Copyleft and the GNU General Public License:

A Comprehensive Tutorial and Guide

As a public, collaborative project, this Guide is primarily composed of the many contributions received via its public contribution process. Please review its Git logs for full documentation of all contributions, and Appendix A contains a list of third-party works from which some material herein was adapted.
The most recent version is available online at `https://copyleft.org/guide/`. Patches are indeed welcome to this material. Sources can be found in the Git repository at `https://k.copyleft.org/guide/`.

CONTENTS

PREFACE

This tutorial is the culmination of nearly a decade of studying and writing about software freedom licensing and the GPL. Each part of this tutorial is a course unto itself, educating the reader on a myriad of topics from the deep details of the GPLv2 and GPLv3, common business models in the copyleft licensing area (both the friendly and unfriendly kind), best practices for compliance with the GPL, for engineers, managers, and lawyers, as well as real-world case studies of GPL enforcement matters.

It is unlikely that all the information herein is necessary to learn all at once, and therefore this tutorial likely serves best as a reference book. The material herein has been used as the basis for numerous live tutorials and discussion groups since 2002, and the materials have been periodically updated. They likely stand on their own as excellent reference material.

However, if you are reading these course materials without attending a live tutorial session, please note that this material is merely a summary of the highlights of the various CLE and other tutorial courses based on this material. Please be aware that during the actual courses, class discussion and presentation supplements this printed curriculum. Simply reading this material is **not equivalent** to attending a course.

Part I

Detailed Analysis of the GNU GPL and Related Licenses

This part of the tutorial gives a comprehensive explanation of the most popular Free Software copyright license, the GNU General Public License ("GNU GPL", or sometimes just "GPL") – both version 2 ("GPLv2") and version 3 ("GPLv3") – and teaches lawyers, software developers, managers and businesspeople how to use the GPL (and GPL'd software) successfully both as a community-building "Constitution" for a software project, and to incorporate copylefted software into a new Free Software business and in existing, successful enterprises.

To benefit from this part of the tutorial, readers should have a general familiarity with software development processes. A basic 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 or advise software businesses that modify and/or redistribute software under the 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 this part of the tutorial, readers can expect to have learned the following:

- The freedom-defending purpose of various terms in the GNU GPLv2 and GPLv3.
- The differences between GPLv2 and GPLv3.
- The redistribution options under the GPLv2 and GPLv3.
- The obligations when modifying GPLv2'd or GPLv3'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 LGPLv2.1 and LGPLv3, and how they differ from the GPLv2 and GPLv3, respectively.
- The basics to begin understanding the complexities regarding derivative and combined works of software.

CHAPTER 1

WHAT IS SOFTWARE FREEDOM?

Study of the GNU General Public License (herein, abbreviated as *GNU GPL* or just *GPL*) must begin by first considering the broader world of software freedom. The GPL was not created in a vacuum. 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 preeminent organization that upholds, defends and promotes the philosophy of software freedom. A prerequisite for understanding both of the popular versions of the GPL (GPLv2 and GPLv3) and their terms and conditions is a basic understanding of the principles behind them. The GPL family of licenses are unlike nearly all other software licenses in that they are 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 website at `http://fsf.org/ philosophy/free-sw.html`. This section presents an abbreviated version that will focus on the parts that are most pertinent to the GPL.

A particular user has software freedom with respect to a particular program if that user has the following freedoms:

- The freedom to run the program, for any purpose.
- The freedom to study how the program works, and modify it
- The freedom to redistribute copies.
- The freedom to distribute copies of modified versions to others.

The focus on "a particular user" is particularly pertinent here. It is not uncommon for a subset of a specific program's user base to have these freedoms, while other users of the same version the program have none or only some of these freedoms. Section 12.2 talks in detail about how this can unfortunately happen even if a program is released under the GPL.

Many people refer to software with these freedoms as "Open Source." Besides having a different political focus from those who call such software by the name "Free Software",[1], those who call the software "Open Source" are often focused on a side issue. Specifically, 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 that applies to that source code.

Software freedom is only complete when no restrictions are imposed on how these freedoms are exercised. Specifically, users and programmers can exercise these freedoms noncommercially or commercially. Licenses

[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/licensing/essays/free-software-for-freedom.html`.

that grant these freedoms for noncommercial activities but prohibit them for commercial activities are considered non-free. The Open Source Initiative (*OSI*) (the arbiter of what is considered "Open Source") also regards such licenses as inconsistent with its "Open Source Definition".

In general, software for which any of these freedoms are restricted in any way is called "nonfree" software. Some use the term "proprietary software" more or less interchangeably with "nonfree software". The FSF published a useful explanation of various types of software and how they relate to one another.

Keep in mind that none of the terms "software freedom", "open source" and "free software" are known to be trademarked or otherwise legally restricted by any organization in any jurisdiction. As such, it's quite common that these terms are abused and misused by parties who wish to bank on the popularity of software freedom. When one considers using, modifying or redistributing a software package that purports to be Open Source or Free Software, one **must** verify that the license grants software freedom.

Furthermore, throughout this text, we generally prefer the term "software freedom", as this is the least ambiguous term available to describe software that meets the Free Software Definition. For example, it is well known and often discussed that the adjective "free" has two unrelated meanings in English: "free as in freedom" and "free as in price". Meanwhile, the term "open source" is even more confusing, because it appears to refer only to the "freedom to study", which is merely a subset of one of the four freedoms.

The remainder of this section considers each of each component of software freedom in detail.

1.1.1 The Freedom to Run

The first tenet of software freedom is the user's fully unfettered right to run the program. The software's license must permit any conceivable use of the software. 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; but such is quite common today. Most End User License Agreements (EULAs) that cover most proprietary software typically restrict some types of uses. Such restrictions of any kind are an unacceptable restriction on software freedom.

1.1.2 The Freedom to Change and Modify

Perhaps the most useful right of software freedom is the users' right to change, modify and adapt the software to suit their needs. Access to the source code and related build and installation scripts are an essential part of this freedom. Without the source code, and the ability to build and install the binary applications from that source, users cannot effectively exercise this freedom.

Programmers directly benefit from this freedom. However, this freedom remains important to users who are not programmers. While it may seem counterintuitive at first, non-programmer users often 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 user groups. To make use of such help they must either have the freedom to recruit programmers who might altruistically assist them to modify their software, or to at least follow rote instructions to make basic modifications themselves.

More commonly, users also exercise this freedom commercially. Each user, or group of users, may 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 modifications.

1.1.3 The Freedom to Copy and Share

Users share Free Software in a variety of ways. Software freedom 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). Licenses that respect software freedom, therefore, permit altruistic sharing of software among friends.

The commercial environment also benefits from this freedom. Commercial sharing includes selling copies of Free Software: that is, Free Software can be distributed for any monetary price to anyone. Those who

redistribute Free Software commercially also have the freedom to selectively distribute (i.e., you can pick your customers) and to set prices at any level that redistributor sees fit.

Of course, most 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 billions of 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, some distribution vendors provide immediate security and upgrade distribution via a special network service. Such distribution is not necessarily contradictory with software freedom.

(Section 12.2 of this tutorial talks in detail about some common 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 software freedom community is built on the pillar of altruistic sharing of improved Free Software. Historically it was typical for a Free Software project to sprout a mailing list where improvements would be shared freely among members of the development community.[2] Such noncommercial sharing is the primary reason that Free Software thrives.

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 augmenters 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 needs the right to modify and improve the software for the customer to correct any problems that are beyond mere user error.

Software freedom licenses also permit any entity to distribute 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 previous section set forth key freedoms and rights that are referred to as "software freedom". This section discusses the licensing mechanisms used to enable software freedom. These licensing mechanisms were ultimately created as a community-oriented "answer" to the existing proprietary software licensing mechanisms. Thus, first, consider carefully why proprietary software exists in the first place.

The primary legal regime that applies to software is copyright law. Proprietary software exists at all only because copyright law governs software.[3] Copyright law, with respect to software, typically governs copying, modifying, and redistributing that software (For details of this in the USA, see § 106 and § 117 of Title 17 of

[2] This is still commonly the case, though today there are additional ways of sharing Free Software.

[3] This statement is admittedly an oversimplification. Patents and trade secrets can cover software and make it effectively non-Free, and 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, and therefore this section focuses on how copyright restrictions make software proprietary.

the *United States Code*).[4] By law (in the USA and in most other jurisdictions), the copyright holder (most typically, 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 is not a natural state, it is a legal construction. In the USA, the Constitution permits, but does not require, the creation of copyright law as federal legislation. Software, since it is an "original work of authorship fixed in any tangible medium of expression ... from which they can be perceived, reproduced, or otherwise communicated, either directly or with the aid of a machine or device" (as stated in 17 USC § 102), is thus covered by the statute, and is copyrighted by default.

However, 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 redistributing those modified versions.[5]

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 can often *disclaim* their copyright. (For example, under USA copyright law it is possible for a copyright holder to engage in conduct resulting in abandonment of copyright.) If copyright is disclaimed, the software is effectively no longer restricted by copyright law. Software not restricted by copyright is in the "public domain."

1.2.1 Public Domain Software

In the USA and other countries that are parties to the Berne Convention on Copyright, software is copyrighted automatically by the author when she fixes the software in a tangible medium. In the software world, this usually means typing the source code of the software into a file.

Imagine if authors could truly disclaim those default controls of copyright law. If so, the software is in the public domain — no longer covered by copyright. Since copyright law is the construction allowing for most restrictions on software (i.e., prohibition of copying, modification, and redistribution), removing the software from the copyright system usually yields software freedom for its users.

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

Copyright holders who state that they are releasing their code into the public domain are effectively renouncing copyright controls on the work. The law gave the copyright holders exclusive controls over the work, and they chose to waive those controls. Software that is, in this sense, in the public domain is conceptualized by the developer as having no copyright and thus no license. The software freedoms discussed in Section 1.1 are all granted because there is no legal system in play to take them away.

Admittedly, a discussion of public domain software is an oversimplified example. Because copyright controls are usually automatically granted and because, in some jurisdictions, some copyright controls cannot be waived (see Section 1.2.4 for further discussion), many copyright holders sometimes incorrectly believe a work has been placed in the public domain. Second, due to aggressive lobbying by the entertainment industry, the "exclusive Right" of copyright, that was supposed to only exist for "Limited Times" according to the USA Constitution, appears to be infinite: simply purchased on the installment plan rather than in whole. Thus, we must assume no works of software will fall into the public domain merely due to the passage of time.

Nevertheless, under USA law it is likely that the typical disclaimers of copyright or public domain dedications we see in the Free Software world would be interpreted by courts as copyright abandonment, leading to a situation in which the user effectively receives a maximum grant of copyright freedoms, similar to a maximally-permissive Free Software license.

[4] Copyright law in general also governs "public performance" of copyrighted works. There is no generally agreed definition for public performance of software and both GPLv2 and GPLv3 do not restrict public performance.

[5] Note that this is again an oversimplification; the complexities with this argument are discussed in Section 1.2.3.

The best example of software known to truly be in the public domain is software that is published by the USA government. Under 17 USC 101 § 105, all works published by the USA Government are not copyrightable in the USA.

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 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 strategy of utilizing copyright law to pursue the policy goal of fostering and encouraging the equal and inalienable right to copy, share, modify and improve creative works of authorship. Copyleft (as a general term) describes any method that utilizes the copyright system to achieve the aforementioned goal. Copyleft as a concept is usually implemented in the details of a specific copyright license, such as the GNU General Public License (GPL) and the Creative Commons Attribution Share Alike License (the latter of which is the license of this work itself). Copyright holders of creative work can unilaterally implement these licenses for their own works to build communities that collaboratively share and improve those copylefted creative works.

Copyleft uses functional parts of the copyright system to achieve an unusual result (legal protection for free sharing). Copyleft modifies, or "hacks" copyright law, which is usually employed to strengthen the rights of authors or publishers, to strengthen instead the rights of users. Thus, Copyleft is a legal strategy and mechanism 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's "reciprocity" or "share and share alike" rule protects both developers, who avoid facing a "prioritized" competitor of their project, and users, who can be sure that they will have all four software freedoms — not only in the present version of the program they use, but in all its future improved versions.

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, written as a "copyright license". The GPL implements the concept of copyleft for software-oriented and other functional works of a technical nature. The "CC BY SA" license implements copyleft for works of textual, musical and visual authorship, such as this tutorial.

Copyleft advocates often distinguish between the concept of a "strong copyleft" or a "weak copyleft". However, "strong vs. weak" copyleft is not a dichotomy, it's a spectrum. The strongest copylefts strive to the exclusive rights that copyright grants to authors as extensively as possible to maximize software freedom.

As a copyleft gets "weaker", the copyleft license typically makes "trade offs" that might impede software freedom, but reach other tactic goals for the community of users and developers of the work.

In other words, strong copyleft licenses place the more requirements on how "the work" is licensed. The unit of copyright law is "the work". In that sense, the "work" referenced by the licenses is anything that can be copyrighted or will be subject to the terms of copyright law. Strong copyleft licenses exercise their scope fully. Anything which is "a work" or a "work based on a work" licensed under a strong copyleft is subject to its requirements, including the requirement of complete, corresponding source code[6]. Thus, copyleft licenses, particularly strong ones, seek to ensure the same license covers every version of "work based on the work", as recognized by local copyright law, and thereby achieve the specific strategic policy aim of ensuring software freedom for all users, developers, authors, and readers who encounter the copylefted work.

1.2.3 Software and Non-Copyright Legal Regimes

The use, modification and distribution of software, like many endeavors, simultaneously interacts with multiple different legal regimes. As was noted early via footnotes, copyright is merely the *most common way* to restrict users' rights to copy, share, modify and/or redistribute software. However, proprietary software licenses typically use every mechanism available to subjugate users. For example:

- Unfortunately, despite much effort by many in the software freedom community to end patents that read on software (i.e., patents on computational ideas), they still exist. As such, a software program might otherwise seem to be unrestricted, but a patent might read on the software and ruin everything for its users.[7]
- Digital Restrictions Management (usually called *DRM*) is often used to impose technological restrictions on users' ability to exercise software freedom that they might otherwise be granted.[8] The simplest (and perhaps oldest) form of DRM, of course, is separating software source code (read by humans), from their compiled binaries (read only by computers). Furthermore, 17 USC §1201 often prohibits users legally from circumventing some of these DRM systems.
- Most EULAs also include a contractual agreement that bind users further by forcing them to agree to a contractual, prohibitive software license before ever even using the software.

Thus, most proprietary software restricts users via multiple interlocking legal and technological means. Any license that truly respect the software freedom of all users must not only grant appropriate copyright permissions, but also *prevent* restrictions from other legal and technological means like those listed above.

1.2.4 Non-USA Copyright Regimes

Generally speaking, copyright law operates similarly enough in countries that have signed the Berne Convention on Copyright, and software freedom licenses have generally taken advantage of this international standardization of copyright law. However, copyright law does differ from country to country, and commonly, software freedom licenses like the GPL must be considered under the copyright law in the jurisdiction where any licensing dispute occurs.

Those who are most familiar with the USA's system of copyright often are surprised to learn that there are certain copyright controls that cannot be waived nor disclaimed. Specifically, many copyright regimes outside the USA recognize a concept of moral rights of authors. Typically, moral rights are fully compatible with respecting software freedom, as they are usually centered around controls that software freedom licenses generally respect, such as the right of an authors to require proper attribution for their work.

[6]Copyleft communities' use of the term "strong copyleft" is undoubtedly imprecise. For example, most will call the GNU GPL a "strong copyleft" license, even though the GPL itself has various exceptions, such as the GPLv3's system library exception written into the text of the license itself. Furthermore, the copyleft community continues to debate where the a license cross the line from "strong copyleft" to "license that fails to respect software freedom", although ultimately these debates are actually regarding whether the license fits Free Software definition at all.

[7]See §§ 6, 7.5, 9.14 for more discussion on how the patent system interacts with copyleft, and read Richard M. Stallman's essay, *Let's Limit the Effect of Software Patents, Since We Can't Eliminate Them* for more information on the problems these patents present to society.

[8]See § 9.5 for more information on how GPL deals with this issue.

1.3 A Community of Equality

The previous section described the principles of software freedom, a brief introduction to mechanisms that typically block these freedoms, and the simplest ways that copyright holders might grant those freedoms to their users for their copyrighted works of software. The previous section also introduced the idea of *copyleft*: a licensing mechanism to use copyright to not only grant software freedom to users, but also to uphold those rights against those who might seek to curtail them.

Copyleft, as defined in § 1.2.2, is a general term for this mechanism. The remainder of this text will discuss details of various real-world implementations of copyleft – most notably, the GPL.

This discussion begins first with some general explanation of what the GPL is able to do in software development communities. After that brief discussion in this section, deeper discussion of how GPL accomplishes this in practice follows in the next chapter.

Simply put, though, the GPL ultimately creates a community of equality for both business and noncommercial users.

1.3.1 The Noncommercial Community

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 the 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 Community

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.[9] 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 the 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

[9] See Andrew Tridgell's "A bit of history and a bit of fun"

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.[10]

[10]This quotation is Copyright © 2002, Lawrence Lessig. It is licensed under the terms of the "Attribution License" version 1.0 or any later version as published by Creative Commons.

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.

CHAPTER 2

A TALE OF TWO COPYLEFT LICENSES

While determining the proper methodology and criteria to yield an accurate count remains difficult, the GPL is generally considered one of the most widely used Free Software licenses. For most of its history — for 16 years from June 1991 to June 2007 — there was really only one version of the GPL, version 2.

However, the GPL had both earlier versions before version 2, and, more well known, a revision to version 3.

2.1 Historical Motivations for the General Public License

The earliest license to grant software freedom was likely the Berkeley Software Distribution ("BSD") license. This license is typical of what are often called lax, highly permissive licenses. Not unlike software in the public domain, these non-copyleft licenses (usually) grant software freedom to users, but they do not go to any effort to uphold that software freedom for users. The so-called "downstream" (those who receive the software and then build new things based on that software) can restrict the software and distribute further.

The GNU's Not Unix ("GNU") project, which Richard M. Stallman ("RMS") founded in 1984 to make a complete Unix-compatible operating system implementation that assured software freedom for all. However, RMS saw that using a license that gave but did not assure software freedom would be counter to the goals of the GNU Project. RMS invented "copyleft" as an answer to that problem, and began using various copyleft licenses for the early GNU Project programs.[1]

2.2 Proto-GPLs And Their Impact

The earliest copyleft licenses were specific to various GNU programs. For example, The Emacs General Public License was likely the first copyleft license ever published. Interesting to note that even this earliest copyleft license contains a version of the well-known GPL copyleft clause:

> You may modify your copy or copies of GNU Emacs ... provided that you also ... cause the whole of any work that you distribute or publish, that in whole or in part contains or is a derivative of GNU Emacs or any part thereof, to be licensed at no charge to all third parties on terms identical to those contained in this License Agreement.

This simply stated clause is the fundamental innovation of copyleft. Specifically, copyleft *uses* the copyright holders' controls on permission to modify the work to add a conditional requirement. Namely, down-

[1] RMS writes more fully about this topic in his essay entitled simply *The GNU Project*. For those who want to hear the story in his own voice, speech recordings of his talk, *The Free Software Movement and the GNU/Linux Operating System* are also widely available

stream users may only have permission to modify the work if they pass along the same permissions on the modified version that came originally to them.

These original program-specific proto-GPLs give an interesting window into the central ideas and development of copyleft. In particular, reviewing them shows how the text of the GPL we know has evolved to address more of the issues discussed earlier in § 1.2.3.

2.3 The GNU General Public License, Version 1

In January 1989, the FSF announced that the GPL had been converted into a "subroutine" that could be reused not just for all FSF-copyrighted programs, but also by anyone else. As the FSF claimed in its announcement of the GPLv1:[2]

> To make it easier to copyleft programs, we have been improving on the legalbol architecture of the General Public License to produce a new version that serves as a general-purpose subroutine: it can apply to any program without modification, no matter who is publishing it.

This, like many inventive ideas, seems somewhat obvious in retrospect. But, the FSF had some bright people and access to good lawyers when it started. It took almost five years from the first copyleft licenses to get to a generalized, reusable GPLv1. In the context and mindset of the 1980s, this is not surprising. The idea of reusable licensing infrastructure was not only uncommon, it was virtually nonexistent! Even the early BSD licenses were simply copied and rewritten slightly for each new use.[3] The GPLv1's innovation of reusable licensing infrastructure, an obvious fact today, was indeed a novel invention for its day.[4]

2.4 The GNU General Public License, Version 2

The GPLv2 was released two and a half years after GPLv1, and over the following sixteen years, it became the standard for copyleft licensing until the release of GPLv3 in 2007 (discussed in more detail in the next section).

While this tutorial does not discuss the terms of GPLv1 in detail, it is worth noting below the three key changes that GPLv2 brought:

- Software patents and their danger are explicitly mentioned, inspiring (in part) the addition of GPLv2 §§5–7. (These sections are discussed in detail in § 7.2, § 7.3 and § 7.5 of this tutorial.)
- GPLv2 §2's copyleft terms are expanded to more explicitly discuss the issue of combined works. (GPLv2 §2 is discussed in detail in § 5.1 in this tutorial).
- GPLv2 §3 includes more detailed requirements, including the phrase "the scripts used to control compilation and installation of the executable", which is a central component of current GPLv2 enforcement. (GPLv2 §3 is discussed in detail in § 5.2 in this tutorial).

The next chapter discusses GPLv2 in full detail, and readers who wish to dive into the section-by-section discussion of the GPL should jump ahead now to that chapter. However, the most interesting fact to note here is how GPLv2 was published with little fanfare and limited commentary. This contrasts greatly with the creation of GPLv3.

[2] The announcement of GPLv1 was published in the GNU's Bulletin, vol 1, number 6 dated January 1989. (Thanks very much to Andy Tai for his consolidation of research on the history of the pre-v1 GPL's.)

[3] It remains an interesting accident of history that the early BSD problematic "advertising clause" (discussion of which is somewhat beyond the scope of this tutorial) lives on into current day, simply because while the University of California at Berkeley gave unilateral permission to remove the clause from *its* copyrighted works, others who adapted the BSD license with their own names in place of UC-Berkeley's never have.

[4] We're all just grateful that the FSF also opposes business method patents, since the FSF's patent on a "method for reusable licensing infrastructure" would have not expired until 2006!

2.5 The GNU General Public License, Version 3

RMS began drafting GPLv2.2 in mid-2002, and FSF ran a few discussion groups during that era about new text of that license. However, rampant violations of the GPL required more immediate attention of FSF's licensing staff, and as such, much of the early 2000's was spent doing GPL enforcement work.[5] In 2006, FSF began in earnest drafting work for GPLv3.

The GPLv3 process began in earnest in January 2006. It became clear that many provisions of the GPL could benefit from modification to fit new circumstances and to reflect what the entire community learned from experience with version 2. Given the scale of revision it seems proper to approach the work through public discussion in a transparent and accessible manner.

