance, we recommend the second course in this series, *GPL Compliance Case Studies and Legal Ethics in Free Software Licensing*, in which we discuss some real GPL violation cases that FSF has worked to resolve. Consideration of such cases can help give insight on how to handle GPL compliance in new situations.

The GNU General Public License

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change Free Software—to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of Free Software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of Free Software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new Free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this Free Software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any Free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a Free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

Terms and Conditions For Copying, Distribution and Modification

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program," below, refers to any such program or work, and 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. (Hereinafter, translation is included without limitation in the term "modification.") Each licensee is addressed as "you."

   Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

   You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

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.

   (c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

   These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this

License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

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,

   (b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

   (c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

   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, 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.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

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. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who

receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

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. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

   Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version," you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author

Appendix: How to Apply These Terms to Your New Programs

The hypothetical commands `show w` and `show c` should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than `show w` and `show c`; they could even be mouse-clicks or menu items—whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a “copyright disclaimer” for the program, if necessary. Here is a sample; alter the names:

> Yoyodyne, Inc., hereby disclaims all copyright interest in the program
> ‘Gnomovision’ (which makes passes at compilers) written by James Hacker.
>
> signature of Ty Coon, 1 April 1989
> Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

The GNU Lesser General Public License

[This is the first released version of the Lesser GPL. It also counts as the successor of the GNU Library Public License version 2, hence the version number 2.1.]

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change Free Software—to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software packages—typically libraries—of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of Free Software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of Free Software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new Free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the Free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any Free program. We wish to make sure that a company cannot effectively restrict the users of a Free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-Free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the "Lesser" General Public License because it does Less to protect the user's freedom than the ordinary General Public License.

It also provides other Free Software developers Less of an advantage over competing non-Free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-Free programs must be allowed to use the library. A more frequent case is that a Free library does the same job as widely used non-Free libraries. In this case, there is little to gain by limiting the Free library to Free Software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-Free programs enables a greater number of people to use a large body of Free software. For example, permission to use the GNU C Library in non-Free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the library has the freedom and the wherewithal to run that program using a modified version of the library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library." The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

GNU Lesser General Public License Terms and Conditions For Copying, Distribution and Modification

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you."

   A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

   The "library," below, refers to any such software library or work which has been distributed under these terms. A "work based on the library" means either the library or any derivative work under copyright law: that is to say, a work containing the library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification.")

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, 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 library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the library is not restricted, and output from such a program is covered only if its contents constitute a work based on the library (independent of the use of the library in a tool for writing it). Whether that is true depends on what the library does and what the program that uses the library does.

1. You may copy and distribute verbatim copies of the library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the library.

   You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the library or any portion of it, thus forming a work based on the library, 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) The modified work must itself be a software library.

   (b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.

   (c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

(d) If a facility in the modified library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

    (For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the library.

In addition, mere aggregation of another work not based on the library with the library (or with a work based on the library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

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. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the library into a program that is not a library.

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, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

   If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object 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.

   However, linking a "work that uses the library" with the library creates an executable that is a derivative of the library (because it contains portions of the library), rather than a "work that uses the library." The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

   When a "work that uses the library" uses material from a header file that is part of the library, the object code for the work may be a derivative work of the library even though the source code is not. Whether this is true is especially significant if the work can be linked without the library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

   If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the library will still fall under Section 6.)

Otherwise, if the work is a derivative of the library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the library itself.

6. As an exception to the Sections above, you may also combine or link a "work that uses the library" with the library to produce a work containing portions of the library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

   You must give prominent notice with each copy of the work that the library is used in it and that the library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

   (a) Accompany the work with the complete corresponding machine-readable source code for the library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the library, with the complete machine-readable "work that uses the library," as object code and/or source code, so that the user can modify the library and then relink to produce a modified executable containing the modified library. (It is understood that the user who changes the contents of definitions files in the library will not necessarily be able to recompile the application to use the modified definitions.)

   (b) Use a suitable shared library mechanism for linking with the library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

(c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

(d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

(e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be 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.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the library together in an executable that you distribute.

7. You may place library facilities that are a work based on the library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the library and of the other library facilities is otherwise permitted, and provided that you do these two things:

   (a) Accompany the combined library with a copy of the same work based on the library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

   (b) Give prominent notice with the combined library of the fact that part of it is a work based on the library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the library is void, and will automatically terminate your rights under this

License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the library (or any work based on the library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the library or works based on it.

10. Each time you redistribute the library (or any work based on the library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

11. 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 library at all. For example, if a patent license would not permit royalty-free redistribution of the library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the library.

    If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

    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. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

    Each version is given a distinguishing version number. If the library specifies a version number of this License which applies to it and "any later version," you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the library into other Free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the Free status of all derivatives of our Free software and of promoting the sharing and reuse of software generally.

No Warranty

15. Because the library is licensed free of charge, there is no warranty for the library, to the extent permitted by

How to Apply These Terms to Your New Libraries

James Hacker.

signature of Ty Coon, 1 April 1990
Ty Coon, President of Vice

That's all there is to it!