The GPLv3 process continued through June 2007, culminating in publication of GPLv3 and LGPLv3 on 29 June 2007, AGPLv3 on 19 November 2007, and the GCC Runtime Library Exception on 27 January 2009.

All told, four discussion drafts of GPLv3, two discussion drafts of LGPLv3 and two discussion drafts of AGPLv3 were published and discussed. Ultimately, FSF remained the final arbiter and publisher of the licenses, and RMS himself their primary author, but input was sought from many parties, and these licenses do admittedly look and read more like legislation as a result. Nevertheless, all of the "v3" group are substantially better and improved licenses.

GPLv3 and its terms are discussed in detail in Chapter 9.

2.6 The Innovation of Optional "Or Any Later" Version

An interesting fact of all GPL licenses is that there are ultimately multiple choices for use of the license. The FSF is the primary steward of GPL (as discussed later in § 8.1 and § 9.17). However, those who wish to license works under GPL are not required to automatically accept changes made by the FSF for their own copyrighted works.

Each licensor may chose three different methods of licensing, as follows:

- explicitly name a single version of GPL for their work (usually indicated in shorthand by saying the license is "GPLvX-only"), or
- name no version of the GPL, thus they allow their downstream recipients to select any version of the GPL they choose (usually indicated in shorthand by saying the license is simply "GPL"), or
- name a specific version of GPL and give downstream recipients the option to choose that version "or any later version as published by the FSF" (usually indicated by saying the license is "GPLvX-or-later")[6]

Oddly, this flexibility has received (in the opinion of the authors, undue) criticism, primarily because of the complex and oft-debated notion of "license compatibility" (which is explained in detail in § 9.10). Copyleft licenses are generally incompatible with each other, because the details of how they implement copyleft differs. Specifically, copyleft works only because of its requirement that downstream licensors use the *same* license for combined and modified works. As such, software licensed under the terms of "GPLv2-only" cannot be combined with works licensed "GPLv3-or-later". This is admittedly a frustrating outcome.

Other copyleft licenses that appeared after GPL, such as the Creative Commons "Attribution-Share Alike" licenses, the Eclipse Public License and the Mozilla Public License **require** all copyright holders choosing to use any version of those licenses to automatically allow use of their copyrighted works under new versions.[7] Of course, Creative Commons, the Eclipse Foundation, and the Mozilla Foundation (like the FSF) have generally served as excellent stewards of their licenses. Copyright holders using those licenses seems to

[5] More on GPL enforcement is discussed in Part II of this tutorial.

[6] The shorthand of "GPLX+" is also popular for this situation. The authors of this tutorial prefer "-or-later" syntax, because it (a) mirrors the words "or" and "later from the licensing statement, (b) the X+ doesn't make it abundantly clear that X is clearly included as a license option and (c) the + symbol has other uses in computing (such as with regular expressions) that mean something different.

[7] CC-BY-SA-2.0 and greater only permit licensing of adaptations under future versions; 1.0 did not have any accomodation for future version compatibility.

find it acceptable to fully delegate all future licensing decisions for their copyrights to these organizations without a second thought.

However, note that FSF gives herein the control of copyright holders to decide whether or not to implicitly trust the FSF in its work of drafting future GPL versions. The FSF, for its part, does encourage copyright holders to chose by default "GPLvX-or-later" (where X is the most recent version of the GPL published by the FSF). However, the FSF **does not mandate** that a choice to use any GPL requires a copyright holder ceding its authority for future licensing decisions to the FSF. In fact, the FSF considered this possibility for GPLv3 and chose not to do so, instead opting for the third-party steward designation clause discussed in Section 9.17.

2.7 Complexities of Two Simultaneously Popular Copylefts

Obviously most GPL advocates would prefer widespread migration to GPLv3, and many newly formed projects who seek a copyleft license tend to choose a GPLv3-based license. However, many existing copylefted projects continue with GPLv2-only or GPLv2-or-later as their default license.

While GPLv3 introduces many improvements — many of which were designed to increase adoption by for-profit companies — GPLv2 remains a widely used and extremely popular license. The GPLv2 is, no doubt, a good and useful license.

However, unlike GPLv1 before it, GPLv2 remains an integral part of the copyleft licensing infrastructure. As such, those who seek to have expertise in current topics of copyleft licensing need to study both the GPLv2 and GPLv3 family of licenses.

Furthermore, GPLv3 is more easily understood by first studying GPLv2. This is not only because of their chronological order, but also because much of the discussion material available for GPLv3 tends to talk about GPLv3 in contrast to GPLv2. As such, a strong understanding of GPLv2 helps in understanding most of the third-party material found regarding GPLv3. Thus, the following chapter begins a deep discussion of GPLv2.

CHAPTER 3

RUNNING SOFTWARE AND VERBATIM COPYING

This chapter begins the deep discussion of the details of the terms of GPLv2. In this chapter, we consider the first two sections: GPLv2 §§0–2. These are the straightforward sections of the GPL that define the simplest rights that the user receives.

3.1 GPLv2 §0: Freedom to Run

GPLv2 §0, the opening section of GPLv2, sets forth that copyright law governs the work. 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, GPLv2 §0 makes an inference that copyright law is the only system that can restrict the software. 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 GPLv2 (which they cannot; see Sections 7.3 and 7.5). This is very important, because the Free Software community heavily supports users' rights to "fair use" and "unregulated use" of copyrighted material. GPLv2 asserts through this clause that it supports users' rights to fair and unregulated uses.

Fair use (called "fair dealing" in some jurisdictions) of copyrighted material is an established legal doctrine that permits certain activities regardless of whether copyright law would otherwise restrict those 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 portions of the text in a larger work so as to criticize or suggest changes. This fair use right is commonly used on mailing lists when discussing potential improvements or changes to Free Software.

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. The right to read something that you have access to is and should remain unregulated and unrestricted.

Thus, the GPLv2 protects users' fair and unregulated use rights precisely by not attempting to cover them. Furthermore, the GPLv2 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 GPLv2 §0.

The bulk of GPLv2 §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" or "combined work". In fact, it would be incorrect and problematic if the GPL attempted to define these terms. A copyright license, in fact, has no control over the rules of copyright themselves. Such rules are the domain of copyright law and the courts — not the licenses that utilize those systems.

Copyright law as a whole does not propose clear and straightforward guidelines for identifying the derivative and/or combined works of software. However, no copyright license — not even the GNU GPL — can be blamed for this. Legislators and court opinions must give us guidance in borderline cases. Meanwhile, lawyers will likely based their conclusions on the application of rules made in the context of literary or artistic copyright to the different context of computer programming and by analyzing the (somewhat limited) case law and guidance available from various sources. (Chapter 14.1 discusses this issue in depth.)

3.2 GPLv2 §1: Verbatim Copying

GPLv2 §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 GPLv2 §1. These are in some ways the most important part of the redistribution, which is why they are mentioned by name. 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 users might be surprised the software is GPL'd. Thus throughout the GPL, there are specific references to the importance of notifying others down the distribution chain that they have rights under GPL.

GPL disclaims all warranties that legally can be disclaimed (which is discussed later in sections 8.3 and 8.4). Users generally rarely expect their software comes with any warranties, since typically all EULAs and other Free Software licenses disclaim warranties too. However, since many local laws require "consipicous" warranty disclaimers, GPLv2 §1 explicitly mentions the importance of keeping warranty disclaimers in tact upon redistribution.

Note finally that GPLv2 §1 creates groundwork for the important defense of commercial freedom. GPLv2 §1 clearly states that in the case of verbatim copies, one may make money. Re-distributors are fully permitted to charge for the re-distribution 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 12.2 for more discussion on making a profit from Free Software redistribution.)

CHAPTER 4

DERIVATIVE WORKS: STATUTE AND CASE LAW

As described in the earlier general discussion of copyleft, strong copyleft licenses such as the GPL seek to uphold software freedom via the copyright system. This principle often causes theoretical or speculative dispute among lawyers, because "the work" — the primary unit of consideration under most copyright rules – is not a unit of computer programming. In order to determine whether a "routine" an "object", a "function", a "library" or any other unit of software is part of one "work" when combined with other GPL'd code, we must ask a question that copyright law will not directly answer in the same technical terms.

Therefore, this chapter digresses from discussion of GPL's exact text to consider the matter of combined and/or derivative works — a concept that we must understand fully before considering GPLv2 §§2–3. At least under USA copyright law, The 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 GPLv2 §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.

4.1 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.

4.2 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); 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.

4.2.1 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.

4.2.2 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, constants, 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.

4.2.3 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.

4.3 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."

4.4 No Protection for "Methods of Operation"

The First Circuit has taken the position that the AFC test is inapplicable when the works in question relate to unprotectable elements set forth in § 102(b). Their approach results in a much narrower definition of derivative work for software in comparison to other circuits. Specifically, 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 Int'l., Inc., 49 F.3d 807 (1st Cir. 1995). In Lotus, the court held that a menu command hierarchy for a computer program was uncopyrightable because it did not merely explain and present the program's 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 Circuit's 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. As a result, courts in the First Circuit that apply the AFC test do so only after applying a broad interpretation of § 102(b) to filter out unprotected elements. E.g., Real View, LLC v. 20-20 Technologies, Inc., 683 F. Supp.2d 147, 154 (D. Mass. 2010).

4.5 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 among those circuits that have a standard for defining a software derivative work.

4.6 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 having 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.

In the case of Oracle America v. Google, 872 F. Supp.2d 974 (N.D. Cal. 2012), the Northern District of California District Court examined the question of whether the application program interfaces (APIs) associated with the Java programming language are entitled to copyright protection. While the court expressly declined to rule whether all APIs are free to use without license (872 F. Supp.2d 974 at 1002), the court held that the command structure and taxonomy of the APIs were not protectable under copyright law. Specifically, the court characterized the command structure and taxonomy as both a "method of operation" (using an approach not dissimilar to the First Circuit's analysis in Lotus) and a "functional requirement for compatibility" (using Sega v. Accolade, 977 F.2d 1510 (9th Cir. 1992) and Sony Computer Ent. v. Connectix, 203 F.3d 596 (9th Cir. 2000) as analogies), and thus unprotectable subject matter under § 102(b).

Perhaps not surprisingly, there have been few other 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 over-aggressiveness on the part of the plaintiff.

4.7 How Much Do Derivative Works Matter?

It is certainly true that GPL intends for any work that is determined a "derivative work" under copyright law must be licensed as a whole under GPL, as will be discussed in the following chapter. However, as we finish up our discussion derivative works, we must note that preparation of a derivative work is by far not the only way to create a new work covered by GPL.

In fact, while derivative work preparation is perhaps the most exciting area of legal issues to consider, the more mundane ways to create a new work covered by GPL are much more common. For example, copyright statutes generally require permission from the copyright holder to grant explicit permission to modify a work in any manner. As discussed in the next chapter, the GPL *does* grant such permission, but requires the modified work must also be licensed under the terms of the GPL (and only GPL: see§ 7.3 in this

tutorial). Determining whether software was modified is a substantially easier analysis than the derivative work discussions and considerations in this chapter.

The question of derivative works, when and how they are made, is undoubtedly an essential discussion in the interpretation and consideration of copyleft. That is why this chapter was included in this tutorial. However, as we return from this digression and resume discussion of the detailed text of the GPLv2, we must gain a sense of perspective: most GPL questions center around questions of modification and distribution, not preparation of derivative works. Derivative work preparation is ultimately a small subset of the types of modified versions of the software a developer might create, thus, while an excessive focus on derivative works indulges us in the more exciting areas of copyleft, we must keep a sense of perspective regarding their relative importance.

CHAPTER 5

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, GPLv2 §§2–3, define the central core rights and requirements of GPLv2.

5.1 GPLv2 §2: Share and Share Alike

For many, this is where the "magic" happens that defends software freedom upon redistribution. GPLv2 §2 is the only place in GPLv2 that governs the modification controls of copyright law. If users distribute modified versions a GPLv2'd program, they must follow the terms of GPLv2 §2 in making those changes. Thus, this sections ensures that the body of GPL'd software, as it continues and develops, remains Free as in freedom.

To achieve that goal, GPLv2 §2 first sets forth that the rights of redistribution of modified versions are the same as those for verbatim copying, as presented in GPLv2 §1. Therefore, the details of charging money, keeping copyright notices intact, and other GPLv2 §1 provisions are intact here as well. However, there are three additional requirements.

5.1.1 The Simpler Parts of GPLv2 §2

The first (GPLv2 §2(a)) requires that modified files carry "prominent notices" explaining what changes were made and the date of such changes. This section does not prescribe some specific way of marking changes nor does it control the process of how changes are made. Primarily, GPLv2 §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, GPLv2 §2(a) serves an academic purpose — ensuring that future developers can use a diachronic approach to understand the software.

GPLv2 §2(c), a relatively simple section, requires that any program which (before modification) "normally reads commands interactively when run" and displays or prints legal information also display all copyright notices, warranty disclaimer, modification indications and a pointer to the license, even in modified versions. The requirement is relatively simple, and relates to an important policy goal of copyleft: downstream users should be informed of their rights. Its implications and details are straightforward and simple.

5.1.2 GPLv2 §2(b)

Meanwhile, GPLv2 §2(b) requires careful and extensive study. Its four short lines embody the some of the essential legal details of "share and share alike". These 46 words are considered by some to be the most worthy of careful scrutiny because they can be a source of great confusion when not properly understood.

In considering GPLv2 §2(b), first note the qualifier: it *only* applies to derivative, combined and/or modified works that "you distribute or publish". Despite years of education efforts on this matter, many still believe that modifiers of GPL'd software *must* publish or otherwise share their changes. On the contrary, GPLv2 §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 GPLv2 §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 copyright law. If, under copyright law, the modified version "contains or is derived from" the GPL'd software, then the requirements of GPLv2 §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 GPLv2 §2(b) describes what the licensee must do if she distributes or publishes a modified version of the work — 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.

The work "as a whole" is what is to be licensed. This is an important point that GPLv2 §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 combined and/or 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, GPLv2 §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 proprietary product — would its copyright holders give you permission to create and distribute $\mathcal{G}$+$\mathcal{I}$ without paying them a hefty sum?) The license of $\mathcal{G}$, the GPL, states the options for the copyright holder of $\mathcal{I}$ who may want to create and distribute $\mathcal{G}$+$\mathcal{I}$. The GPL's pre-granted permission to create and distribute combined and/or derivative works, provided the terms of the 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 condition is any way unreasonable is simply ludicrous.

The GPL recognizes what is outside its scope. When a programmer's work is "separate and independent" from any GPL'd program code with which it could be combined, then the obligations of copyleft do not extend to the work separately distributed. Thus, Far from attempting to extend copyleft beyond the scope of copyright, the licenses explicitly recognize.

Thus, GPL recognizes what is outside its scope. When a programmer's work is "separate and independent" from any GPL'd program code with which it could be combined, then copyleft obligations do not

[1] Most Free Software enthusiasts believe there is a **moral** obligation to redistribute changes that are generally useful, and they often encourage companies and individuals to do so. However, there is a clear distinction between what one **ought** to do and what one **must** do.

extend to the independent work separately distributed. Thus, far from attempting to extend copyleft beyond the scope of copyright, GPL explicitly limits the scope of copyleft to the scope of copyright.

GPL does not, however (as is sometimes suggested) distinguish "dynamic" from "static" linking of program code. It is occasionally suggested that a subroutine "dynamically" linked to GPL'd code is, by virtue of the linking alone, inherently outside the scope of copyleft on the main work. This is a misunderstanding. When two software components are joined together to make one work (whether a main and some library subroutines, two objects with their respective methods, or a program and a "plugin") the combination infringes the copyright on the components if the combination required copyright permission from the component copyright holders, as such permission was either not available or was available on terms that were not observed.

In other words, when combining other software with GPL'd components, the only available permission is GPL. The combiner must observe and respect the GPL observed on the combination as a whole. It matters not if that combination is made with a linker before distribution of the executable, is made by the operating system in order to share libraries for execution efficiency at runtime, or results from runtime references in the language at runtime (as in Java programs).

The next phrase of note in GPLv2 §2(b) is "licensed ... at no charge." This phrase confuses many. The sloppy reader points out this as "a contradiction in GPL" because (in their confused view) that clause of GPLv2 §2 says that re-distributors cannot charge for modified versions of GPL'd software, but GPLv2 §1 says that they can. Avoid this confusion: the "at no charge" **does not** prohibit re-distributors 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 contract or the act of transferring the copy to the customer, but they may not charge a separate licensing fee for the software.

GPLv2 §2(b) further states that the software must "be licensed ... to all third parties." This too yields some confusion, and feeds the misconception mentioned earlier — that all modified versions must be 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 GPLv2 §2(b) does require redistributors to license the whole work in a way that extends to all third parties who may ultimately receive a copy of the software.

In summary, GPLv2 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 GPLv2. When an entity *chooses* to redistribute a work based on GPL'd software, the license of that whole work must be GPL and only GPL. In this manner, GPLv2 §2(b) dovetails nicely with GPLv2 §6 (as discussed in Section 7.3 of this tutorial).

The final paragraph of GPLv2 §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. A programmer must expend actual effort 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.

5.1.3 Right to Private Modification

The issue of private modifications of GPLv2'd works deserves special attention. While these rights are clearly explicit in GPLv3 §2¶2 (see § 9.4 of this tutorial for details), the permission to create private modifications is mostly implicit in GPLv2. Most notably, the requirements of GPLv2 §2 (and GPLv2 §3, which will be discussed next) are centered around two different copyright controls: both modification *and* distribution. As

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

such, GPLv2 §2's requirements need only be met when a modified version is distributed; one need not follow them for modified versions that are not distributed.[3]

However, the careful reader of GPLv2 will notice that, unlike GPLv3, no other clauses of the license actually give explicit permission to make private modifications. Since modification of software is a control governed by copyright, a modifier needs permission from the copyright holder to engage in that activity.

In practice, however, traditional GPLv2 interpretation has always assumed that blanket permission to create non-distributed modified versions was available, and the FSF has long opined that distribution of modified versions is never mandatory. This issue is one of many where GPLv3 clarifies in explicit text the implicit policy and intent that was solidified via long-standing interpretation of GPLv2.

5.2 GPLv2 §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 be 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 modified versions (and/or derivative works) of the source code. Applying a systematic process (i.e., "compilation"[4]) 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 modified 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 modified works of the software's source code. GPLv2 §3 addresses the matter of creation and distribution of binary versions.

Under GPLv2 §3, binary versions may be created and distributed under the terms of GPLv2 §1–2, so all the material previously discussed applies here. However, GPLv2 §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. GPLv2 §3 must ensure that the binaries are never distributed without the source code, so that these freedoms are passed through the distribution chain.

GPLv2 §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 GPLv2 §3(a).

GPLv2 §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).

5.2.1 Complete, Corresponding Source (CCS)

Under GPLv2 §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". GPLv2 §3's penultimate paragraph explains in detail what is meant by "complete". In essence, it is all the material that a programmer of average skill would

[3] As a matter of best practice, it's useful to assume that all software may eventually be distributed later, even if there no plans for distribution at this time. Too often, GPL violations occur because of a late distribution decision of software that was otherwise never intended for distribution.

[4] "Compilation" in this context refers to the automated computing process of converting source code into binaries. It has absolutely nothing to do with the term "compilation" in copyright statues.

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.

Based on the appearance of those two words, GPL theorists will often refer to the source code required under the previsions of this section as "Complete, Corresponding Source", sometimes abbreviated as CCS. CCS is not a formal, defined term in GPLv2, but rather, GPL theorists coined the acronym CCS to embody not just the concepts of "complete" and "corresponding" as found in GPLv2, but the entirety of GPLv2's requirements for source code provisioning. In other words, GPL theorists might say: "the company provided some source, but it wasn't CCS", which would mean the source code failed in some ways to meet some term of GPLv2.

Indeed, CCS needs completely include not just that source which is directly translated by the compiler into object code, but other materials necessary to convert the source into equivalent binaries. Specifically, GPLv2 §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 modified works from the sources provided.

This requirement is not merely of theoretical value. If you pay a high price for a copy of GPL'd binaries (which comes with CCS, 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. Such distributions violate GPL, since the downstream users cannot effectively "control compilation and installation" of the binaries.

5.2.2 Additional Source Provision Options

Software distribution comes in many forms. Embedded manufacturers, for example, have the freedom to put GPL'd software into mobile devices 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 unfeasible.

GPLv2 §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 — the phrase seeks to pre-adapt to changes in technology. When GPLv2 was first published in June 1991, distribution on magnetic tape was still common, and CD was relatively new. By 2002, CD was the default. By 2007, DVD's were the default. Now, it's common to give software on USB drives and SD cards. This language in the license 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 in industrialized countries, it is indeed customary to simply download a CDs' worth of data quickly. However, even today in the USA, many computer users are not connected to the Internet, and most people connected to the Internet still have limited download speeds. 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 are 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 GPLv2 §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 GPLv2 §3.

As is shown above, under GPLv2 §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, GPLv2§ 3(b) is available.

GPLv2 §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 GPLv2 §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, GPLv2 §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 GPLv2 §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 GPLv2 §§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.

GPLv2 §3(c) is created to save her some trouble, because by itself GPLv2 §3(b) would unfairly favor large companies. GPLv2 §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 re-distributors, 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, GPLv2 §3(b)'s compromise primarily benefits companies that distribute binary software commercially. Without GPLv2 §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 GPLv2 §3(b)-compliant offer, would be required under GPLv2 (sans GPLv2 §3(c)) to acquire the source, merely to give a copy of the software to a friend who needed it. GPLv2 §3(c) reshifts this burden to entity who benefits from GPLv2 §3(b).

GPLv2 §3(c) allows those who undertake *noncommercial* distribution to simply pass along a GPLv2 §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 GPLv2 §3(a), or write her own GPLv2 §3(b)-compliant source offer.

This process is precisely the reason why a GPLv2 §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 GPLv2 §3(b) must thus be prepared to honor all incoming source code requests. For this and the many other additional necessary complications under GPLv2 §§3(b–c), it is only rarely a better option than complying via GPLv2 §3(a).

CHAPTER 6

GPL'S IMPLIED PATENT GRANT

We digress again briefly from our section-by-section consideration of GPLv2 to consider the interaction between the terms of GPL and patent law. The GPLv2, 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, 1451 (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 GPLv2 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 GPLv2, the distributor impliedly licenses those patents to the GPLv2 licensee with respect to the GPLv2'd software.

An interesting issue regarding this implied patent license of GPLv2'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 GPLv2 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 other Free Software licenses are specifically limited to the patent claims covered by the code as licensed by the patentee.

In contrast, a GPLv2 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 modified 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 among 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 GPLv2 ensures that the community mutually benefits from the licensing of patents to any single community member.

Note that simply because GPLv2'd software has an implied patent license does not mean that any patents held by a distributor of GPLv2'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 GPLv2 by the patent holder, and

(b) any party that does not comply with the GPLv2 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 program under the GPLv2, then it cannot assert the patent against any party based on that party's use of Company $\mathcal{A}$'s GPL'd Web browsing software program, or on that party's creation and use of modified versions of that GPL'd program. However, if a party uses that program without complying with the GPLv2, then Company $\mathcal{A}$ can assert both copyright infringement claims against the non-GPLv2-compliant party and infringement of the patent, because the implied patent license only extends to use of the software in accordance with the GPLv2. Further, if Company $\mathcal{B}$ distributes a competitive advanced Web browsing program that is not a modified version of Company $\mathcal{A}$'s GPL'd Web browsing software 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 also distributed under the GPLv2, 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 GPLv2 implied patent license, as only those competitors who adopt and comply with the GPLv2'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 GPLv2. 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 GPLv2, 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 7

DEFENDING FREEDOM ON MANY FRONTS

Chapters 3 and 5 presented the core freedom-defending provisions of GPLv2, which are in GPLv2 §§0–3. GPLv2§§ 4–7 of the GPLv2 are designed to ensure that GPLv2 §§0–3 are not infringed, are enforceable, are kept to the confines of copyright law but also not trumped by other copyright agreements or components of other entirely separate legal systems. In short, while GPLv2 §§0–3 are the parts of the license that defend the freedoms of users and programmers, GPLv2 §§4–7 are the parts of the license that keep the playing field clear so that §§ 0–3 can do their jobs.

7.1 GPLv2 §4: Termination on Violation

GPLv2 §4 is GPLv2'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. (See 7.4 for expanded discussion of GPLv2 irrevocability.)

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 holders have the right to relicense the same work under different licenses (see Section 12.2 of this tutorial), or to stop distributing the GPLv2'd version (assuming GPLv2 §3(b) was never used), but they may not revoke the rights under GPLv2 already granted.

In fact, when an entity loses 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 decide when GPLv2 §4 termination occurs (if ever); rather, the actions of the licensee determine that.

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. GPLv2 §4 says that if you undertake any of those other types (e.g., redistributing binary-only in violation of GPLv2 §3), then all rights under the license — even those otherwise permitted for those who have not violated — terminate automatically.

GPLv2 §4 makes GPLv2 enforceable. If licensees fail to adhere to the license, then they are stuck without any permission under to engage in activities covered by copyright law. They must completely cease and desist from all copying, modification and distribution of the GPL'd software.

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

At that point, violating licensees must gain the forgiveness of the copyright holders to have their rights restored. Alternatively, the violators could negotiate another agreement, separate from GPL, with the copyright holder. Both are common practice, although Chapter 13.3 explains further key differences between these two very different uses of GPL.

7.2 GPLv2 §5: Acceptance, Copyright Style

GPLv2 §5 brings us to perhaps the most fundamental misconception and common confusion about GPLv2. 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." Usually, no rights are transferred between parties. By contrast, the GPL is primarily a permission slip to undertake activities that would otherwise have been prohibited by copyright law. As such, GPL 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 GPLv2 §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.

While GPL is by default a copyright license, it is certainly still possible to consider GPL as a contract as well. For example, some distributors chose to "wrap" their software in an acceptance ceremony to the GPL, and nothing in the GPL prohibits that use. Furthermore, the ruling in *Jacobsen v. Katzer, 535 F.3d 1373, 1380 (Fed.Cir.2008)* indicates that **both** copyright and contractual remedies may be sought by a copyright holder seeking to enforce a license designed to uphold software freedom.

7.3 GPLv2 §6: GPL, My One and Only

A point that was glossed over in Section 7.1's discussion of GPLv2 §4 was the irrevocable nature of the GPL. The GPLv2 is indeed irrevocable, and it is made so formally by GPLv2 §6.

The first sentence in GPLv2 §6 ensures that as software propagates down the distribution chain, that each licensor can pass along the license to each new licensee. Under GPLv2 §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 GPLv2. 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 GPLv2 §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 GPLv2.

The final sentence of GPLv2 §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 GPLv2 §3(c), as they cannot be assured that the issuer of the offer will honor their GPLv2 §3 obligations.

In short, GPLv2 §6 says that your license for the software is your one and only copyright license allowing you to copy, modify and distribute the software.

GPLv2 §6 is GPLv2's "automatic downstream licensing" provision[2]. Each time you redistribute a GPL'd

[2]This section was substantially expanded for clarity and detail in GPLv3 §10.

program, the recipient automatically receives a license from each original licensor to copy, distribute or modify the program subject to the conditions of the license. The redistributor need not take any to ensure the downstream recipient's acceptance of the license terms. This places every copyright holder in the chain of descent of the code in legal privity, or direct relationship, with every downstream redistributor. Two legal effects follow. First, downstream parties who remain in compliance have valid permissions for all actions (including modification and redistribution) even if their immediate upstream supplier of the software has been terminated for license violation[3]. Downstream's licensed rights are not dependent on compliance of their upstream, because their licenses issue directly from the copyright holder. Second, automatic termination cannot be cured by obtaining additional copies from an alternate supplier: the license permissions emanate only from the original licensors, and if they have automatically terminated permission, no act by any intermediate license holder can restore those terminated rights[4].

7.4 GPLv2 Irrevocability

This section digresses briefly to examine the manner in which GPLv2§§ 4–6 interact together to assure that the license grant is irrevocable. There are two legal theories why a contributor cannot terminate their license grant. First is an argument that the text of the GPL prevents it; second is that a contributor would be estopped from succeeding on an infringement claim for continued use of the code even if it wasn't removed.

7.4.1 The text of the GPLv2

The GPLv2 have several provisions that, when taken together, can be construed as an irrevocable license from each contributor. First, the GPLv2 says "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" (GPLv2§5, emphasis added). A contributor by definition is modifying the code and therefore has agreed to all the terms in the GPLv2, which includes the web of mechanisms in the GPLv2 that ensure the code can be used by all.

More specifically, the downstream license grant says "the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions." (GPLv2§6). So in this step, the contributor has granted a license to the downstream, on the condition that the downstream complies with the license terms.

That license granted to downstream is irrevocable, again provided that the downstream user complies with the license terms: "[P]arties 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" (GPLv2§4).

Thus, anyone downstream of the contributor (which is anyone using the contributor's code), has an irrevocable license from the contributor. A contributor may claim to revoke their grant, and subsequently sue for copyright infringement, but a court would likely find the revocation was ineffective and the downstream user had a valid license defense to a claim of infringement.

Nevertheless, for purposes of argument, we will assume that for some reason the GPLv2 is not enforceable against the contributor[5], or that the irrevocable license can be revoked[6]. In that case, the application

[3] While this is legally true, as a practical matter, a failure of "complete, corresponding source" (CCS) provisioning by an upstream could make it effectively impossible for a downstream party to engage in a commercial redistribution pursuant to GPLv2 §3(a–b). (§ 18.2 in the Compliance Guide portion of this tutorial discussed related details.)

[4] While nearly all attorneys and copyleft theorists are in agreement on this point, German copyleft legal expert Till Jaeger vehemently disagrees. Jaeger's position is as follows: under German copyright law, a new copy of GPL'd software is a "fresh" license under GPL, and if compliance continues from that point further, the violator's permissions under copyright law are automatically restored, notwithstanding the strict termination provision in GPLv2 §4. However, in practice, this issue is only salient with regard to proprietary relicensing business models, since other copyright holders typically formally restore distributions rights once the only remaining compliance issue is "you lost copyright permission due to GPLv2 §4". Therefore, the heated debates, which have raged between Jaeger and almost everyone else in the copyleft community for nearly a decade, regard an almost moot and wholly esoteric legal detail.

[5] For example, the argument has been made that there may be a failure of consideration on the part of the contributor. While *Jacobsen v. Katzer*, 535 F.3d 1373 (Fed. Cir. 2008) is accepted as holding that there is consideration received by the contributor in a FOSS license, the posture of the case was one where the contributor advocated for the theory, not against it. The author is not aware of any other decisions that have analyzed the question in any depth, so it perhaps could be challenged in the right factual situation.

[6] A contract without a definable duration can be terminated on reasonable notice. *Great W. Distillery Prod. v. John A.*

of promissory estoppel will likely mean that the contributor still cannot enforce their copyright against downstream users.

7.4.2 Promissory estoppel

"Promissory estoppel" is a legal theory that says, under some circumstances, a promise is enforceable against the promisee even after the promisee tries to renege on the promise. The test for how and when promissory estoppel applies differs from state to state, but generally where there is a "promise which the promisor should reasonably expect to induce action or forbearance on the part of the promisee or a third person and which does induce such action or forbearance is binding if injustice can be avoided only by enforcement of the promise."[7] Breaking it down, it is:

1. where there is a clear and definite promise;
2. where the promisor has a reasonable expectation that the offer will induce action or forbearance on the part of the promisee;
3. which does induce actual and reasonable action or forbearance by the promisee; and
4. which causes a detriment which can only be avoided by the enforcement of the promise.

In this case, the promisor is the contributor. This should be an easy standard to meet in any widely used software.

1. The promise is contained in the GPL, which is a promise that one can continue to use the licensed software as long as the terms of the license are met.
2. A contributor knows that there is a broad user base and users consume the software relying on the grant in the GPL as assuring their continued ability to use the software (one might even say it is the *sine qua non* of the intent of the GPL).
3. Users do, in fact, rely on the promises in the GPL, as they ingest the software and base their businesses on their continued ability to use the software.
4. Whether the user will suffer detriment is case-specific, but using Linux, a software program that is often fundamental to the operation of a business, as an example, the loss of its use would have a significantly detrimental, perhaps even fatal, effect on the continued operation of the business.

7.4.3 Conclusion

Whether as a matter of a straightforward contractual obligation, or as a matter of promissory estoppel, a contributor's attempt to revoke a copyright license grant and then enforce their copyright against a user is highly unlikely to succeed.

7.5 GPLv2 §7: "Give Software Liberty or Give It Death!"

In essence, GPLv2 §7 is a verbosely worded way of saying for non-copyright systems what GPLv2 §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

Wathen Distillery Co., 10 Cal. 2d 442, 447, 74 P.2d 745, 747 (1937). The term nevertheless can be a term of indefinite length where its continuing effect is tied to the conduct of the parties. *Id.*

[7] *Kajima/Ray Wilson v. Los Angeles Cty. Metro. Transp. Auth.*, 23 Cal. 4th 305, 310, 1 P.3d 63, 66 (2000), *citing* Restatement (Second) of Contracts §90(1) (1979).

their patents, or grant a royalty-free, irrevocable, non-exclusive license to those patents. Many companies have chosen the latter.

Thus, GPLv2 §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.

7.6 GPLv2 §8: Excluding Problematic Jurisdictions

GPLv2 §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 GPLv2'd software can continue in free and unabated distribution in the countries where such controls do not exist.

As far as is currently known, GPLv2 §8 has very rarely been formally used by copyright holders. Admittedly, some have used GPLv2 §8 to explain various odd special topics of distribution (usually related in some way to GPLv2 §7). However, generally speaking, this section is not proven particularly useful in the more than two decades of GPLv2 history.

Meanwhile, despite many calls by the FSF (and others) for those licensors who explicitly use this section to come forward and explain their reasoning, no one ever did. Furthermore, research conducted during the GPLv3 drafting process found exactly one licensor who had invoked this section to add an explicit geographical distribution limitation, and the reasoning for that one invocation was not fitting with FSF's intended spirit of GPLv2 §8. As such, GPLv2 §8 was not included at all in GPLv3.

CHAPTER 8

ODDS, ENDS, AND ABSOLUTELY NO WARRANTY

GPLv2 §§0–7 constitute the freedom-defending terms of the GPLv2. The remainder of the GPLv2 handles administrivia and issues concerning warranties and liability.

8.1 GPLv2 §9: FSF as Stewards of GPL

FSF reserves the exclusive right to publish future versions of the GPL; GPLv2 §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.

(Chapter 2 discusses more about the various versions of GPL.)

8.2 GPLv2 §10: Relicensing Permitted

GPLv2 §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.

8.3 GPLv2 §11: No Warranty

Most warranty disclaimer language shouts at you. The Uniform Commercial Code §2-316, which most of the USA's states and commonwealths have adopted as their local law, allows disclaimers of warranty, provided that the disclaimer is "conspicuous". There is apparently general acceptance that ALL CAPS is the preferred way to make something conspicuous, and that has over decades worked its way into the voodoo tradition of warranty disclaimer writing.

That said, there is admittedly some authority under USA law suggesting that conspicuousness can be established by capitalization and is absent when a disclaimer has the same typeface as the terms surrounding it (see *Stevenson v. TRW, Inc.*, 987 F.2d 288, 296 (5th Cir. 1993)). While GPLv3's drafters doubted that such authority would apply to copyright licenses like the GPL, the FSF has nevertheless left warranty and related disclaimers in ALL CAPS throughout all versions of GPL.[1]

[1]One of the authors of this tutorial, Bradley M. Kuhn, has often suggested the aesthetically preferable compromise of a SPECIFICALLY DESIGNED "SMALL CAPS" FONT, SUCH AS THIS ONE, AS AN ALTERNATIVE TO WRITING IN ALL CAPS IN THE DEFAULT FONT (LIKE THIS), since the latter adds more ugliness than conspicuousness. Kuhn once engaged in reversion war

Critics have occasionally questioned GPL's enforceability in some jurisdictions because its disclaimer of warranties is impermissibly broad. However, critics have generally failed to articulate specific precedents in their jurisdictions that would directly indicate a problem with GPL's warranty disclaimer. Meanwhile, Article 35 of the United Nations Convention on Contracts for the International Sale of Goods (often abbreviated "CISG", which many countries have adopted) permits the disclaimer of warranties, so jurisdictions adopting this treaty allow some form of warranty disclaimer[2]. Nevertheless, to account for possible jurisdictional variances regarding this or any other issue, GPLv2 §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 GPLv2 §11 is that GPLv2 §1 permits the sale of warranty as an additional service, which GPLv2 §11 affirms.

8.4 GPLv2 §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 GPLv2 §12. GPLv2 §11 thus gets rid of all implied warranties that can legally be disavowed. GPLv2 §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, GPLv2 §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 GPLv2 §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.

with a lawyer who disagreed, but that lawyer never answered Kuhn's requests for case law that argues THIS IS INHERENTLY MORE CONSPICUOUS Than this is.

[2]Scholars continue to debate to what extent CISG applies to software licenses. For example, Diedrich concluded that "CISG is prima facie applicable to international transactions involving the transfer of computer software for a price", but Sono disagrees with this "prevailing view", presenting an "analysis [that] restricts the applicability of the CISG to software transactions by excluding 'license contracts"'. (See Frank Diedrich, *The CISG and Computer Software Revisited*, 6 Vindobona Journal of International Commercial Law and Arbitration, Supplement 55–75 (2002), and Hiroo Sono, *The Applicability and Non-Applicability of the CISG to Software Transactions*, Camilla B. Andersen & Ulrich G. Schroeter eds., Sharing International Commercial Law across National Boundaries: Festschrift for Albert H. Kritzer on the Occasion of his Eightieth Birthday, Wildy, Simmonds & Hill Publishing (2008) 512–526.)

CHAPTER 9

GPL VERSION 3

This chapter discusses the text of GPLv3. Much of this material herein includes text that was adapted (with permission) from text that FSF originally published as part of the so-called "rationale documents" for the various discussion drafts of GPLv3.

The FSF ran a somewhat public process to develop GPLv3, and it was the first attempt of its kind to develop a Free Software license this way. Ultimately, RMS was the primary author of GPLv3, but he listened to feedback from all sorts of individuals and even for-profit companies. Nevertheless, in attempting to understand GPLv3 after the fact, the materials available from the GPLv3 process have a somewhat "drinking from the firehose" effect. This chapter seeks to explain GPLv3 to newcomers, who perhaps are familiar with GPLv2 and who did not participate in the GPLv3 process.

Those who wish to drink from the firehose and take a diachronic approach to GPLv3 study by reading the step-by-step public drafting process of the GPLv3 (which occurred from Monday 16 January 2006 through Monday 19 November 2007) should visit `http://gplv3.fsf.org/`.

9.1 Understanding GPLv3 As An Upgraded GPLv2

Ultimately, GPLv2 and GPLv3 co-exist as active licenses in regular use. As discussed in Chapter 2, GPLv1 was never regularly used alongside GPLv2. However, given GPLv2's widespread popularity and existing longevity by the time GPLv3 was published, it is not surprising that some licensors still prefer GPLv2-only or GPLv2-or-later. GPLv3 gained major adoption by many projects, old and new, but many projects have not upgraded due to (in some cases) mere laziness and (in other cases) policy preference for some of GPLv2's terms and/or policy opposition to GPLv3's terms.

Given this "two GPLs world" is reality, it makes sense to consider GPLv3 in terms of how it differs from GPLv2. Also, most of the best GPL experts in the world must deal regularly with both licenses, and admittedly have decades of experience with GPLv2 while the most experience with GPLv3 that's possible is by definition less than a decade. These two factors usually cause even new students of GPL to start with GPLv2 and move on to GPLv3, and this tutorial follows that pattern.

Overall, the changes made in GPLv3 admittedly *increased* the complexity of the license. The FSF stated at the start of the GPLv3 process that they would have liked to oblige those who have asked for a simpler and shorter GPL. Ultimately, the FSF gave priority to making GPLv3 a better copyleft license in the spirit of past GPL's. Obsession for concision should never trump software freedom.

The FSF had many different, important goals in seeking to upgrade to GPLv3. However, one important goal that is often lost in the discussion of policy minutia is a rather simple but important issue. Namely, FSF sought to assure that GPLv3 was more easily internationalized than GPLv2. In particular, the FSF sought to ease interpretation of GPL in other countries by replacement of USA-centric[1] copyright phrases

[1]See Section 1.2.4 of this tutorial for a brief discussion about non-USA copyright systems.

and wording with neutral terminology rooted in description of behavior rather than specific statute. As can be seen in the section-by-section discussion of GPLv3 that follows, nearly every section had changes related to issues of internationalization.

9.2 GPLv3 §0: Giving In On "Defined Terms"

One of lawyers' most common complaints about GPLv2 is that defined terms in the document appear throughout. Most licenses define terms up-front. However, the GPL was always designed both as a document that should be easily understood both by lawyers and by software developers: it is a document designed to give freedom to software developers and users, and therefore it should be comprehensible to that constituency.

Interestingly enough, one coauthor of this tutorial who is both a lawyer and a developer pointed out that in law school, she understood defined terms more quickly than other law students precisely because of her programming background. For developers, having `#define` (in the C programming language) or other types of constants and/or macros that automatically expand in the place where they are used is second nature. As such, adding a defined terms section was not terribly problematic for developers, and thus GPLv3 adds one. Most of these defined terms are somewhat straightforward and bring forward better worded definitions from GPLv2. Herein, this tutorial discusses a few of the new ones.

GPLv3 §0 includes definitions of five new terms not found in any form in GPLv2: "modify" "covered work", "propagate", "convey", and "Appropriate Legal Notices".

9.2.1 Modify and the Work Based on the Program

GPLv2 included a defined term, "work based on the Program", but also used the term "modify" and "based on" throughout the license. GPLv2's "work based on the Program" definition made use of a legal term of art, "derivative work", which is peculiar to USA copyright law.[2] GPLv2 always sought to cover all rights governed by relevant copyright law, in the USA and elsewhere. Even though differently-labeled concepts corresponding to the derivative work are recognized in all copyright law systems, these counterpart concepts might differ to some degree in scope and breadth from the USA derivative work. GPLv3 therefore internationalizes on this issue by removing GPLv2's references to derivative works and by providing a more globally useful definition. GPLv3 drops all reference to USA "derivative works" and returns to the base concept only: GPL covers the licensed work and all works where copyright permission from the licensed work's copyright holder.

The new definitions returns to the common elements of copyright law. Copyright holders of works of software have the exclusive right to form new works by modification of the original — a right that may be expressed in various ways in different legal systems. GPLv3 operates to grant this right to successive generations of users (particularly through the copyleft conditions set forth in GPLv3 §5, as described later in this tutorial in its § 9.8). Here in GPLv3 §0, "modify" refers to basic copyright rights, and then this definition of "modify" is used to define "modified version of" and "work based on" as synonyms.

9.2.2 The Covered Work

GPLv3 uses a common license drafting technique of building upon simpler definitions to make complex ones. The Program is a defined term found throughout GPLv2, and the word "covered" and the phrase "covered by this license" are used in tandem with the Program in GPLv2, but not as part of a definition. GPLv3 offers a single term "covered work", which enables some of the wording in GPLv3 to be simpler and clearer than its GPLv2 counterparts.

Next, to avoid locking GPLv3 into specific copyright statues, the GPLv3 defines two terms that are otherwise exotic to the language of international copyright.

[2]Ironically, most criticism of USA-specific legal terminology in GPLv2's "work based on the Program" definition historically came not primarily from readers outside the USA, but from those within it. The FSF noted in that it did not generally agree with these views, and expressed puzzlement by the energy with which they were expressed, given the existence of many other, more difficult legal issues implicated by the GPL. Nevertheless, the FSF argued that it made sense to eliminate usage of local copyright terminology to good effect.

9.2.3 Propagate

To "propagate" a work covered by the license means any activity in a locale that requires permission of copyright holders in that locale's legal system. However, personal use or modification for personal use are activities explicitly excluded from "propagation" *regardless* of domestic copyright law.

The term "propagate" serves two purposes. First, "propagate" provides a simple and convenient means for distinguishing between the kinds of uses of a work that GPL imposes conditions on and the kinds of uses that GPL does not (for the most part) impose conditions on.

Second, "propagate" helps globalize GPL in its wording and effect: "derivative work" was in fact not the only term commonly used by local copyright statutes. A term like "distribute" (or its equivalent in languages other than English) is also used in several national copyright statutes. Practical experience with GPLv2 revealed the awkwardness of using the term "distribution" in a license intended for global use: the scope of "distribution" in the copyright context can differ from country to country. The GPL never necessarily intended the specific meaning of "distribution" that exists under USA (or any other country's) copyright law.

Indeed, even within a single country and language, the term distribution may be ambiguous; as a legal term of art, distribution varies significantly in meaning among those countries that recognize it. For example, comments during GPLv3's drafting process indicated that in at least one country, distribution may not include network transfers of software but may include interdepartmental transfers of physical copies within an organization. Meanwhile, the copyright laws of many countries, as well as certain international copyright treaties, recognize "making available to the public" or "communication to the public" as one of the exclusive rights of copyright holders.

Therefore, the GPLv3 defines the term "propagate" by reference to activities that require permission under "applicable copyright law", but excludes execution and private modification from the definition. GPLv3's definition also gives examples of activities that may be included within "propagation" but it also makes clear that, under the copyright laws of a given country, "propagation" may include other activities as well.

Thus, propagation is defined by behavior, and not by categories drawn from some particular national copyright statute. This helps not only with internationalization, but also factually-based terminology aids in developers' and users' understanding of the GPL.

As a further benefit, because "propagation" includes all exclusive rights granted under any particular copyright regime, the term automatically accounts for all exclusive rights under that regime.

9.2.4 Convey

Next, GPLv3 defines a subset of propagate — "convey". Conveying includes activities that constitute propagation of copies to others. As with the definition of propagate, GPLv3 thus addresses transfers of copies of software in behavioral rather than statutory terms. Any propagation that enables other parties to receive or make copies of the work, is called "conveying". Usually, conveying is the activity that triggers most of the other obligations of GPLv3.

9.2.5 Appropriate Legal Notices

GPLv2 used the term "appropriate copyright notice and disclaimer of warranty" in two places, which is a rather bulky term. Also, experience with GPLv2 and other licenses that grant software freedom showed throughout the 1990s that the scope of types of notices that need preservation upon conveyance were more broad that merely the copyright notices. The Appropriate Legal Notice definition consolidates the material that GPLv2 traditionally required preserved into one definition.

9.2.6 Other Defined Terms

Note finally that not all defined terms in GPLv3 appear in GPLv3 §0. Specifically, those defined terms that are confined in use to a single section are defined in the section in which they are used, and GPLv3 §1 contains those definitions focused on source code. In this tutorial, those defined terms are discussed in the section where they are defined and/or used.

9.3 GPLv3 §1: Understanding CCS

Ensuring that users have the source code to the software they receive and the freedom to modify remains the paramount right embodied in the Free Software Definition (found in § 1.1 of this tutorial). As such, GPLv3 §1 is likely one of the most important sections of GPLv3, as it contains all the defined terms related to this important software freedom.

9.3.1 Source Code Definition

First, GPLv3 §1 retains GPLv2's definition of "source code" and adds an explicit definition of "object code" as "any non-source version of a work". Object code is not restricted to a narrow technical meaning and is understood broadly to include any form of the work other than the preferred form for making modifications to it. Object code therefore includes any kind of transformed version of source code, such as bytecode or minified Javascript. The definition of object code also ensures that licensees cannot escape their obligations under the GPL by resorting to shrouded source or obfuscated programming.

9.3.2 CCS Definition

The definition of CCS,[3] or, as GPLv3 officially calls it, "Corresponding Source" in GPLv3 §1¶4 is possibly the most complex definition in the license.

The CCS definition is broad so as to protect users' exercise of their rights under the GPL. The definition includes particular examples to remove any doubt that they are to be considered CCS. GPLv3 seeks to make it completely clear that a licensee cannot avoid complying with the requirements of the GPL by dynamically linking a subprogram component to the original version of a program. The examples also clarify that the shared libraries and dynamically linked subprograms that are included in Corresponding Source are those that the work is "specifically" designed to require, which clarifies that they do not include libraries invoked by the work that can be readily substituted by other existing implementations. While copyleft advocates never doubted this was required under GPLv2's definition of CCS, GPLv3 makes it abundantly clear with an extra example.

The GPL, as always, seeks to ensure users are truly in a position to install and run their modified versions of the program; the CCS definition is designed to be expansive to ensure this software freedom. However, although the definition of CCS is expansive, it is not sufficient to protect users' freedoms in many circumstances. For example, a GPL'd program, or a modified version of such a program, might be locked-down and restricted. The requirements in GPLv3 §6 (discussed in Section 9.9 of this tutorial) handle that issue. (Early drafts of GPLv3 included those requirements in the definition of CCS; however, given that the lock-down issue only comes up in distribution of object code, it is more logical to place those requirements with the parts of GPLv3 dealing directly with object code distribution).

The penultimate paragraph in GPLv3§2 notes that GPLv3's CCS definition does not require source that can be automatically generated. Many code generators, preprocessors and take source code as input and sometimes even have output that is still source code. Source code should always be whatever the original programmer preferred to modify.

GPLv3§1's final paragraph removes any ambiguity about what should be done on source-only distributions. Specifically, the right to convey source code that does not compile, does not work, or otherwise is experimental in-progress work is fully permitted, *provided that* no object code form is conveyed as well. Indeed, when combined with the permissions in GPLv3§ 5, it is clear that if one conveys *only* source code, one can never be required to provide more than that. One always has the right to modify a source code work by deleting any part of it, and there can be no requirement that free software source code be a whole functioning program.

[3]Note that the preferred term for those who work regularly with both GPLv2 and GPLv3 is "Complete Corresponding Source", abbreviated to "CCS". Admittedly, the word "complete" no longer appears in GPLv3 (which uses the word "all" instead). However, both GPLv2 and the early drafts of GPLv3 itself used the word "complete", and early GPLv3 drafts even called this defined term "Complete Corresponding Source". Meanwhile, use of the acronym "CCS" (sometimes, "C&CS") was so widespread among GPL enforcers that its use continues even though GPLv3-focused experts tend to say just the defined term of "Corresponding Source".

9.3.3 The System Library Exception

The previous section skipped over one part of the CCS definition, the so-called system library exception. The "System Libraries" definition (and the "Standard Interface" and "Major Component" definitions, which it includes) are designed to permit certain distribution arrangements that are considered reasonable by copyleft advocates. The system library exception is designed to allow copylefted software to link with these libraries when prohibition of that linking would hurt software freedom more than it would hurt proprietary software.

The system library exception has two parts. Part (a) rewords the GPLv2 exception for clarity replacing GPLv2's words "unless that component itself accompanies the executable" with "which is not part of the Major Component". The goal here is to not require disclosure of source code of certain libraries, such as necessary Microsoft Windows DLLs (which aren't part of Windows' kernel but accompany it) that are required for functioning of copylefted programs compiled for Windows.

However, in isolation, (a) would be too permissive, as it would sometimes allow distributors to evade important GPL requirements. Part (b) reigns in (a). Specifically, (b) specifies only a few functionalities that a system library may provide and still qualify for the exception. The goal is to ensure system libraries are truly adjunct to a major essential operating system component, compiler, or interpreter. The more low-level the functionality provided by the library, the more likely it is to be qualified for this exception.

Admittedly, the system library exception is a frequently discussed topic of obsessed GPL theorists. The amount that has been written on the system library exception (both the GPLv2 and GPLv3 versions of it), if included herein, could easily increase this section of the tutorial to a length greater than all the others.

Like any exception to the copyleft requirements of GPL, would-be GPL violators frequently look to the system library exception as a potential software freedom circumvention technique. When considering whether or not a library qualifies for the system library exception, here is a pragmatic thesis to consider, based on the combined decades of experience in GPL interpretation of this tutorial's authors: the harder and more strained the reader must study and read the system library exception, the more likely it is that the library in question does not qualify for it.

9.4 GPLv3 §2: Basic Permissions

GPLv3 §2 can roughly be considered as an equivalent to GPLv2 §0 (discussed in § 3.1 of this tutorial). However, the usual style of improvements found in GPLv3 are found here as well. For example, the first sentence of GPLv3 §2 furthers the goal internationalization. Under the copyright laws of some countries, it may be necessary for a copyright license to include an explicit provision setting forth the duration of the rights being granted. In other countries, including the USA, such a provision is unnecessary but permissible.

GPLv3 §2¶1 also acknowledges that licensees under the GPL enjoy rights of copyright fair use, or the equivalent under applicable law. These rights are compatible with, and not in conflict with, the freedoms that the GPL seeks to protect, and the GPL cannot and should not restrict them.

However, note that (sadly to some copyleft advocates) the unlimited freedom to run is confined to the *unmodified* Program. This confinement is unfortunately necessary since Programs that do not qualify as a User Product in GPLv3 §6 (see § 9.9.2 in this tutorial) might have certain unfortunate restrictions on the freedom to run.[4]

GPLv3 §2¶2 distinguishes between activities of a licensee that are permitted without limitation and activities that trigger additional requirements. Specifically, GPLv3 §2¶2 guarantees the basic freedoms of privately modifying and running the program. While these basic freedoms were generally considered a standard part of users' rights under GPLv2 as well, the GPLv3 states them herein more explicitly. In other words, there is no direct analog to the first sentence of GPLv3 §2¶2 in GPLv2 (See § 5.1.3 of this tutorial for more on this issue.)

Also, GPLv3 §2¶2 gives an explicit permission for a client to provide a copy of its modified software to a contractor exclusively for that contractor to modify it further, or run it, on behalf of the client. However, the client can *only* exercise this control over its own copyrighted changes to the GPL-covered program. The parts of the program it obtained from other contributors must be provided to the contractor with the usual GPL freedoms. Thus, GPLv3 permits users to convey covered works to contractors operating exclusively on

[4] See § 1.1.1 of this tutorial for the details on "the freedom to run".

the users' behalf, under the users' direction and control, and to require the contractors to keep the users' copyrighted changes confidential, but *only if* the contractor is limited to acting on the users' behalf (just as the users' employees would have to act).

The strict conditions in this "contractors provision" are needed so that it cannot be twisted to fit other activities, such as making a program available to downstream users or customers. By making the limits on this provision very narrow, GPLv3 ensures that, in all other cases, contractor gets the full freedoms of the GPL that they deserve.

The FSF was specifically asked to add this "contractors provisions" by large enterprise users of Free Software, who often contract with non-employee developers, working offsite, to make modifications intended for the user's private or internal use, and often arrange with other companies to operate their data centers. Whether GPLv2 permits these activities is not clear and may depend on variations in copyright law in different jurisdictions. The practices seem basically harmless, so FSF decided to make it clear they are permitted.

GPLv3 §2's final paragraph includes an explicit prohibition of sublicensing. This provision ensures that GPL enforcement is always by the copyright holder. Usually, sublicensing is regarded as a practical convenience or necessity for the licensee, to avoid having to negotiate a license with each licensor in a chain of distribution. The GPL solves this problem in another way — through its automatic licensing provision found in GPLv3 §10 (which is discussed in more detail in § 9.13 of this tutorial).

9.5 GPLv3's views on DRM and Device Lock-Down

The issues of DRM, device lock-down and encryption key disclosure were the most hotly debated during the GPLv3 process. FSF's views on this were sadly frequently misunderstood and, comparing the provisions related to these issues in the earliest drafts of GPLv3 to the final version of GPLv3 shows the FSF's willingness to compromise on tactical issues to reach the larger goal of software freedom.

Specifically, GPLv3 introduced provisions that respond to the growing practice of distributing GPL-covered programs in devices that employ technical means to restrict users from installing and running modified versions. This practice thwarts the expectations of developers and users alike, because the right to modify is one of the core freedoms the GPL is designed to secure.

Technological measures to defeat users' rights. These measures are often described by such Orwellian phrases, such as "digital rights management," which actually means limitation or outright destruction of users' legal rights, or "trusted computing," which actually means selling people computers they cannot trust. However, these measures are alike in one basic respect. They all employ technical means to turn the system of copyright law (where the powers of the copyright holder are limited exceptions to general freedom) into a virtual prison, where everything not specifically permitted is utterly forbidden. This system of "para-copyright" was created well after GPLv2 was written — initially through legislation in the USA and the EU, and later in other jurisdictions as well. This legislation creates serious civil or even criminal penalties to escape from these restrictions (commonly and aptly called "jail-breaking a device"), even where the purpose in doing so is to restore the users' legal rights that the technology wrongfully prevents them from exercising.

GPLv2 did not address the use of technical measures to take back the rights that the GPL granted, because such measures did not exist in 1991, and would have been irrelevant to the forms in which software was then delivered to users. GPLv3 addresses these issues, particularly because copylefted software is ever more widely embedded in devices that impose technical limitations on the user's freedom to change it.

However, FSF always made a clear distinction to avoid conflating these "lock-down" measures with legitimate applications that give users control, as by enabling them to choose higher levels of system or data security within their networks, or by allowing them to protect the security of their communications using keys they can generate or copy to other devices for sending or receiving messages. Such technologies present no obstacles to software freedom and the goals of copyleft.

The public GPLv3 drafting process sought to balance these positions of copyleft advocates with various disparate views of the larger Free-Software-using community. Ultimately, FSF compromised to the GPLv3§3 and GPLv3§6 provisions that, taken together, are a minimalist set of terms sufficient to protect the software freedom against the threat of invasive para-copyright.

The compromises made were ultimately quite reasonable. The primary one is embodied in GPLv3§6's

"User Product" definition (see § 9.9.2 in this tutorial for details). Additionally, some readers of early GPLv3 drafts seem to have assumed GPLv3 contained a blanket prohibition on DRM; but it does not. In fact, no part of GPLv3 forbids DRM regarding non-GPL'd works; rather, GPLv3 forbids the use of DRM specifically to lock-down restrictions on users' ability to install modified versions of the GPL'd software itself, but again, *only* with regard to User Products.

9.6 GPLv3 §3: What Hath DMCA Wrought

As discussed in § 1.2.3 of this tutorial, 17 USC §1201 and related sections[5] prohibits users from circumventing technological measures that implement DRM. Since this is part of copyright law and the GPL is primarily a copyright license, and since what the DMCA calls "circumvention" is simply "modifying the software" under the GPL, GPLv3 must disclaim that such anti-circumvention provisions are not applicable to the GPLv3'd software. GPLv3§3 shields users from being subjected to liability under anti-circumvention law for exercising their rights under the GPL, so far as the GPL can do so.

First, GPLv3§3¶1 declares that no GPL'd program is part of an effective technological protection measure, regardless of what the program does. Early drafts of GPLv3§3¶1 referred directly to the DMCA, but the final version instead includes instead an international legal reference to anticircumvention laws enacted pursuant to the 1996 WIPO treaty and any similar laws. Lawyers outside the USA worried that a USA statutory reference could be read as indicating a choice for application of USA law to the license as a whole. While the FSF did not necessarily agree with that view, the FSF decided anyway to refer to the WIPO treaty rather than DMCA, since several national anticircumvention laws were (or will likely be) structured more similarly to the anticircumvention provisions of the DMCA in their implementation of WIPO. Furthermore, the addition of "or similar laws" provides an appropriate catch-all.

GPLv3§3¶2 states precisely that a conveying party waives the power to forbid circumvention of technological measures only to the extent that such circumvention is accomplished through the exercise of GPL rights in the conveyed work. GPLv3§3¶2 makes clear that the referenced "legal rights" are specifically rights arising under anticircumvention law. and refers to both the conveying party's rights and to third party rights, as in some cases the conveying party will also be the party legally empowered to enforce or invoke rights arising under anticircumvention law.

These disclaimers by each licensor of any intention to use GPL'd software to stringently control access to other copyrighted works should effectively prevent any private or public parties from invoking DMCA-like laws against users who escape technical restriction measures implemented by GPL'd software.

9.7 GPLv3 §4: Verbatim Copying

GPLv3 §4 is a revision of GPLv2 §1 (as discussed in § 3.2 of this tutorial). There are almost no changes to this section from the GPLv2 §1, other than to use the new defined terms.

The only notable change, of "a fee" to "any price or no price", is in the first sentence of GPLv3§4¶2. The GPLv2§1¶1 means that the GPL permits one to charge money for the distribution of software. Despite efforts by copyleft advocates to explain this in GPLv2 itself and in other documents, there are evidently some people who still believe that GPLv2 allows charging for services but not for selling copies of software and/or that the GPL requires downloads to be gratis. Perhaps this is because GPLv2 referred to charging a "fee"; the term "fee" is generally used in connection with services.

GPLv2's wording also referred to "the physical act of transferring." The intention was to distinguish charging for transfers from attempts to impose licensing fees on all third parties. "Physical" might be read, however, as suggesting "distribution in a physical medium only".

To address these two issues, GPLv3 says "price" in place of "fee," and removes the term "physical."

GPLv3 §4 has also been revised from its corresponding section in GPLv2 in light of the GPLv3 §7 (see § 9.9.3 in this tutorial for more). Specifically, a distributor of verbatim copies of the program's source code must obey any existing additional terms that apply to parts of the program pursuant to GPLv3 §7. In

[5]These sections of the USC are often referred to as the "Digital Millennium Copyright Act", or "DMCA", as that was the name of the bill that so-modified these sections of the USC.

addition, the distributor is required to keep intact all license notices, including notices of such additional terms.

Finally, there is no harm in explicitly pointing out what ought to be obvious: that those who convey GPL-covered software may offer commercial services for the support of that software.

9.8 GPLv3 §5: Modified Source

GPLv3§5 is the rewrite of GPLv2§2, which was discussed in § 5.1 of this tutorial. This section discusses the changes found in GPLv3§5 compared to GPLv2§2.

GPLv3§5(a) still requires modified versions be marked with "relevant date", but no longer says "the date of any change". The best practice is to include the date of the latest and/or most significant changes and who made those. Of course, compared to its GPLv2§2(a), GPLv3§5(a) slightly relaxes the requirements regarding notice of changes to the program. In particular, the modified files themselves need no longer be marked. This reduces administrative burdens for developers of modified versions of GPL'd software.

GPLv3§5(b) is a new but simple provision. GPLv3§5(b) requires that the license text itself must be unmodified (except as permitted by GPLv3§7; see § 9.9.3 in this tutorial). Furthermore, it removes any perceived conflict between the words "keep intact all notices" in GPLv3§4, since operating under GPLv3§5 still includes all the requirements of GPLv3§4 by reference.

GPLv3§5(c) is the primary source-code-related copyleft provision of GPL. (The object-code-related copyleft provisions are in GPLv3§6, discussed in § 9.9 of this tutorial). Compared to GPLv2§2(b), GPLv3§5(c) states that the GPL applies to the whole of the work. Such was stated already in GPLv2§2(b), in "in whole or in part", but this simplified wording makes it clear it applies to the entire covered work.

Another change in GPLv3§5(c) is the removal of the words "at no charge," which was often is misunderstood upon naïve reading of in GPLv2§(b) (as discussed in § 5.1.2 of this tutorial).

Note that of GPLv2 §2's penultimate and ante-penultimate paragraphs are now handled adequately by the definitions in GPLv3§0 and as such, have no direct analogs in GPLv3.

GPLv2 §2's final paragraph, however, is reworded and expanded into the final paragraph of GPLv3§5, which now also covers issues related to copyright compilations (but not compilations into object code — that's in the next section!). The intent and scope is the same as was intended in GPLv2.

9.9 GPLv3 §6: Non-Source and Corresponding Source

GPLv3 §6 states the compliance obligations for distributing "non-source forms" of a program (which means any form other than CCS). As noted in § 9.2, "object code" in GPLv3 is defined broadly to mean any non-source version of a work, and thus includes not only binaries or executables, but also obfuscated, minimized, compressed or otherwise non-preferred forms for modification. Thus, GPLv3 §6 clarifies and revises GPLv2 §3. Indeed, GPLv3 §6's CCS requirement under closely parallels the provisions of GPLv2 §3, with changes designed to make compliant provisioning easier under contemporary technological conditions. Distributors of GPLv3'd object code must provide access to the corresponding source code, in one of four specified ways.

GPLv3 §6(a–b) now apply specifically to distribution of object code in a physical product. Physical products include embedded systems, as well as physical software distribution media such as CDs. As in GPLv2 §3 (discussed in § 5.2 of this tutorial), the distribution of object code may either be accompanied by the machine-readable source code, or it may be accompanied by a valid written offer to provide the machine-readable source code. However, unlike in GPLv2, that offer cannot be exercised by any third party; rather, only those "who possess the object code" can exercise the offer. (Note that this is a substantial narrowing of requirements of offer fulfillment, and is a wonderful counterexample to dispute claims that the GPLv3 has more requirements than GPLv2.)

GPLv3 §6(b) further revises the requirements for the written offer to provide source code. As before, the offer must remain valid for at least three years. In addition, even after three years, a distributor of a product containing GPL'd object code must offer to provide source code for as long as the distributor also continues to offer spare parts or customer support for the product model. This is a reasonable and

appropriate requirement; a distributor should be prepared to provide source code if he or she is prepared to provide support for other aspects of a physical product.

GPLv3 §6(a–b) clarifies that the medium for software interchange on which the machine-readable source code is provided must be a durable physical medium. GPLv3 §6(b)(2), however, permits a distributor to instead offer to provide source code from a network server instead, which is yet another example GPLv3 looser in its requirements than GPLv2 (see § 5.2.2 for details).

GPLv3§6(c) gives narrower permission than GPLv2§3(c). The "pass along" option for GPLv3§6(c)(1) offers is now available only for individual distribution of object code; moreover, such individual distribution can occur only "occasionally and noncommercially." A distributor cannot comply with the GPL merely by making object code available on a publicly-accessible network server accompanied by a copy of the written offer to provide source code received from an upstream distributor.

GPLv3 §6(d) revises and improves GPLv2 §3's final paragraph. When object code is provided by offering access to copy the code from a designated place (such as by enabling electronic access to a network server), the distributor must merely offer equivalent access to copy the source code "in the same way through the same place". This wording also permits a distributor to offer a third party access to both object code and source code on a single network portal or web page, even though the access may include links to different physical servers. For example, a downstream distributor may provide a link to an upstream distributor's server and arrange with the operator of that server to keep the source code available for copying for as long as the downstream distributor enables access to the object code. Thus, the obligation remains on the party distributing object code to point prominently ("next to" the object code download) to the third-party source code provisioning server, and to ensure that this third-party server remains in operation for required period. This codifies formally the typical historical interpretation of GPLv2.

Furthermore, under GPLv3 §6(d), distributors may charge for the conveyed object code; however, those who pay to obtain the object code must be given equivalent and gratis access to obtain the CCS. (If distributors convey the object code gratis, distributors must likewise make CCS available without charge.) Those who do not obtain the object code from that distributors (perhaps because they choose not to pay the fee for object code) are outside the scope of the provision; distributors are under no specific obligation to give CCS to someone who has not purchased an object code download under GPLv3 §6(d). (Note: this does not change nor impact any obligations under GPLv3 §6(b)(2); GPLv3 §6(d) is a wholly different provision.)

9.9.1 GPLv3 §6(e): Peer-to-Peer Sharing Networks

GPLv3 §6(e) allows provision of CCS via another server when the binary or other non-source form is distributed by peer-to-peer protocols such as BitTorrent. Here the requirement is only that each peer be effectively informed of the location of the source code on a server as above.

GPLv3 really did require this addition, even though it adds complexity to a key section of GPL. In particular, Decentralized peer-to-peer file sharing present a challenge to the unidirectional view of distribution that is implicit in GPLv2 and initial drafts of GPLv3. Identification of an upstream/downstream link in BitTorrent distribution is neither straightforward nor reasonable; such distribution is multidirectional, cooperative and (somewhat) anonymous. In peer-to-peer distribution systems, participants act both as transmitters and recipients of blocks of a particular file, but they perceive the experience merely as users and receivers, and not as distributors in any conventional sense. At any given moment of time, most peers will not have the complete file.

Meanwhile, GPLv3 §6(d) permits distribution of a work in object code form over a network, provided that the distributor offers equivalent access to copy the Corresponding Source Code "in the same way through the same place". This wording might be interpreted to permit peer-to-peer distribution of binaries *if* they are packaged together with the CCS, but such packaging is impractical, for at least three reasons. First, even if the CCS is packaged with the object code, it will only be available to a non-seeding peer at the end of the distribution process, but the peer will already have been providing parts of the binary to others in the network. Second, in practice, peer-to-peer forms of transmission are poorly suited means for distributing CCS. In large distributions, packaging CCS with the object code may result in a substantial increase in file size and transmission time. Third, in current practice, CCS packages themselves tend *not* to be transmitted through BitTorrent — owing to reduced demand – thus, there generally will be too few participants downloading the same source package at the same time to enable effective seeding and

distribution.

GPLv3 §6(e) addresses these issues. If a licensee conveys such a work of object code using peer-to-peer transmission, that licensee is in compliance with GPLv3 §6 if the licensee informs other peers where the object code and its CCS are publicly available at no charge under subsection GPLv3 §6(d). The CCS therefore need not be provided through the peer-to-peer system that was used for providing the binary.

Second, GPLv3§9 also clarifies that ancillary propagation of a covered work that occurs as part of the process of peer-to-peer file transmission does not require acceptance, just as mere receipt and execution of the Program does not require acceptance. Such ancillary propagation is permitted without limitation or further obligation.

9.9.2 User Products, Installation Information and Device Lock-Down

As discussed in § 9.5 of this tutorial, GPLv3 seeks to thwart technical measures such as signature checks in hardware to prevent modification of GPL'd software on a device.

To address this issue, GPLv3 §6 requires that parties distributing object code provide recipients with the source code through certain means. When those distributors pass on the CCS, they are also required to pass on any information or data necessary to install modified software on the particular device that included it. (This strategy is not unlike that used in LGPLv2.1 to enable users to link proprietary programs to modified libraries.)

User Products

The scope of these requirements is narrow. GPLv3 §6 introduces the concept of a "User Product", which includes devices that are sold for personal, family, or household use. Distributors are only required to provide Installation Information when they convey object code in a User Product.

In brief, the right to convey object code in a defined class of "User Products," under certain circumstances, depends on providing whatever information is required to enable a recipient to replace the object code with a functioning modified version.

This was a compromise that was difficult for the FSF to agree to during the GPLv3 drafting process. However, companies and governments that use specialized or enterprise-level computer facilities reported that they actually *want* their systems not to be under their own control. Rather than agreeing to this as a concession, or bowing to pressure, they ask for this as a *preference*. It is not clear that the GPL should interfere here, since the main problem lies elsewhere.

While imposing technical barriers to modification is wrong regardless of circumstances, the areas where restricted devices are of the greatest practical concern today fall within the User Product definition. Most, if not all, technically-restricted devices running GPL-covered programs are consumer electronics devices. Moreover, the disparity in clout between the manufacturers and these users makes it difficult for the users to reject technical restrictions through their weak and unorganized market power. Even limited to User Products, this provision addresses the fundamental problem.

The core of the User Product definition is a subdefinition of "consumer product" adapted from the Magnuson-Moss Warranty Act, a federal consumer protection law in the USA found in 15 USC §2301: "any tangible personal property which is normally used for personal, family, or household purposes." The USA has had three decades of experience of liberal judicial and administrative interpretation of this definition in a manner favorable to consumer rights.[6] Ideally, this body of interpretation[7] will guide interpretation of the consumer product subdefinition in GPLv3 §6, and this will hopefully provide a degree of legal certainty advantageous to device manufacturers and downstream licensees alike.

One well-established interpretive principle under Magnuson-Moss is that ambiguities are resolved in favor of coverage. That is, in cases where it is not clear whether a product falls under the definition of consumer product, the product will be treated as a consumer product.[8] Moreover, for a given product, "normally used" is understood to refer to the typical use of that type of product, rather than a particular use by

[6]The Magnuson-Moss consumer product definition itself has been influential in the USA and Canada, having been adopted in several state and provincial consumer protection laws.

[7]The FSF, however, was very clear that incorporation of such legal interpretation was in no way intended to work as a general choice of USA law for GPLv3.

[8]16 CFR § 700.1(a); *McFadden v. Dryvit Systems, Inc.*, 54 UCC Rep. Serv.2d 934 (D. Ore. 2004).

a particular buyer. Products that are commonly used for personal as well as commercial purposes are consumer products, even if the person invoking rights is a commercial entity intending to use the product for commercial purposes.[9] Even a small amount of "normal" personal use is enough to cause an entire product line to be treated as a consumer product under Magnuson-Moss.[10]

However, Magnuson-Moss is not a perfect fit because in the area of components of dwellings, the settled interpretation under Magnuson-Moss is under-inclusive. Depending on how such components are manufactured or sold, they may or may not be considered Magnuson-Moss consumer products.[11] Therefore, GPLv3 defines User Products as a superset of consumer products that also includes "anything designed or sold for incorporation into a dwelling."

Thus, the three sentences in the center of GPLv3's User Product definition encapsulate the judicial and administrative principles established over the past three decades in the USA concerning the Magnuson-Moss consumer product definition. First, it states that doubtful cases are resolved in favor of coverage under the definition. Second, it indicates that the words "normally used" in the consumer product definition refer to a typical or common use of a class of product, and not the status of a particular user or expected or actual uses by a particular user. Third, it clearly states that the existence of substantial non-consumer uses of a product does not negate a determination that it is a consumer product, unless such non-consumer uses represent the only significant mode of use of that product.

It should be clear from these added sentences that it is the general mode of use of a product that determines objectively whether or not it is a consumer product. One could not escape the effects of the User Products provisions by labeling what is demonstrably a consumer product in ways that suggest it is "for professionals", for example.

Installation Information

With the User Products definition complete, the "Installation Information" definition uses that to define what those receiving object code inside a User Product must receive.

Installation Information is information that is "required to install and execute modified versions of a covered work . . . from a modified version of its" CCS, in the same User Product for which the covered work is conveyed. GPLv3 provides guidance concerning how much information must be provided: it "must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made." For example, the information provided would be insufficient if it enabled a modified version to run only in a disabled fashion, solely because of the fact of modification (regardless of the actual nature of the modification). The information need not consist of cryptographic keys; Installation Information may be "any methods, procedures, authorization keys, or other information".

Note that GPLv3 does not define "continued functioning" further. However, GPLv3 does provide some additional guidance concerning the scope of GPLv3-compliant action or inaction that distributors of technically-restricted User Products can take with respect to a downstream recipient who replaces the conveyed object code with a modified version. First of all, GPLv3 makes clear that GPLv3 implies no obligation "to continue to provide support service, warranty, or updates" for such a work.

Second, most technically-restricted User Products are designed to communicate across networks. It is important for both users and network providers to know when denial of network access to devices running modified versions becomes a GPL violation. GPLv3 permits denial of access in two cases: "when the modification itself materially and adversely affects the operation of the network," and when the modification itself "violates the rules and protocols for communication across the network". The second case is deliberately drawn in general terms, and it serves as a foundation for reasonable enforcement policies that respect recipients' right to modify while recognizing the legitimate interests of network providers.

[9] 16 CFR § 700.1(a). Numerous court decisions interpreting Magnuson-Moss are in accord; see, e.g., *Stroebner Motors, Inc. v. Automobili Lamborghini S.p.A.*, 459 F. Supp.2d 1028, 1033 (D. Hawaii 2006).

[10] *Tandy Corp. v. Marymac Industries, Inc.*, 213 U.S.P.Q. 702 (S.D. Tex. 1981). In this case, the court concluded that TRS-80 microcomputers were consumer products, where such computers were designed and advertised for a variety of users, including small businesses and schools, and had only recently been promoted for use in the home.

[11] Building materials that are purchased directly by a consumer from a retailer, for improving or modifying an existing dwelling, are consumer products under Magnuson-Moss, but building materials that are integral component parts of the structure of a dwelling at the time that the consumer buys the dwelling are not consumer products. 16 C.F.R. §§ 700.1(c)–(f); Federal Trade Commission, Final Action Concerning Review of Interpretations of Magnuson-Moss Warranty Act, 64 Fed. Reg. 19,700 (April 22, 1999); see also, e.g., *McFadden*, 54 U.C.C. Rep. Serv.2d at 934.

Note that GPLv3 permits the practice of conveying object code in a mode not practically susceptible to modification by any party, such as code burned in ROM or embedded in silicon. The goal of the Installation Information requirement is to ensure the downstream licensee receives the real right to modify when the device manufacturer or some other party retains that right. Accordingly, GPLv3§6's ante-penultimate paragraph states that the requirement to provide Installation Information "does not apply if neither you nor any third party retains the ability to install modified object code on the User Product".

Finally, GPLv3§6 makes it clear that there is also no requirement to provide warranty or support for the User Product itself.

9.9.3 GPLv3 §7: Additional Permissions

The GPL is a statement of permissions, some of which have conditions. Additional terms — terms that supplement those of the GPL — may come to be placed on, or removed from, GPL-covered code in certain common ways. Copyleft licensing theorists have generally called those added terms "additional permissions" if they grant exceptions from the conditions of the GPL, and "additional requirements" if they add conditions to the basic permissions of the GPL. The treatment of additional permissions and additional requirements under GPLv3 is necessarily asymmetrical, because they do not raise the same interpretive issues; in particular, additional requirements, if allowed without careful limitation, could transform a GPL'd program into a non-free one.

Due to the latter fear, historically, GPLv2 did not permit any additional requirements. However, over time, many copyright holders generally tolerated certain types of benign additional requirements merely through a "failure to enforce" estoppel-esque scenario. Therefore, GPLv3 allows for some specific limited requirement variations that GPLv2 technically prohibits.

With these principles in the background, GPLv3 §7 answers the following questions:

1. How does the presence of additional terms on all or part of a GPL'd program affect users' rights?
2. When and how may a licensee add terms to code being distributed under the GPL?
3. When may a licensee remove additional terms?

Additional permissions present the easier case. Since the mid-1990s, permissive exceptions often appeared alongside GPLv2 to allow combination with certain non-free code. Typically, downstream stream recipients could remove those exceptions and operate under pure GPLv2. Similarly, LGPLv2.1 is in essence a permissive variant of GPLv2, and it permits relicensing under the GPL.

These practices are now generalized via GPLv3 §7. A licensee may remove any additional permission from a covered work, whether it was placed by the original author or by an upstream distributor. A licensee may also add any kind of additional permission to any part of a work for which the licensee has, or can give, appropriate copyright permission. For example, if the licensee has written that part, the licensee is the copyright holder for that part and can therefore give additional permissions that are applicable to it. Alternatively, the part may have been written by someone else and licensed, with the additional permissions, to that licensee. Any additional permissions on that part are, in turn, removable by downstream recipients. As GPLv3 §7¶1 explains, the effect of an additional permission depends on whether the permission applies to the whole work or a part.

Indeed, LGPLv3 is itself simply a list of additional permissions supplementing the terms of GPLv3. GPLv3§7 has thus provided the basis for recasting a formally complex license as an elegant set of added terms, without changing any of the fundamental features of the existing LGPL. LGPLv3 is thus a model for developers wishing to license their works under the GPL with permissive exceptions. The removability of additional permissions under GPLv3§7 does not alter any existing behavior of the LGPL since the LGPL has always allowed relicensing under the ordinary GPL.

9.10 GPLv3 §7: Understanding License Compatibility

A challenge that faced the Free Software community heavily through out the early 2000s was the proliferation of incompatible Free Software licenses. Of course, the GPL cannot possibly be compatible with all such

licenses. However, GPLv3 contains provisions that are designed to reduce license incompatibility by making it easier for developers to combine code carrying non-GPL terms with GPL'd code.

This license compatibility issue arises for three reasons. First, the GPL is a strong copyleft license, requiring modified versions to be distributed under the GPL. Second, the GPL states that no further restrictions may be placed on the rights of recipients. Third, all other software freedom respecting licenses in common use contain certain requirements, many of which are not conditions made by the GPL. Thus, when GPL'd code is modified by combination with code covered by another formal license that specifies other requirements, and that modified code is then distributed to others, the freedom of recipients may be burdened by additional requirements in violation of the GPL. It can be seen that additional permissions in other licenses do not raise any problems of license compatibility.

GPLv3 took a new approach to the issue of combining GPL'd code with code governed by the terms of other software freedom licenses. Traditional GPLv2 license compatibility theory (which was not explicitly stated in GPLv2 itself, but treated as a license interpretation matter by the FSF) held that GPLv2 allowed such combinations only if the non-GPL licensing terms permitted distribution under the GPL and imposed no restrictions on the code that were not also imposed by the GPL. In practice, the FSF historically supplemented that policy with a structure of exceptions for certain kinds of combinations.

GPLv3 §7 implements a more explicit policy on license compatibility. It formalizes the circumstances under which a licensee may release a covered work that includes an added part carrying non-GPL terms. GPLv3 §7 distinguish between terms that provide additional permissions, and terms that place additional requirements on the code, relative to the permissions and requirements established by applying the GPL to the code.

As discussed in the previous section of this tutorial, GPLv3 §7 first and foremost explicitly allows added parts covered by terms with additional permissions to be combined with GPL'd code. This codifies the existing practice of regarding such licensing terms as compatible with the GPL. A downstream user of a combined GPL'd work who modifies such an added part may remove the additional permissions, in which case the broader permissions no longer apply to the modified version, and only the terms of the GPL apply to it.

In its treatment of terms that impose additional requirements, GPLv3§7 extends the range of licensing terms with which the GPL is compatible. An added part carrying additional requirements may be combined with GPL'd code, but only if those requirements belong to a set enumerated in GPLv3§7. There are, of course, limits on the acceptable additional requirements, which ensures that enhanced license compatibility does not defeat the broader software-freedom-defending terms of the GPL. Unlike terms that grant additional permissions, terms that impose additional requirements cannot be removed by a downstream user of the combined GPL'd work, because only in the pathological case[12] would a user have the right to do so.

In general, the types of additional requirements were those terms in regular use by other non-copyleft Free Software licenses that the FSF found unobjectionable. The specific details GPLv3's permitted additional requirements hat GPLv3 are as follows:

7(a): This provision allows alternative "disclaimer of warranty" forms. Copyright holders can disclaim warranty or limit liability differently from the terms as provided under GPLv3§ §15–16. Drafters included this permission to advance the internationalization goals of GPLv3; international treaties lack adequate harmonization for laws regarding warranty and disclaimer.

7(b): This provision allows alternative requirements for preservation of appropriate legal notices. GPLv3 permits additional requirements regarding preservation of legal notices, including on output from execution of covered works. Preserved information can include information about the origins of the code or alterations of the code.

7(c): This provision allows prohibition of misrepresentation of original material. The provision yields compatibility with non-copyleft Free Software licenses that require marking of modified versions in "reasonable"ways which differ from GPL's own precise marking requirements.

7(d): This provision allows limitations on the use of names of licensor for publicity purposes. This provision also yields additional compatibility with non-copyleft Free Software licenses that prohibit the use of

[12]Theoretically, a user could collect copyright assignment from all known contributors and then do this, but this would indeed be the pathological case.

the licensor's name on unmodified versions (or other prohibitions on advertising rights). The third clause of the 3-Clause BSD License, for example, long considered de-facto compatible with GPLv2 anyway, is via this clause unequivocally compatible with GPLv3. However, this clause *does not* make GPL compatible with the old BSD advertising clause that the FSF long ago identified as problematic.

7(e): This provision clarifies that refusal to grant trademark rights for a GPLv3'd covered work remains compatible with GPLv3. Again, some non-copyleft permissive licenses include such clauses.

7(f): This provision allows indemnification requirements of authors and licensors. The FSF specifically designed this clause to achieve GPLv3 compatibility for the Apache Software License, Version 2.0.

During the GPLv3 drafting process, some questioned the necessity of GPLv3 §7; those critics suggested that it creates complexity that did not previously exist. However, by the time of GPLv3's drafting, many existing GPLv2'd software packages already combined with various non-copylefted Free Software licensed code that carried such additional terms. Therefore, GPLv3 §7 is rationalized existing practices of those package authors and modifiers, since it sets clear guidelines regarding the removal and addition of these additional terms. With its carefully limited list of allowed additional requirements, GPLv3§7 accomplishes additional objectives as well, since it permits the expansion of the base of code available for GPL developers, while also encouraging useful experimentation with requirements the GPLv3 does not include by default.

However, any other non-permissive additional terms apart from those stated above are considered "further" restrictions which GPLv3 §10 prohibits. Furthermore, as a compliance matter, if you add additional terms in accordance with GPLv3 §7, you must ensure that the terms are placed in the relevant source files or provide a conspicuous notice about where to find the additional terms.

9.11 GPLv3 §8: A Lighter Termination

GPLv2 provided for automatic termination of the rights of a person who copied, modified, sublicensed, or distributed a work in violation of the license. Automatic termination can be too harsh for those who have committed an inadvertent violation, particularly in cases involving distribution of large collections of software having numerous copyright holders. A violator who resumes compliance with GPLv2 technically needs to obtain forgiveness from all copyright holders, and even contacting them all might be impossible.

GPLv3 §8 now grants opportunities for provisional and permanent reinstatement of rights. The termination procedure provides a limited opportunity to cure license violations. If a licensee has committed a first-time violation of the GPL with respect to a given copyright holder, but the licensee cures the violation within 30 days following receipt of notice of the violation, then any of the licensee's GPL rights that have been terminated by the copyright holder are "automatically reinstated".

Finally, if a licensee violates the GPL, a contributor may terminate any patent licenses that it granted under GPLv3 §11, in addition to any copyright permissions the contributor granted to the licensee.

9.12 GPLv3 §9: Acceptance

GPLv3 §9 means what it says: mere receipt or execution of code neither requires nor signifies contractual acceptance under the GPL. Speaking more broadly, GPLv3 is intentionally structured as a unilateral grant of copyright permissions, the basic operation of which exists outside of any law of contract. Whether and when a contractual relationship is formed between licensor and licensee under local law do not necessarily matter to the working of the license.

9.13 GPLv3 §10: Explicit Downstream License

GPLv3 §10 is a generally straightforward section that ensures that everyone downstream receives licenses from all copyright holders. Each time you redistribute a GPL'd program, the recipient automatically receives a license, under the terms of GPL, from every upstream licensor whose copyrighted material is present in

the work you redistribute. You could think of this as creating a three-dimensional rather than linear flow of license rights. Every recipient of the work is "in privity," or is directly receiving a license from every licensor.

This mechanism of automatic downstream licensing is central to copyleft's function. Every licensor independently grants licenses, and every licensor independently terminates the license on violation. Parties further downstream from the infringing party remain licensed, so long as they don't themselves commit infringing actions. Their licenses come directly from all the upstream copyright holders, and are not dependent on the license of the breaching party who distributed to them. For the same reason, an infringer who acquires another copy of the program has not thereby acquired any new license rights: once any upstream licensor of that program has terminated the license for breach of its terms, no new automatic license will issue to the recipient just by acquiring another copy[13]

Meanwhile, one specific addition in GPLv3 here in GPLv3 §10 deserves special mention. Specifically, GPLv3 removed the words "at no charge" from GPLv2 §2(b) (which, BTW, became GPLv3 §5(b)) because it contributed to a misconception that the GPL did not permit charging for distribution of copies. The purpose of the "at no charge" wording was to prevent attempts to collect royalties from third parties. The removal of these words created the danger that the imposition of licensing fees would no longer be seen as a license violation. Therefore, GPLv3 §10 adds a new explicit prohibition on imposition of licensing fees or royalties. This section is an appropriate place for such a clause, since it is a specific consequence of the general requirement that no further restrictions be imposed on downstream recipients of GPL-covered code.

9.14 GPLv3 §11: Explicit Patent Licensing

Software patenting is a harmful and unjust policy, and should be abolished; recent experience makes this all the more evident. Since many countries grant patents that can apply to and prohibit software packages, in various guises and to varying degrees, GPLv3 seeks to protect the users of GPL-covered programs from those patents, while at the same time making it feasible for patent holders to contribute to and distribute GPL-covered programs as long as they do not attack the users of those programs.

It is generally understood that GPLv2 implies some limits on a licensee's power to assert patent claims against the use of GPL-covered works. However, the patent licensing practice that GPLv2 §7 (corresponding to GPLv3 §12) is designed to prevent is only one of several ways in which software patents threaten to make free programs non-free and to prevent users from exercising their rights under the GPL. GPLv3 takes a more comprehensive approach to combating the danger of patents.

GPLv2 §7 has seen some success in deterring conduct that would otherwise result in denial of full downstream enjoyment of GPL rights, and thus it is preserved in GPLv3 §12. Experience has shown that more is necessary, however, to ensure adequate community safety where companies act in concert to heighten the anticompetitive use of patents that they hold or license.

Therefore, GPLv3 is designed to reduce the patent risks that distort and threaten the activities of users who make, run, modify and share Free Software. At the same time, GPLv3 gives favorable consideration to practical goals such as certainty and administrability for patent holders that participate in distribution and development of GPL-covered software. GPLv3's policy requires each such patent holder to provide appropriate levels of patent assurance to users, according to the nature of the patent holder's relationship to the program.

In general, GPLv3 provides for two classes of patent commitments:

- Grant of license to claims in contributor versions: GPLv3 §11 introduces an affirmative grant of rights to patent claims by those who contribute code to GPL'd programs. The intent is to prevent parties from aggressively asserting patents against users of code those parties have themselves modified — in theory preventing betrayal by "insiders" of the copyleft community. A contributor's patent claims necessarily infringed by the version of the program created by the incorporation of its modifications are licensed to all subsequent users and modifiers of the program, or programs based on the program. No patent claims only infringed by subsequent modifications by other parties are thus licensed. Patent claims acquired after the making of the "contributor version" necessarily infringed by that version are also licensed by this provision at the time of their acquisition or perfection.

[13] Footnote 3 also applies here in discussion of GPLv3 just as it did in discussion of GPLv2.

- Prohibition of enforcement of patent claims against those to whom you distribute: GPLv3 §10 makes explicit that licensees who directly distribute may not make demands for acceptance of patent licenses or payment of patent royalties from distribution recipients. This provision establishes a uniform rule of patent exhaustion with respect to GPL'd programs regardless of the domestic patent law in any particular system or locale.

The following two subsections discuss in order each of the above mentioned classes of patent commitments.

9.14.1 The Contributor's Explicit Patent License

Specifically, the ideal might have been for GPLv3 to feature a patent license grant triggered by all acts of distribution of GPLv3-covered works. The FSF considered it during the GPLv3 drafting process, but many patent-holding companies objected to this policy. They have made two objections: (1) the far-reaching impact of the patent license grant on the patent holder is disproportionate to the act of merely distributing code without modification or transformation, and (2) it is unreasonable to expect an owner of vast patent assets to exercise requisite diligence in reviewing all the GPL-covered software that it provides to others. Some expressed particular concern about the consequences of "inadvertent" distribution.

The argument that the impact of the patent license grant would be "disproportionate", that is to say unfair, is not valid. Since software patents are weapons that no one should have, and using them for aggression against free software developers is an egregious act (thus preventing that act cannot be unfair).

However, the second argument seems valid in a practical sense. A typical GNU/Linux distribution includes thousands of programs. It would be quite difficult for a re-distributor with a large patent portfolio to review all those programs against that portfolio every time it receives and passes on a new version of the distribution. Moreover, this question raises a strategic issue. If the GPLv3 patent license requirements convince patent-holding companies to remain outside the distribution path of all GPL-covered software, then these requirements, no matter how strong, will cover few patents.

GPLv3 therefore makes a partial concession which would lead these companies to feel secure in doing the distribution themselves. GPLv3 §11 applies only to those distributors that have modified the program. The other changes we have made in sections 10 and 11 provide strengthened defenses against patent assertion and compensate partly for this concession.

Therefore, GPLv3 §11 introduces the terms "contributor", "contributor version", and "essential patent claims", which are used in the GPLv3 §11¶3. Viewed from the perspective of a recipient of the Program, contributors include all the copyright holders for the Program, other than copyright holders of material originally licensed under non-GPL terms and later incorporated into a GPL-covered work. The contributors are therefore the initial GPLv3 licensors of the Program and all subsequent upstream licensors who convey, under the terms of GPLv3 §5, modified covered works. Thus, the "contributor version" includes the material the contributor has copied from the upstream version that the contributor has modified. GPLv3 §11¶3 does not apply to those that redistribute the program without change.[14] In other words, the "contributor version" includes not just the material added or altered by the contributor, but also the pre-existing material the contributor copied from the upstream version and retained in the modified version. (GPLv3's usage of "contributor" and "contribution" should not be confused with the various other ways in which those terms are used in certain other free software licenses.[15])

Some details of the "essential patent claims" definition deserve special mention. "Essential patent claims", for a given party, are a subset of the claims "owned or controlled" by the party. They do include sublicensable claims that have been licensed to the contributor by a third party.[16] Most commercial patent license agreements that permit sublicensing do so under restrictive terms that are inconsistent with the requirements of the GPL. For example, some patent licenses allow the patent licensee to sublicense but require collection of royalties from any sublicensees. The patent licensee could not distribute a GPL-covered program and grant the recipient a patent sublicense for the program without violating section 12 of GPLv3.[17] In rare cases,

[14] An implied patent license from the distributor, however, often arises. See § 6 in this tutorial

[15] Cf., e.g., Apache License, version 2.0, section 1; Eclipse Public License, version 1.0, section 1; Mozilla Public License, version 1.1, section 1.1.

[16] This issue is typically handled in other software freedom licenses having patent licensing provisions by use of the unhelpful term "licensable," which is either left undefined or is given an ambiguous definition.

[17] GPLv3 also provides an example in section 12 that makes this point clear.

however, a conveying party can freely grant patent sublicenses to downstream recipients without violating the GPL.

Additionally, "essential patent claims" are those patents "that would be infringed by some manner, permitted by this License, of making, using, or selling the work". This intends to make clear that a patent claim is "essential" if some mode of usage would infringe that claim, even if there are other modes of usage that would not infringe.

Finally, "essential patent claims ... do not include claims that would be infringed only as a consequence of further modification of the work." The set of essential patent claims licensed is fixed by the particular version of the work that was contributed. The claim set cannot expand as a work is further modified downstream. (If it could, then any software patent claim would be included, since any software patent claim can be infringed by some further modification of the work.)[18]

Ideally, this contributor patent policy will result in fairly frequent licensing of patent claims by contributors. A contributor is charged with awareness of the fact that it has modified a work and provided it to others; no act of contribution should be treated as inadvertent. GPLv3's rule also requires no more work, for a contributor, than the weaker rule proposed by the patent holders. Under their rule, the contributor must always compare the entire work against its patent portfolio to determine whether the combination of the modifications with the remainder of the work cause it to read on any of the contributor's patent claims.

Finally, GPLv3's explicit patent license for contributors has an interesting and useful side effect. When a company with a large number of such claims acquires the program's modifier, all claims held or thereafter acquired by the purchaser are automatically licensed under this provision. For example, Microsoft's acquisition of Nokia resulted in the automatic licensing of all Microsoft patent claims now or hereafter acquired which read on any contributor version of any GPLv3 program ever modified by Nokia.

9.14.2 Conveyors' Patent Licensing

The remaining patent licensing in GPLv3 deals with patent licenses that are granted by conveyance. The licensing is not as complete or far reaching as the contributor patent licenses discussed in the preceding section.

The term "patent license," as used in GPLv3 §11¶4–6, is not meant to be confined to agreements formally identified or classified as patent licenses. GPLv3 §11¶3 makes this clear by defining "patent license," for purposes of the subsequent three paragraphs, as "any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement)"

GPLv3 §11¶5 is commonly called GPLv3's downstream shielding provision. It responds particularly to the problem of exclusive deals between patent holders and distributors, which threaten to distort the free software distribution system in a manner adverse to developers and users. The fundamental idea is to make a trade-off between assuring a patent license for downstream and making (possibly patent-encumbered) CCS publicly available.

Simply put, in nearly all cases in which the "knowingly relying" test is met, the patent license will indeed not be sublicensable or generally available to all on free terms. If, on the other hand, the patent license is generally available under terms consistent with the requirements of the GPL, the distributor is automatically in compliance, because the patent license has already been extended to all downstream recipients. Finally, if the patent license is sublicensable on GPL-consistent terms, the distributor may choose to grant sublicenses to downstream recipients instead of causing the CCS to be publicly available. (In such a case, if the distributor is also a contributor, it will already have granted a patent sublicense anyway, and so it need not do anything further to comply with the third paragraph.)

Admittedly, public disclosure of CCS is not necessarily required by other sections of the GPL, and the FSF in drafting GPLv3 did not necessarily wish to impose a general requirement to make source code available to all, which has never been a GPL condition. However, many vendors who produce products that include copylefted software, and who are most likely to be affected by the downstream shielding provision, lobbied for the addition of the source code availability option, so it remains.

[18] However, "the work" should not be understood to be restricted to a particular mechanical affixation of, or medium for distributing, a program, where the same program might be provided in other forms or in other ways that may be captured by other patent claims held by the contributor.

Meanwhile, two specific alternatives to the source code availability option are also available. The distributor may comply by disclaiming the patent license it has been granted for the conveyed work, or by arranging to extend the patent license to downstream recipients.[19] The GPL is intended to permit private distribution as well as public distribution, and the addition of these options ensures that this remains the case, even though it remains likely that distributors in this situation will usually choose the source code availability option.

Note that GPLv3 §11¶5 is activated only if the CCS is not already otherwise publicly available. (Most often it will, in fact, already be available on some network server operated by a third party.) Even if it is not already available, the option to "cause the Corresponding Source to be so available" can then be satisfied by verifying that a third party has acted to make it available. That is to say, the affected distributor need not itself host the CCS to take advantage of the source code availability option. This subtlety may help the distributor avoid certain peculiar assumptions of liability.

Note that GPLv3 §11¶6–7 are designed to stop distributors from colluding with third parties to offer selective patent protection. GPLv3 is designed to ensure that all users receive the same rights; arrangements that circumvent this make a mockery of free software, and we must do everything in our power to stop them.

First, GPLv3 §11¶6 states that any license that protects some recipients of GPL'd software must be extended to all recipients of the software. If conveyors arrange to provide patent protection to some of the people who get the software from you, that protection is automatically extended to everyone who receives the software, no matter how they get it.

Second, GPLv3 §11¶7 prohibits anyone who made such an agreement from distributing software released under GPLv3. Conveyors are prohibited from distributing software under GPLv3 if the conveyor makes an agreement of that nature in the future.

The date in GPLv3 §11¶7 likely seems arbitrary to those who did not follow the GPLv3 drafting process. This issue was hotly debated during the drafting of GPLv3, but ultimately one specific deal of this type — a deal between Microsoft and Novell for Microsoft to provide so-called "coupons" to Microsoft customers to redeem for copies of Novell's GNU/Linux distribution with a Microsoft patent license – was designed to be excluded.

The main reason for this was a tactical decision by the FSF. FSF believed they can do more to protect the community by allowing Novell to use software under GPLv3 than by forbidding it to do so. This is because of paragraph 6 of section 11 (corresponding to paragraph 4 in Draft 3). It will apply, under the Microsoft/Novell deal, because of the coupons that Microsoft has acquired that essentially commit it to participate in the distribution of the Novell SLES GNU/Linux system.

The FSF also gave a secondary reason: to avoid affecting other kinds of agreements for other kinds of activities. While GPLv3 sought to distinguish pernicious deals of the Microsoft/Novell type from business conduct that is not particularly harmful, the FSF also did not assume success in that drafting, and thus there remained some risk that other unchangeable past agreements could fall within the scope of GPLv3 §11¶7. In future deals, distributors engaging in ordinary business practices can structure the agreements so that they do not fall under GPLv3 §11¶7.

9.15 GPLv3 §12: Familiar as GPLv2 §7

GPLv2 §12 remains almost completely unchanged from the text that appears in GPLv2 §7. This is an important provision that ensures a catch-all to ensure that nothing "surprising" interferes with the continued conveyance safely under copyleft.

The wording in the first sentence of GPLv3 §12 has been revised slightly to clarify that an agreement – such as a litigation settlement agreement or a patent license agreement – is one of the ways in which conditions may be "imposed" on a GPL licensee that may contradict the conditions of the GPL, but which do not excuse the licensee from compliance with those conditions. This change codifies the historical interpretation of GPLv2.

GPLv3 removed the limited severability clause of GPLv2 §7 as a matter of tactical judgment, believing that this is the best way to ensure that all provisions of the GPL will be upheld in court. GPLv3 also

[19] The latter option, if chosen, must be done "in a manner consistent with the requirements of this License"; for example, it is unavailable if extension of the patent license would result in a violation of GPLv3 §12.

removed the final sentence of GPLv2 section 7, which the FSF consider to be unnecessary.

9.16 GPLv3 §13: The Great Affero Compromise

The Affero GPL was written with the expectation that its additional requirement would be incorporated into the terms of GPLv3 itself. Many software freedom advocates, including some authors of this tutorial, advocated heavily for that, and fully expected it to happen.

The FSF, however, chose not to include the Affero clause in GPLv3, due to what it called "irreconcilable views from different parts of the community". Many commercial users of Free Software were opposed to the inclusion of a mandatory Affero-like requirement in the body of GPLv3 itself. In fact, some wealthier companies even threatened to permanently fund forks of many FSF copyrighted-programs under GPLv2 if the Affero clause appeared in GPLv3.

Meanwhile, there was disagreement even among copyleft enthusiasts about the importance of the provision. A coalition never formed, and ultimately the more powerful interests implicitly allied with the companies who deeply opposed the Affero clause such that the FSF felt the Affero clause would need its own license, but one compatible with GPLv3.

GPLv3 §13 makes GPLv3 compatible with the AGPLv3, so that at least code can be shared between AGPLv3'd and GPLv3'd projects, even if the Affero clause does not automatically apply to all GPLv3'd works.

Meanwhile, those who criticize the permission to link with code under the Affero GPL should recognize that most other free software licenses also permit such linking. In particular, when a combined work is made by linking GPLv3-covered code with AGPLv3-covered code, the copyleft on one part will not extend to the other part. In such combinations, the Affero requirement will apply only to the part originally brought into the combination under the Affero license. In theory, those who receive such a combination and do not wish to use code under the Affero requirement may remove the Affero-covered portion of the combination. (Admittedly, in practice, de-mingling of combined code can be technically difficult.)

9.17 GPLv3 §14: So, When's GPLv4?

No substantive change has been made in section 14. The wording of the section has been revised slightly to make it clearer.

It's unclear when the FSF might consider publishing GPLv4. However, this section makes it clear that the FSF is the sole authority who can decide such.

The main addition to this section allows a third-party proxy to be appointed by contributors who wish someone else to make relicensing to new versions of GPL when they are released. This is a "halfway" point between using "-only" or "-or-later" by consolidating the decision-making on that issue to a single authority.

9.18 GPLv3 §15–17: Warranty Disclaimers and Liability Limitation

No substantive changes have been made in sections 15 and 16.

Finally, the FSF shortened the section on "How to Apply These Terms to Your New Programs" to just the bare essentials.

CHAPTER 10

THE LESSER GPL

As we have seen in our consideration of the GPL, its text is specifically designed to cover all possible derivative, modified and/or combined works under copyright law. Our goal in designing the GPL was to maximize its use of the controls of copyright law to maximize the number of works that were covered by GPL.

However, while the strategic goal of software freedom 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. The Lesser General Public License is sometimes described as a "weak copyleft" license, because code licensed under LGPL's terms can be combined with code under non-free licenses, and is sometimes used in that fashion.

10.1 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 the GPL were somewhat different. Applications released under the GPL would never themselves become part of proprietary software. However, if glibc were released under the GPL, it would require that any application distributed for the GNU/Linux platform be released under the 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 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 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 de-facto 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 the 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 the 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.

10.2 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, LGPLv2.1 §1, the rules for verbatim copying of source, are equivalent to those in GPLv2 §1.

Second, LGPLv2.1 §8 is equivalent GPLv2 §4. In both licenses, this section handles termination in precisely the same manner.

LGPLv2.1 §9 is equivalent to GPLv2 §5. Both sections assert that the license is a copyright license, and handle the acceptance of those copyright terms.

LGPLv2.1 §10 is equivalent to GPLv2 §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.

LGPLv2.1 §11 is GPLv2 §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.

LGPLv2.1 §12 adds the same features as GPLv2 §8. These sections are used to allow original copyright holders to forbid distribution in countries with draconian laws that would otherwise contradict these licenses.

LGPLv2.1 §13 sets up the FSF as the steward of the LGPL, just as GPLv2 §9 does for GPL. Meanwhile, LGPLv2.1 §14 reminds licensees that copyright holders can grant exceptions to the terms of LGPL, just as GPLv2 §10 reminds licensees of the same thing.

Finally, the assertions of no warranty and limitations of liability are identical; thus LGPLv2.1 §15 and LGPLv2.1 §16 are the same as GPLv2 §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 certain types of combinations, modifications and derivations. The LGPL does not stretch the requirements as far as copyright law does regarding what works must be relicensed under the same terms. Therefore, LGPL must in detail explain which works can 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.

10.3 Additions to the Preamble

Most of the 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 the LGPL.

In short, FSF designed the 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 the 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

The LGPL's preamble sets forth the limits to which the license seeks to go in chasing these goals. The 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 combined works: "works that use the library," and "works based on the library." Unlike the GPL, the LGPL must draw some lines regarding permissibly proprietary combined works. We do this here in this license because we specifically seek to liberalize the rights afforded to those who make combined works. In the GPL, we reach as far as copyright law allows. In the 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.

10.4 An Application: A Work that Uses the Library

In the effort to allow certain proprietary works and prohibit others, the LGPL distinguishes between two classes of works: "works based on the library," and "works that use the library." The distinction is drawn on the bright line of binary (or runtime) combined works and modified versions of source code. We will first consider the definition of a "work that uses the library," which is set forth in LGPLv2.1 §5.

We noted in our discussion of GPLv2 §3 (discussed in Section 5.2 of this document) that binary programs when compiled and linked with GPL'd software are covered as a whole by GPL. 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 works are controlled by the terms of the license (in GPLv2 §3), and distributors of such binary works must release full corresponding source.

The LGPL, by contrast, allows partial proprietarization of such binary works. 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 combined work, called $\mathcal{L}$+$\mathcal{I}$, that the user can run.
- Since $\mathcal{L}$+$\mathcal{I}$ is a based on 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 the LGPL does.

We will talk about the specific restrictions LGPLv2.1 places on "works that use the library" in detail in Section 10.7. For now, focus on the logic related to how the LGPLv2.1 places requirements on the license of $\mathcal{L}$+$\mathcal{I}$. Note, first of all, the similarity between this explanation and that in Section 5.1.2, which discussed the combination of otherwise separate and independent works with GPL'd code. Effectively, what LGPLv2.1 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 LGPLv2.1 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 work based on the LGPLv2.1'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 LGPLv2.1 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 combine into a single work with $\mathcal{L}$, either by static (compile-time) or dynamic (runtime) 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.

10.5 The Library, and Works Based On It

In short, a "work based on the library" could be defined as any work based on the 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 derivative, combined and/or modified works under copyright law, in the same sense that the 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 determining if a given work is a derivative or otherwise a modified version of another 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 and/or modified work of, and/or a combined work with the LGPL'd work under copyright law?
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."

You can also perform a similar same analysis through careful consideration of the license text itself. LGPLv2 §2(a) states that if a licensed work is a software library (defined in LGPLv2 §0 as "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"), you have permission to distribute modified versions only if those versions are themselves libraries. LGPLv2.1 code can therefore not be compliantly taken from its context in a library and placed in a non-library modified version or work based on the work. For its part, LGPLv2 §6 does not provide an exception for this rule: a combination may be made of a modified version of an LGPL'd library with other code, but the LGPL'd code must continue to be structured as a library, and to that library the terms of the license continue to apply.

Either way you view the rules, these issues are admittedly complicated. Nevertheless, In our years of work with the LGPLv2.1, 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 particularly large work in some ways seemed to be both to both a work that used the library and 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 LGPLv2.1 applied to each work as prescribed.

Finally, note though that, since the LGPLv2.1 can be easily upgraded to GPLv2-or-later, in the worst case you simply need to comply as if the software was licensed under GPLv2. The only reason you must consider the question of whether you have a "work that uses the library" or a "work based on the library" is when you wish to take advantage of the "weak copyleft" effect of the Lesser GPL. If GPLv2-or-later is an acceptable license (i.e., if you plan to copyleft the entire work anyway), you may find this an easier option.

10.6 Subtleties in Defining the Application

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 LGPLv2.1 §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 LGPLv2.1 §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 the 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 modified version of the 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. To understand this phenomenon, we consider that a "work that uses the library" is typically one whose final binary is a work based on the Program, but whose source is not. However, sometimes, there are cases where the object code — that intermediate step between source and final binary — is a work created by copying and modifying 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 final combined work 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 combined work is created.

LGPLv2.1 §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 LGPLv2.1'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 LGPLv2.1 are the same regardless of the details of the compilation progress.

10.7 LGPLv2.1 §6 & LGPLv2.1 §5: Combining the Works

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 LGPLv2.1 §6. LGPLv2.1 §6 is much like GPLv2 §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 LGPLv2.1 §6 to make sure that a user who wishes to modify or update the library can do so.

LGPLv2.1 §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 LGPLv2.1 compliance on a "work that uses the library."

LGPLv2.1 §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 LGPLv2.1 §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.

LGPLv2.1 §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.

Almost all known LGPL'd distributions exercise either LGPLv2.1 §6(a) or LGPLv2.1 §6(b). However, LGPLv2.1 §6 provides three other options. LGPLv2.1 §6(c) allows for a written offer for CCS (akin to GPLv2 §3(b)). CCS may also be distributed by network under the terms of LGPLv2.1 §6(c). Furthermore, under LGPLv2.1 §6(e) the distributor may "verify" that the user has already received, or at least that the distributor has already sent to this particular user, the relevant source[1].

Finally, LGPLv3 §6 also requires that:

> 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.

[1] Policy motivations for LGPLv2.1 §6(d) are unclear, but it presumably intended to prevent requiring duplicate deliveries in "whole distribution" situations.

This is not identical to the roughly parallel requirements of GPLv2 and GPLv3. Compliance requires slightly different measures with respect to the "credits" or "licenses" or "about" screens in interactive programs.

10.8 Distributing 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 LGPLv2.1 §2 is nearly identical in its terms and requirements to GPLv2 §2.

There are, however, subtle differences and additions. For example not only is CCS required (as would be with normal versions of GPL), but also the CCS provided must enable a developer to regenerate the modified version of the entire combined work, using with a modified version of the LGPL'd work (as a replacement for the version a distributor provided). For example, LGPL'd code is statically linked to a non-copyleft executable, the required source code must also include sufficient material to split the distributed executable and relink with a modified version of the library.

10.9 And the Rest

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

- Allowing a licensing "upgrade" from the LGPL to the GPL (in LGPLv2.1 §3). Note, however, LGPLv2.1 §3 allows relicensing of works under its terms instead under the terms of GPLv2-or-later. This provides, for example, a pathway for those who do not want to use code under the requirements of LGPLv2.1 to do so under GPLv2 or GPLv3 at their discretion.
- Binary distribution of the library only, covered in LGPLv2.1 §4, which is effectively equivalent to LGPLv2.1 §3
- Creating aggregates of libraries that are separate and independent works from each other, and distributing them as a unit (in LGPLv2.1 §7)

Due to time constraints, we cannot cover these additional terms in detail, but they are mostly straightforward. The key to understanding LGPLv2.1 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 LGPLv2.1 is close enough to GPL that the concepts discussed in our more extensive GPL unit can be directly applied.

CHAPTER 11

LGPLV3

LGPLv3 was designed to rectify architectural flaws in the GNU family of licenses. Historically , LGPLv2.1 was a textual modification of GPLv2. Reconciliation of licensing terms upon combination of LGPLv2.1'd and GPLv2'd works is cumbersome, from a licensing bookkeeping perspective.

LGPLv3 redresses this historical problem through extensive use of GPLv3 §7's exception architecture. LGPLv3 is therefore a set of additional permission to GPLv3.

11.1 Section 0: Additional Definitions

LGPLv3 §0 defines the "Library" – a work that presents one or more interfaces at which a "use" can be made by an "Application." Class inheritance is "deemed" a use of an interface. An "Application," which is other program code using one or more "Library" interfaces can be combined with the code on the other side of the interfaces it uses to form a "Combined Work."

11.2 LGPLv3 §1: Exception to GPLv3 §3

LGPLv3 §1 excepts away the interference with use of LGPLv3 code as part of "effective technological measures" of access limitation for other copyrighted works provided otherwise by GPLv3 §3.

11.3 LGPLv3 §2: Conveying Modified Versions

LGPLv3 §2 continues to require, as LGPLv2.1 §2(d) requires, that the Library not be modified to require keys, tokens, tables, or other global non-argument data unrelated to function. This is again stated as a "good faith effort" requirement, but failure to cure on notice is strong evidence of the absence of good faith. LGPLv3 §2(b) permits removal of the permissions entirely (as prescribed by GPLv3 §7); however, such removal reduces the license of the entire covered work back to pure GPLv3. Thus, exercising LGPLv3 §2(b) as a compliance alternative to LGPLv3 §2(a) likely creates more compliance obligations than it removes.

11.4 LGPLv3 §3: Object Code Incorporating Material from Library Header Files

LGPLv3 §3's front matter assures incorporation of smaller header files into non-copylefted object code can proceed unimpeded. More complex header files (those that do not meet the limitations provided in

the section), can still be incorporated into object code, a copy of appropriate licensing information must accompany distribution (per LGPLv3 §3(a–b).

11.5 LGPLv3 §4: Combined Works

LGPLv3 §4 is the combination permission at the heart of LGPLv3. It restates the license limitation provision of LGPLv2.1 §2 to clarify that the terms on the Combined Work may not prohibit user modification of the Library code, or the debugging of such modifications to the Library code by means of whatever reverse engineering is necessary.

LGPLv3 §4(d)(0) contains the source provision requirement, for the Minimal Corresponding Source, which "means the Corresponding Source for the Combined Work, excluding any source code for portions of the Combined Work that, considered in isolation, are based on the Application, and not on the Linked Version [of the Library]". The alternative to the provision of source code is distribution by way of the "shared library" mechanism under LGPLv3 §4(d)(1), described with respect to LGPLv2.1 §6.

In addition, LGPLv3 §4(e) requires the delivery of "installation information" required to install the modified version of the Library in "user products" under GPLv3 §6. Where Library Minimal Corresponding Source is not made available under LGPLv3 §4(d)(1), LGPLv3 §4(e) reaffirms that "installation information" must still be compliantly delivered under the terms of GPLv3 §6.

All other provisions of GPLv3 are in force as previously described, and are not excepted by the additional permission granted in LGPLv3.

If the distributor of the combined work intends not to distribute or offer the source code of the LGPL'd components, the LGPL'd work must be separately distributed (subject to source code delivery requirements as part of that separate distribution) and packaged in a "shared library" mechanism, which means that it:

4(d)(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

4(d)(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.

Taken all together, LGPLv3 §4's primary implications for redistributors are two-fold, as follows:

- If you create a program that links through a shared library mechanism to a work that is separately distributed under LGPLv3, then you can distribute the resultant program under a license of your choice and you need not convey the LGPLv3'd work's source code. If you distribute the library along with your program, or are the separate distributor of the work in another context or as another product, you must distribute its corresponding source under the terms of LGPLv3 or GPLv3-or-later.

- If you choose to statically link or otherwise combine your program with an LGPLv3'd work via mechanisms other than a shared library, you may choose your own license for the work provided the license terms limitations for user modification, reverse engineering and debugging are met, and given that the LGPL'd components are still governed by LGPL's terms. You must offer or provide CCS for the LGPL'd components. The source code material provided must be sufficient to regenerate the combined work with a user-modified version of the LGPL'd components.

CHAPTER 12

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.

12.1 Using GPL'd Software In-House

As discussed in Sections 3.1 and 7.2 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 the 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 GPLv2 §§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.

12.2 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 make substantial revenue from this model. Some choose this model because they have found that for higher-end hardware, 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, is the right choice. For instance IBM can be assured that proprietary versions of the their software will not exist to compete on their hardware.

For example, charging 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.

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 that compete in this space. Modern industry demands the trust created by GPL protected code-bases. 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 holds exclusive rights to relicense.[1] 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 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 at best problematic and at worst predatory 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.

12.3 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 clearly and legally separate and independent works, merely aggregated together on the same device.

However, it sometimes is easier, quicker, and cheaper to simply improve an existing GPL'd application than to start from scratch. In exchange for this amazing benefit, the license requires that the modifier gives 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 <licensing@fsf.org>.

If you are particularly interested in matters of GPL compliance, we recommend the next two parts, which include both recommendations on good compliance and compliance case studies.

[1] Entities typically hold exclusive relicensing rights either by writing all the software under their own copyrights, collecting copyright assignments from all contributors, or by otherwise demanding unconditional relicensing permissions from all contributors via some legal agreement

Part II

A Practical Guide to GPL Compliance

EXECUTIVE SUMMARY

This is a guide to effective compliance with the GNU General Public License (GPL) and related licenses. Copyleft advocates usually seek to assist the community with GPL compliance cooperatively. This guide focuses on complying from the start, so that readers can learn to avoid enforcement actions entirely, or, at least, minimize the negative impact when enforcement actions occur. This guide introduces and explains basic legal concepts related to the GPL and its enforcement by copyright holders. It also outlines business practices and methods that lead to better GPL compliance. Finally, it recommends proper post-violation responses to the concerns of copyright holders.

CHAPTER 13

BACKGROUND

Copyright law grants exclusive rights to authors. Authors who chose copyleft seek to protect the freedom of users and developers to copy, share, modify and redistribute the software. However, copyleft is ultimately implemented through copyright, and the GPL is primarily and by default a copyright license. (See § 1.2 for more about the interaction between copyright and copyleft.) Copyright law grants an unnatural exclusive control to copyright holders regarding copyright-controlled permissions related to the work. Therefore, copyright holders (or their agents) are the ultimately the sole authorities to enforce copyleft and protect the rights of users. Actions for copyright infringement are the ultimate legal mechanism for enforcement. Therefore, copyright holders, or collaborative groups of copyright holders, have historically been the actors in GPL enforcement.

The earliest of these efforts began soon after the GPL was written by Richard M. Stallman (RMS) in 1989, and consisted of informal community efforts, often in public Usenet discussions.[1] Over the next decade, the Free Software Foundation (FSF), which holds copyrights in many GNU programs, was the only visible entity actively enforcing its GPL'd copyrights on behalf of the software freedom community. FSF's enforcement was generally a private process; the FSF contacted violators confidentially and helped them to comply with the license. Most violations were pursued this way until the early 2000's.

By that time, Linux-based systems such as GNU/Linux and BusyBox/Linux had become very common, particularly in embedded devices such as wireless routers. During this period, public ridicule of violators in the press and on Internet fora supplemented ongoing private enforcement and increased pressure on businesses to comply. In 2003, the FSF formalized its efforts into the GPL Compliance Lab, increased the volume of enforcement, and built community coalitions to encourage copyright holders to together settle amicably with violators. Beginning in 2004, Harald Welte took a more organized public enforcement approach and launched gpl-violations.org, a website and mailing list for collecting reports of GPL violations. On the basis of these reports, Welte successfully pursued many enforcement actions in Europe, including formal legal action. Harald earns the permanent fame as the first copyright holder to bring legal action in a court regarding GPL compliance.

In 2007, two copyright holders in BusyBox, in conjunction with the Software Freedom Conservancy ("Conservancy"), filed the first copyright infringement lawsuit based on a violation of the GPL in the USA. While lawsuits are of course quite public, the vast majority of Conservancy's enforcement actions are resolved privately via cooperative communications with violators. As both FSF and Conservancy have worked to bring individual companies into compliance, both organizations have encountered numerous violations resulting from preventable problems such as inadequate attention to licensing of upstream software, misconceptions about the GPL's terms, and poor communication between software developers and their management. This document highlights these problems and describe best practices to encourage corporate Free Software users to reevaluate their approach to GPL'd software and avoid future violations.

[1] One example is the public outcry over NeXT's attempt to make the Objective-C front-end to GCC proprietary. RMS, in fact, handled this enforcement action personally and the Objective-C front-end is still part of upstream GCC today.

Both FSF and Conservancy continue GPL enforcement and compliance efforts for software under the GPL, the GNU Lesser Public License (LGPL) and other copyleft licenses. In doing so, both organizations have found that most violations stem from a few common, avoidable mistakes. All copyleft advocates hope to educate the community of commercial distributors, redistributors, and resellers on how to avoid violations in the first place, and to respond adequately and appropriately when a violation occurs.

13.1 Who Has Compliance Obligations?

All distributors of modified or unmodified versions of copylefted works unmodified versions of the works have compliance obligations. Common methods of modifying the works include innumerable common acts, such as:

- embedding those works as executable copies into a device,
- transferring a digital copy of executable copies to someone else,
- posting a patch to the copylefted software to a public mailing list.

Such distributors have obligations to (at least) the users to whom they (or intermediary parties) distribute those copies. In some cases, distributors have obligations to third parties not directly receiving their distribution of the works (depending on the distributors chosen licensing options, as described later in § 15.1). In addition, distributors have compliance obligations to upstream parties, such as preservation of reasonable legal notices embedded in the code, and appropriate labeling of modified versions.

Online service providers and distributors alike have other compliance obligations. In general, they must refrain from imposing any additional restrictions on downstream parties. Most typically, such compliance problems arise from "umbrella licenses:" EULAs, or sublicenses that restrict downstream users' rights under copyleft. (See § 7.3 and § 9.13).

Patent holders having claims reading on GPL'd works they distribute must refrain from enforcing those claims against parties to whom they distribute. Furthermore, patent holders holding copyrights on GPLv3'd works must further grant an explicit patent license for any patent claims reading on the version they distributed, and therefore cannot enforce those specific patent claims against anyone making, using or selling a work based on their distributed version. All parties must refrain from acting as a provider of services or distributor of licensed works if they have accepted, or had imposed on them by judicial action, any legal conditions that would prevent them from meeting any obligation under GPL. (See § 7.5, § 9.14 and § 9.15.

13.2 What Are The Risks of Non-Compliance?

Copyleft experts have for decades observed a significant mismatch between the assumptions most businesses make about copyleft compliance and the realities. Possibly due to excessive marketing of proprietary tools and services from the for-profit compliance industry, businesses perennially focus on the wrong concerns. This tutorial seeks to educate those businesses about what actually goes wrong, what causes disputes, and how to resolve those disputes.

Many businesses currently invest undue resources to avoid unlikely risks that have low historical incidence of occurrence and low cost of remediation, while leaving unmanaged the risks that have historically resulted in all the litigation and other adverse outcomes. For example, some "compliance industry"[2] vendors insist that great effort must be expended to carefully list, in the menus or manuals of embedded electronics products, copyright notices for every last copyright holder that contributed to the Free Software included

[2] "Compliance industry" refers to third-party for-profit companies that market proprietary software tools and/or consulting services that purport to aid businesses with their Free Software license compliance obligations, such as those found in GPL and other copyleft licenses. This tutorial leaves the term in quotes throughout, primarily to communicate the skepticism most of this tutorial's authors feel regarding the mere existence of this industry. Not only do copyleft advocates object on principle to proprietary software tools in general, and to their ironic use specifically to comply with copyleft, but also to the "compliance industry" vendors' marketing messaging, which some copyleft advocates claim as a cause in the risk misassessments discussed herein. Bradley M. Kuhn, specifically, regularly uses the term "compliance industrial complex" to analogize the types of problems in this industry to those warned against in the phrase of origin.

in the product. While nearly all Free Software licenses, including copylefts like GPL, require preservation and display of copyright notices, failure to meet this specific requirement is trivially remedied. Therefore, businesses should spend just reasonable efforts to properly display copyright notices, and note that failure to do so is simply remedied: add the missing copyright notice!

13.3 Understanding Who's Enforcing

The mismatch between actual compliance risk and compliance risk management typically results from a misunderstanding of licensor intentions. For-profit businesses often err by assuming other actors have kindred motivations. The primary enforcers of the GPL, however, have goals that for-profit businesses will find strange and perhaps downright alien.

Specifically, community-oriented GPL enforcement organizations (called "COGEOs" throughout the remainder of this tutorial) are typically non-profit charities (such as the FSF and Software Freedom Conservancy) who declare, as part of their charitable mission, advancement of software freedom for all users. In the USA, these COGEOs are all classified as charitable under the IRS's 501(c)(3) designation, which is reserved for organizations that have a mission to enhance the public good.

As such, these COGEOs enforce GPL primarily to pursue the policy goals and motivations discussed throughout this tutorial: to spread software freedom further. As such, COGEOs are unified in their primary goal to bring the violator back into compliance as quickly as possible, and redress the damage caused by the violation. COGEOs are steadfast in their position in a violation negotiation: comply with the license and respect freedom.

Certainly, other entities do not share the full ethos of software freedom as institutionalized by COGEOs, and those entities pursue GPL violations differently. Oracle, 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 a COGEO would undertake nor endorse, a copyleft license technically permits this behavior. To put a finer point on this practice already discussed in § 12.2, copyleft advocates usually find copyleft enforcement efforts focused on extract alternative proprietary licenses distasteful at best, and a corrupt manipulation of copyleft at worst. Much to the advocates' chagrin, such for-profit enforcement efforts seem to increase rather than decrease.

Thus, unsurprisingly, for-profit adopters of GPL'd software often incorrectly assume that all copyright holders seek royalties. Businesses therefore focus on the risk of so-called "accidental" (typically as the result of unsupervised activity by individual programmers) infringe copyright by incorporating "snippets" of copylefted code into their own proprietary computer program. "Compliance industry" flagship products, therefore, focus on "code scanning" services that purport to detect accidental inclusions. Such effort focuses on proprietary software development and view Free Software as a foreign interloper. Such approach not only ignores current reality that many companies build their products directly on major copylefted projects (e.g., Android vendor's use of the kernel named Linux), but also creates a culture of fear among developers, leading them into a downward spiral of further hiding their necessary reliance on copylefted software in the company's products.

Fortunately, COGEOs regard GPL compliance failures as an opportunity to improve compliance. Every compliance failure downstream represents a loss of rights by their users. The COGEOs are the guardian of its users' and developers' rights. Their activity seeks to restore those rights, and to protect the project's contributors' intentions in the making of their software.

CHAPTER 14

BEST PRACTICES TO AVOID COMMON VIOLATIONS

Unlike highly permissive licenses (such as the ISC license), which typically only require preservation of copyright notices, licensees face many important requirements from the GPL. These requirements are carefully designed to uphold certain values and standards of the software freedom community. While the GPL's requirements may initially appear counter-intuitive to those more familiar with proprietary software licenses, by comparison, its terms are in fact clear and quite favorable to licensees. Indeed, the GPL's terms actually simplify compliance when violations occur.

GPL violations occur (or, are compounded) most often when companies lack sound practices for the incorporation of GPL'd components into their internal development environment. This section introduces some best practices for software tool selection, integration and distribution, inspired by and congruent with software freedom methodologies. Companies should establish such practices before building a product based on GPL'd software.[1]

14.1 Evaluate License Applicability

Political discussion about the GPL often centers around determining the "work" that must be licensed under GPL, or in other words, "what is the derivative and/or combined work that was created". Nearly ever esoteric question asked by lawyers seek to consider that question [2] (perhaps because that question explores exciting legal issues while the majority of the GPL deals with much more mundane ones). Of course, GPL was designed primarily to embody the licensing feature of copyleft.

However, most companies who add complex features to and make combinations with GPL'd software are already well aware of their more complex obligations under the license that require complex legal analysis. And, there are few companies overall that engage in such activities. Thus, in practical reality, this issue is not relevant to the vast majority of companies distributing GPL'd software.

Thus, experienced GPL enforcers find that few redistributors' compliance challenges relate directly to combined work issues in copyleft. Instead, the distributions of GPL'd systems most often encountered typically consist of a full operating system including components under the GPL (e.g., Linux, BusyBox) and components under the LGPL (e.g., the GNU C Library). Sometimes, these programs have been patched or slightly improved by direct modification of their sources, and thus the result is unequivocally a modified version. Alongside these programs, companies often distribute fully independent, proprietary programs,

[1] This document addresses compliance with GPLv2, GPLv3, LGPLv2, and LGPLv3. Advice on avoiding the most common errors differs little for compliance with these four licenses. § 18.1 discusses the key differences between GPL and LGPL compliance.

[2] This tutorial in fact also addresses the issue at length in § 14.1.

developed from scratch, which are designed to run on the Free Software operating system but do not combine with, link to, modify, derive from, or otherwise create a combined work with the GPL'd components.[3] In the latter case, where the work is unquestionably a separate work of creative expression, no copyleft provisions are invoked. The core compliance issue faced, thus, in such a situation, is not an discussion of what is or is not a combined, derivative, and/or modified version of the work, but rather, issues related to distribution and conveyance of binary works based on GPL'd source, but without Complete, Corresponding Source.

As such, issues of software delivery are the primary frustration for GPL enforcers. In particular, the following short list accounts for at least 95% of the GPL violations ever encountered:

- The violator fails to provide required information about the presence of copylefted programs and their applicable license terms in the product they have purchased.
- The violator fails to reliably deliver complete, corresponding source (CCS) for copylefted programs the violator knew were included (i.e., the CCS is either delivered but incomplete, or is not delivered at all).
- Requestors are ignored when they communicate with violator's published addresses requesting fulfillment of businesses' obligations.

This tutorial therefore focuses primarily on these issue. Admittedly, a tiny minority of compliance situations relate to question of derivative, combined, or modified versions of the work. Those situations are so rare, and the details from situation to situation differ greatly. Thus, such situations require a highly fact-dependent analysis and cannot be addressed in a general-purpose document such as this one.

Most companies accused of violations lack a basic understanding of how to comply even in the straightforward scenario. This document provides those companies with the fundamental and generally applicable prerequisite knowledge. For answers to rarer and more complicated legal questions, such as whether your software is a derivative or combined work of some copylefted software, consult with an attorney.[4]

This discussion thus assumes that you have already identified the "work" covered by the license, and that any components not under the GPL (e.g., applications written entirely by your developers that merely happen to run on a Linux-based operating system) distributed in conjunction with those works are separate works within the meaning of copyright law and the GPL. In such a case, the GPL requires you to provide complete corresponding source (CCS)[5] for the GPL'd components and your modifications thereto, but not for independent proprietary applications. The procedures described in this document address this typical scenario.

14.2 Monitor Software Acquisition

Software engineers deserve the freedom to innovate and import useful software components to improve products. However, along with that freedom should come rules and reporting procedures to make sure that you are aware of what software that you include with your product.

The most typical response to an initial enforcement action is: "We didn't know there was GPL'd stuff in there". This answer indicates failure in the software acquisition and procurement process. Integration of third-party proprietary software typically requires a formal arrangement and management/legal oversight before the developers incorporate the software. By contrast, developers often obtain and integrate Free Software without intervention nor oversight. That ease of acquisition, however, does not mean the oversight is any less necessary. Just as your legal and/or management team negotiates terms for inclusion of any proprietary software, they should gently facilitate all decisions to bring Free Software into your product.

Simple, engineering-oriented rules help provide a stable foundation for Free Software integration. For example, simply ask your software developers to send an email to a standard place describing each new Free Software component they add to the system, and have them include a brief description of how they will

[3] However, these programs do often combine with LGPL'd libraries. This is discussed in detail in § 18.1.

[4] If you would like more information on the application of derivative works doctrine to software, a detailed legal discussion is presented in our colleague Dan Ravicher's article, *Software Derivative Work: A Circuit Dependent Determination* and in § 14.1 of this tutorial.

[5] For more on CCS, see § 5.1 and § 9.3 of this tutorial.

incorporate it into the product. Further, make sure developers use a revision control system (such as Git or Mercurial), and store the upstream versions of all software in a "vendor branch" or similar mechanism, whereby they can easily track and find the main version of the software and, separately, any local changes.

Such procedures are best instituted at your project's launch. Once chaotic and poorly-sourced development processes begin, cataloging the presence of GPL'd components becomes challenging.

Such a situation often requires use of a tool to "catch up" your knowledge about what software your product includes. Most commonly, companies choose some software licensing scanning tool to inspect the codebase. However, there are few tools that are themselves Free Software. Thus, GPL enforcers usually recommend the GPL'd FOSSology system, which analyzes a source code base and produces a list of Free Software licenses that may apply to the code. FOSSology can help you build a catalog of the sources you have already used to build your product. You can then expand that into a more structured inventory and process.

14.3 Track Your Changes and Releases

As explained in further detail below, the most important component of GPL compliance is the one most often ignored: proper inclusion of CCS in all distributions of GPL'd software. To comply with GPL's CCS requirements, the distributor *must* always know precisely what sources generated a given binary distribution.

In an unfortunately large number of our enforcement cases, the violating company's engineering team had difficulty reconstructing the CCS for binaries distributed by the company. Here are three simple rules to follow to decrease the likelihood of this occurrence:

- Ensure that your developers are using revision control systems properly.
- Have developers mark or "tag" the full source tree corresponding to builds distributed to customers.
- Check that your developers store all parts of the software development in the revision control system, including README*s*, build scripts, engineers' notes, and documentation.

Your developers will benefit anyway from these rules. Developers will be happier in their jobs if their tools already track the precise version of source that corresponds to any deployed binary.

14.4 Avoid the "Build Guru"

Too many software projects rely on only one or a very few team members who know how to build and assemble the final released product. Such knowledge centralization not only creates engineering redundancy issues, but also thwarts GPL compliance. Specifically, CCS does not just require source code, but scripts and other material that explain how to control compilation and installation of the executable and object code.

Thus, avoid relying on a "build guru", a single developer who is the only one who knows how to produce your final product. Make sure the build process is well defined. Train every developer on the build process for the final binary distribution, including (in the case of embedded software) generating a final firmware image suitable for distribution to the customer. Require developers to use revision control for build processes. Make a rule that adding new components to the system without adequate build instructions (or better yet, scripts) is unacceptable engineering practice.

CHAPTER 15

DETAILS OF COMPLIANT DISTRIBUTION

Distribution of GPL'd works has requirements; copyleft will not function without placing requirements on redistribution. However, some requirements are more likely to cause compliance difficult than others. This chapter[1] explains some the specific requirements placed upon distributors of GPL'd software that redistributors are most likely to overlook, yielding compliance problems.

First, GPLv2§1 and GPLv2§4 require that the full license text must accompany every distribution (either in source or binary form) of each licensed work. Strangely, this requirement is responsible for a surprisingly significant fraction of compliance errors; too often, physical products lack required information about the presence of GPL'd programs and the applicable license terms. Automated build processes can and should carry a copy of the license from the the source distribution into the final binary firmware package for embedded products. Such automation usually achieves compliance regarding license inclusion requirements[2]

15.1 Binary Distribution Permission

The various versions of the GPL are copyright licenses that grant permission to make certain uses of software that are otherwise restricted by copyright law. This permission is conditioned upon compliance with the GPL's requirements.

This section walks through the requirements (of both GPLv2 and GPLv3) that apply when you distribute GPL'd programs in binary (i.e., executable or object code) form, which is typical for embedded applications. Because a binary application derives from a program's original sources, you need permission from the copyright holder to distribute it. § 3 of GPLv2 and § 6 of GPLv3 contain the permissions and conditions related to binary distributions of GPL'd programs.[3] Failure to provide or offer CCS is the single largest failure mode leading to compliance disputes.

GPL's binary distribution sections offer a choice of compliance methods, each of which we consider in turn. Each option refers to the "Corresponding Source" code for the binary distribution, which includes the source code from which the binary was produced. This abbreviated and simplified definition is sufficient for the binary distribution discussion in this section, but you may wish to refer back to this section after reading the thorough discussion of "Corresponding Source" that appears in § 15.2.

[1] Note that this chapter refers heavily to specific provisions and language in GPLv2§3 and GPLv3§6. It may be helpful to review § 5.2 and § 9.9 first, and then have a copy of each license open while reading this section.

[2] At least one COGEO recommends the Yocto Project, since its engineers have designed such features into it build process.

[3] These sections cannot be fully understood in isolation; read the entire license thoroughly before focusing on any particular provision. However, once you have read and understood the entire license, look to these sections to guide compliance for binary distributions.

15.1.1 Option (a): Source Alongside Binary

GPLv2 § 3(a) and v3 § 6(a) embody the easiest option for providing source code: including Corresponding Source with every binary distribution. While other options appear initially less onerous, this option invariably minimizes potential compliance problems, because when you distribute Corresponding Source with the binary, *your GPL obligations are satisfied at the time of distribution.* This is not true of other options, and for this reason, we urge you to seriously consider this option. If you do not, you may extend the duration of your obligations far beyond your last binary distribution.

Compliance under this option is straightforward. If you ship a product that includes binary copies of GPL'd software (e.g., in firmware, or on a hard drive, CD, or other permanent storage medium), you can store the Corresponding Source alongside the binaries. Alternatively, you can include the source on a CD or other removable storage medium in the box containing the product.

GPLv2 refers to the various storage mechanisms as "medi[a] customarily used for software interchange". While the Internet has attained primacy as a means of software distribution where super-fast Internet connections are available, GPLv2 was written at a time when downloading software was not practical (and was often impossible). For much of the world, this condition has not changed since GPLv2's publication, and the Internet still cannot be considered "a medium customary for software interchange". GPLv3 clarifies this matter, requiring that source be "fixed on a durable physical medium customarily used for software interchange". This language affirms that option (a) requires binary redistributors to provide source on a physical medium.

Please note that while selection of option (a) requires distribution on a physical medium, voluntary distribution via the Internet is very useful. This is discussed in detail in § 15.1.2.

15.1.2 Option (b): The Offer

Many distributors prefer to ship only an offer for source with the binary distribution, rather than the complete source package. This option has value when the cost of source distribution is a true per-unit cost. For example, this option might be a good choice for embedded products with permanent storage too small to fit the source, and which are not otherwise shipped with a CD but *are* shipped with a manual or other printed material.

However, this option increases the duration of your obligations dramatically. An offer for source must be good for three full years from your last binary distribution (under GPLv2), or your last binary or spare part distribution (under GPLv3). Your source code request and provisioning system must be designed to last much longer than your product life cycle. Thus, it also increases your compliance costs in the long run.

In addition, if you are required to comply with the terms of GPLv2, you **cannot** use a network service to provide the source code. For GPLv2, the source code offer is fulfilled only with physical media. This usually means that you must continue to produce an up-to-date "source code CD" for years after the product's end-of-life.

Under GPLv2, it is acceptable and advisable for your offer for source code to include an Internet link for downloadable source *in addition* to offering source on a physical medium. This practice enables those with fast network connections to get the source more quickly, and typically decreases the number of physical media fulfillment requests. (GPLv3 § 6(b) permits provision of source with a public network-accessible distribution only and no physical media. We discuss this in detail at the end of this section.)

The following is a suggested compliant offer for source under GPLv2 (and is also acceptable for GPLv3) that you would include in your printed materials accompanying each binary distribution:

> The software included in this product contains copyrighted software that is licensed under the GPL. A copy of that license is included in this document on page *X*. You may obtain the complete Corresponding Source code from us for a period of three years after our last shipment of this product, which will be no earlier than 2011-08-01, by sending a money order or check for $5 to:
> GPL Compliance Division
> Our Company
> Any Town, US 99999

> Please write "source for product Y" in the memo line of your payment.
>
> You may also find a copy of the source at `http://www.example.com/sources/Y/`.
>
> This offer is valid to anyone in receipt of this information.

There are a few important details about this offer. First, it requires a copying fee. GPLv2 permits "a charge no more than your cost of physically performing source distribution". This fee must be reasonable. If your cost of copying and mailing a CD is more than around $10, you should perhaps find a cheaper CD stock and shipment method. It is simply not in your interest to try to overcharge the community. Abuse of this provision in order to make a for-profit enterprise of source code provision will likely trigger enforcement action.

Second, note that the last line makes the offer valid to anyone who requests the source. This is because v2 § 3(b) requires that offers be "to give any third party" a copy of the Corresponding Source. GPLv3 has a similar requirement, stating that an offer must be valid for "anyone who possesses the object code". These requirements indicated in v2 § 3(c) and v3 § 6(c) are so that noncommercial redistributors may pass these offers along with their distributions. Therefore, the offers must be valid not only to your customers, but also to anyone who received a copy of the binaries from them. Many distributors overlook this requirement and assume that they are only required to fulfill a request from their direct customers.

The option to provide an offer for source rather than direct source distribution is a special benefit to companies equipped to handle a fulfillment process. GPLv2 § 3(c) and GPLv3 § 6(c) avoid burdening noncommercial, occasional redistributors with fulfillment request obligations by allowing them to pass along the offer for source as they received it.

Note that commercial redistributors cannot avail themselves of the option (c) exception, and so while your offer for source must be good to anyone who receives the offer (under v2) or the object code (under v3), it *cannot* extinguish the obligations of anyone who commercially redistributes your product. The license terms apply to anyone who distributes GPL'd software, regardless of whether they are the original distributor. Take the example of Vendor V, who develops a software platform from GPL'd sources for use in embedded devices. Manufacturer M contracts with V to install the software as firmware in M's device. V provides the software to M, along with a compliant offer for source. In this situation, M cannot simply pass V's offer for source along to its customers. M also distributes the GPL'd software commercially, so M too must comply with the GPL and provide source (or M's *own* offer for source) to M's customers.

This situation illustrates that the offer for source is often a poor choice for products that your customers will likely redistribute. If you include the source itself with the products, then your distribution to your customers is compliant, and their (unmodified) distribution to their customers is likewise compliant, because both include source. If you include only an offer for source, your distribution is compliant but your customer's distribution does not "inherit" that compliance, because they have not made their own offer to accompany their distribution.

The terms related to the offer for source are quite different if you distribute under GPLv3. Under v3, you may make source available only over a network server, as long as it is available to the general public and remains active for three years from the last distribution of your product or related spare part. Accordingly, you may satisfy your fulfillment obligations via Internet-only distribution. This makes the "offer for source" option less troublesome for v3-only distributions, easing compliance for commercial redistributors. However, before you switch to a purely Internet-based fulfillment process, you must first confirm that you can actually distribute *all* of the software under GPLv3. Some programs are indeed licensed under "GPLv2, *or any later version*" (often abbreviated "GPLv2-or-later"). Such licensing gives you the option to redistribute under GPLv3. However, a few popular programs are only licensed under GPLv2 and not "or any later version" ("GPLv2-only"). You cannot provide only Internet-based source request fulfillment for the latter programs.

If you determine that all GPL'd works in your whole product allow upgrade to GPLv3 (or were already GPLv3'd to start), your offer for source may be as simple as this:

> The software included in this product contains copyrighted software that is licensed under the GPLv3. A copy of that license is included in this document on page X. You may obtain the complete Corresponding Source code from us for a period of three years after our last shipment

> of this product and/or spare parts therefor, which will be no earlier than 2011-08-01, on our website at `http://www.example.com/sources/productnum/`.

Under both GPLv2 and GPLv3, source offers must be accompanied by a copy of the license itself, either electronically or in print, with every distribution.

Finally, it is unacceptable to use option (b) merely because you do not have Corresponding Source ready. We find that some companies choose this option because writing an offer is easy, but producing a source distribution as an afterthought to a hasty development process is difficult. The offer for source does not exist as a stop-gap solution for companies rushing to market with an out-of-compliance product. If you ship an offer for source with your product but cannot actually deliver *immediately* on that offer when your customers request it, you should expect an enforcement action.

15.1.3 Option (c): Noncommercial Offers

As discussed in the last section, GPLv2 § 3(c) and GPLv3 § 6(c) apply only to noncommercial use. These options are not available to businesses distributing GPL'd software. Consequently, companies that redistribute software packaged for them by an upstream vendor cannot merely pass along the offer they received from the vendor; they must provide their own offer or corresponding source to their distributees. We talk in detail about upstream software providers in § 18.2.

15.1.4 Option 6(d) in GPLv3: Internet Distribution

Under GPLv2, your formal provisioning options for Corresponding Source ended with § 3(c). But even under GPLv2, pure Internet source distribution was a common practice and generally considered to be compliant. GPLv2 mentions Internet-only distribution almost as aside in the language, in text at the end of the section after the three provisioning options are listed. To quote that part of GPLv2 § 3:

> 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.

When that was written in 1991, Internet distribution of software was the exception, not the rule. Some FTP sites existed, but generally software was sent on magnetic tape or CDs. GPLv2 therefore mostly assumed that binary distribution happened on some physical media. By contrast, GPLv3 § 6(d) explicitly gives an option for this practice that the community has historically considered GPLv2-compliant.

Thus, you may fulfill your source-provision obligations by providing the source code in the same way and from the same location. When exercising this option, you are not obligated to ensure that users download the source when they download the binary, and you may use separate servers as needed to fulfill the requests as long as you make the source as accessible as the binary. However, you must ensure that users can easily find the source code at the time they download the binary. GPLv3 § 6(d) thus clarifies a point that has caused confusion about source provision in v2. Indeed, many such important clarifications are included in v3 which together provide a compelling reason for authors and redistributors alike to adopt GPLv3.

15.1.5 Option 6(e) in GPLv3: Software Torrents

Peer-to-peer file sharing arose well after GPLv2 was written, and does not easily fit any of the v2 source provision options. GPLv3 § 6(e) addresses this issue, explicitly allowing for distribution of source and binary together on a peer-to-peer file sharing network. If you distribute solely via peer-to-peer networks, you can exercise this option. However, peer-to-peer source distribution *cannot* fulfill your source provision obligations for non-peer-to-peer binary distributions. Finally, you should ensure that binaries and source are equally seeded upon initial peer-to-peer distribution.

15.2 Preparing Corresponding Source

Most enforcement cases involve companies that have unfortunately not implemented procedures like our § 14 recommendations and have no source distribution arranged at all. These companies must work backwards from a binary distribution to come into compliance. Our recommendations in § 14 are designed to make it easy to construct a complete and Corresponding Source release from the outset. If you have followed those principles in your development, you can meet the following requirements with ease. If you have not, you may have substantial reconstruction work to do.

15.2.1 Assemble the Sources

For every binary that you produce, you should collect and maintain a copy of the sources from which it was built. A large system, such as an embedded firmware, will probably contain many GPL'd and LGPL'd components for which you will have to provide source. The binary distribution may also contain proprietary components which are separate and independent works that are covered by neither the GPL nor LGPL.

The best way to separate out your sources is to have a subdirectory for each component in your system. You can then easily mark some of them as required for your Corresponding Source releases. Collecting subdirectories of GPL'd and LGPL'd components is the first step toward preparing your release.

15.2.2 Building the Sources

Few distributors, particularly of embedded systems, take care to read the actual definition of Corresponding Source in the GPL. Consider carefully the definition, from GPLv3:

> The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities.

and the definition from GPLv2:

> 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.

Note that you must include "scripts used to control compilation and installation of the executable" and/or anything "needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities". These phrases are written to cover different types of build environments and systems. Therefore, the details of what you need to provide with regard to scripts and installation instructions vary depending on the software details. You must provide all information necessary such that someone generally skilled with computer systems could produce a binary similar to the one provided.

Take as an example an embedded wireless device. Usually, a company distributes a firmware, which includes a binary copy of Linux[4] and a filesystem. That filesystem contains various binary programs, including some GPL'd binaries, alongside some proprietary binaries that are separate works (i.e., not derived from, nor based on freely-licensed sources). Consider what, in this case, constitutes adequate "scripts to control compilation and installation" or items "needed to generate, install and run" the GPL'd programs.

Most importantly, you must provide some sort of roadmap that allows technically sophisticated users to build your software. This can be complicated in an embedded environment. If your developers use scripts to control the entire compilation and installation procedure, then you can simply provide those scripts to users along with the sources they act upon. Sometimes, however, scripts were never written (e.g., the information on how to build the binaries is locked up in the mind of your "build guru"). In that case, we recommend that you write out build instructions in a natural language as a detailed, step-by-step README.

No matter what you offer, you need to give those who receive source a clear path from your sources to binaries similar to the ones you ship. If you ship a firmware (kernel plus filesystem), and the filesystem

[4] "Linux" refers only to the kernel, not the larger system as a whole.

contains binaries of GPL'd programs, then you should provide whatever is necessary to enable a reasonably skilled user to build any given GPL'd source program (and modified versions thereof), and replace the given binary in your filesystem. If the kernel is Linux, then the users must have the instructions to do the same with the kernel. The best way to achieve this is to make available to your users whatever scripts or process your engineers would use to do the same.

These are the general details for how installation instructions work. Details about what differs when the work is licensed under LGPL is discussed in § 18.1, and specific details that are unique to GPLv3's installation instructions are in § 18.4.

15.2.3 What About the Compiler?

The GPL contains no provision that requires distribution of the compiler used to build the software. While companies are encouraged to make it as easy as possible for their users to build the sources, inclusion of the compiler itself is not normally considered mandatory. The Corresponding Source definition – both in GPLv2 and GPLv3 – has not been typically read to include the compiler itself, but rather things like makefiles, build scripts, and packaging scripts.

Nonetheless, in the interest of goodwill and the spirit of the GPL, most companies do provide the compiler itself when they are able, particularly when the compiler is based on GCC or another copylefted compiler. If you have a GCC-based system, it is your prerogative to redistribute that GCC version (binaries plus sources) to your customers. We in the software freedom community encourage you to do this, since it often makes it easier for users to exercise their software freedom. However, if you chose to take this recommendation, ensure that your GCC distribution is itself compliant.

If you have used a proprietary, third-party compiler to build the software, then you probably cannot ship it to your customers. We consider the name of the compiler, its exact version number, and where it can be acquired as information that *must* be provided as part of the Corresponding Source. This information is essential to anyone who wishes to produce a binary. It is not the intent of the GPL to require you to distribute third-party software tools to your customer (provided the tools themselves are not based on the GPL'd software shipped), but we do believe it requires that you give the user all the essential non-proprietary facts that you had at your disposal to build the software. Therefore, if you choose not to distribute the compiler, you should include a README about where you got it, what version it was, and who to contact to acquire it, regardless of whether your compiler is Free Software, proprietary, or internally developed.

15.3 Best Practices and Corresponding Source

§ 14 and § 15.2 above are closely related. If you follow the best practices outlined above, you will find that preparing your Corresponding Source release is an easier task, perhaps even a trivial one.

Indeed, the enforcement process itself has historically been useful to software development teams. Development on a deadline can lead organizations to cut corners in a way that negatively impacts its development processes. We have frequently been told by violators that they experience difficulty when determining the exact source for a binary in production (in some cases because their "build guru" quit during the release cycle). When management rushes a development team to ship a release, they are less likely to keep release sources tagged and build systems well documented.

We suggest that, if contacted about a violation, product builders use GPL enforcement as an opportunity to improve their development practices. No developer would argue that their system is better for having a mysterious build system and no source tracking. Address these issues by installing a revision system, telling your developers to use it, and requiring your build guru to document his or her work!

15.4 Non-Technical Compliance Issues

Certainly, the overwhelming majority of compliance issues are, in fact, either procedural or technical. Thus, the primary material in this chapter so far has covered those issues. However, a few compliance issues do require more direct consideration of a legal situation. This portion guide does not consider those in detail,

as a careful reading of the earlier chapters of Part I shows various places where legal considerations are necessary for considering compliance activity.

For example, specific compliance issues related to GPLv2§7, GPLv3§7, and GPLv3§11 demand a more traditional approach to legal license compliance. Of course, such analysis and consideration can be complicated, and some are considered in the enforcement case studies that follow in the next part. However, compliance issues related to such sections are not rare, and, as is typical, no specific training is available for dealing with extremely rare occurrences.

15.5 Self-Assessment of Compliance

Most companies that adopt copylefted software believe they have complied. Humans usually have difficult admitting their own mistakes, particularly systematic ones. Therefore, perhaps the most important necessary step to stay in compliance is a company's regular evaluation of their own compliance.

First, exercise a request CCS for all copylefted works from all your upstream providers of software and of components embedding software. Then, perform your own CCS check on this material first, and verify that it meets the requirements. This tutorial presents later a case study of a COGEO's CCS check in § 21, which you can emulate when examining their own CCS.

Second, measure all copyleft compliance from the position of the users[5] downstream from you exercising their rights under GPL. Have those users received notice of the copylefted software included in your product? Is CCS available to the users easily (preferably by automated means)? Ask yourself these questions frequently. If you cannot answer these questions with certainty in the positive, dig deeper and modify your process.

Avoid "compliance industry" marketing distractions and concentrate on the copylefted software you already know is in your product. Historically, the risk from a copylefted code snippet that some programmer dropped in your proprietary product careless of the consequences is a problem far more infrequent and less difficult to resolve. Efficient management of the risks of higher concern lies in making sure you can provide, for example, precisely CCS for a copy of Coreboot, the kernel named Linux, BusyBox, or GNU tar that you included in a product your company shipped two years ago than in the risk of 10 lines of GPL'd Java code an engineer accidentally pasted into the source of your ERP system.

Thus, reject the "compliance industry" suggestions that code scanners find and help solve fundamental compliance problems. Consider how COGEO's tend to use code scanners. FOSSology is indeed an important part of a violation investigation, but such is the last step and catches only some (usually minor) licensing notice problems. Thus, code scanners can help solve minor compliance problems once you have resolved the major ones. Code scanners do not manage risk.

[5]Realizing of course that user very well may not be your own customer.

CHAPTER 16

WHEN THE LETTER COMES

Unfortunately, many GPL violators ignore their obligations until they are contacted by a copyright holder or the lawyer of a copyright holder. You should certainly contact your own lawyer if you have received a letter alleging that you have infringed copyrights that were licensed to you under the GPL. This section outlines a typical enforcement case and provides some guidelines for response. These discussions are generalizations and do not all apply to every alleged violation. However, COGEO's in particular universally follow the processes described herein.

16.1 Communication Is Key

GPL violations are typically only escalated when a company ignores the copyright holder's initial communication or fails to work toward timely compliance. Accused violators should respond very promptly to the initial request. As the process continues, violators should follow up weekly with the copyright holders to make sure everyone agrees on targets and deadlines for resolving the situation.

Ensure that any staff who might receive communications regarding alleged GPL violations understands how to channel the communication appropriately within your organization. Often, initial contact is addressed for general correspondence (e.g., by mail to corporate headquarters or by e-mail to general informational or support-related addresses). Train the staff that processes such communications to escalate them to someone with authority to take action. An uninformed response to such an inquiry (e.g., from a first-level technical support person) can cause negotiations to fail prematurely.

Answer promptly by multiple means (paper letter, telephone call, and email), even if your response merely notifies the sender that you are investigating the situation and will respond by a certain date. Do not let the conversation lapse until the situation is fully resolved. Proactively follow up with synchronous communication means to be sure communications sent by non-reliable means (such as email) were received.

Remember that the software freedom community generally values open communication and cooperation, and these values extend to GPL enforcement. You will generally find that software freedom developers and their lawyers are willing to have a reasonable dialogue and will work with you to resolve a violation once you open the channels of communication in a friendly way.

Furthermore, if the complaint comes from a COGEO, assume they are well-prepared. COGEO's fully investigate compliance issues before raising the issue. The claims and concerns will be substantiated, and immediate denials will likely lead the COGEO to suspect malice rather than honest mistake.

However, the biggest and most perennial mistake that all COGEOs see during enforcement is this: failure to include the violators' software development teams in the enforcement discussions and negotiations. As described above, CCS verification and approval is the most time-consuming and difficult part of resolving most compliance matters. Without direct contact between software developers on both sides, the resolution of the technical issues involved in demonstrating that the binary distributed was built from the source

provided is likely to be tortuous, expensive, and tense. Your lawyers will certainly be understandably reluctant to expose your employees to direct inquiry from potentially adverse parties. However, facilitated exchanges of information among software engineers communicating on technical subjects shortens the time to resolution, substantially reduces the cost of reaching resolution, and prevents unnecessary escalation due to mutual misunderstanding. Furthermore, such frank technical discussion will often be the only way to avoid compliance litigation once a violation has occurred.

Fortunately, these frank discussions will improve your company's relationships. Free Software development communities improve software to benefit everyone, which includes you and your company. When you use copylefted community software in your products, you are part of that community. Therefore, resolving a compliance matter is an occasion to strengthen your relationship to the community, by increasing communication between your developers and the project whose work you use for business benefit.

16.2 Termination

Many redistributors overlook the GPL's termination provision (GPLv2 § 4 and GPLv3 § 8). Under v2, violators forfeit their rights to redistribute and modify the GPL'd software until those rights are explicitly reinstated by the copyright holder. In contrast, v3 allows violators to rapidly resolve some violations without consequence.

If you have redistributed an application under GPLv2[1], but have violated the terms of GPLv2, you must request a reinstatement of rights from the copyright holders before making further distributions, or else cease distribution and modification of the software forever. Different copyright holders condition reinstatement upon different requirements, and these requirements can be (and often are) wholly independent of the GPL. The terms of your reinstatement will depend upon what you negotiate with the copyright holder of the GPL'd program.

Since your rights under GPLv2 terminate automatically upon your initial violation, *all your subsequent distributions* are violations and infringements of copyright. Therefore, even if you resolve a violation on your own, you must still seek a reinstatement of rights from the copyright holders whose licenses you violated, lest you remain liable for infringement for even compliant distributions made subsequent to the initial violation.

GPLv3 is more lenient. If you have distributed only v3-licensed programs, you may be eligible under v3 § 8 for automatic reinstatement of rights. You are eligible for automatic reinstatement when:

- you correct the violation and are not contacted by a copyright holder about the violation within sixty days after the correction, or
- you receive, from a copyright holder, your first-ever contact regarding a GPL violation, and you correct that violation within thirty days of receipt of copyright holder's notice.

In addition to these permanent reinstatements provided under v3, violators who voluntarily correct their violation also receive provisional permission to continue distributing until they receive contact from the copyright holder. If sixty days pass without contact, that reinstatement becomes permanent. Nonetheless, you should be prepared to cease distribution during those initial sixty days should you receive a termination notice from the copyright holder.

Given that much discussion of v3 has focused on its so-called more complicated requirements, it should be noted that v3 is, in this regard, more favorable to violators than v2.

However, note that most Linux-based systems typically include some software licensed under GPLv2-only, and thus the copyright holders have withheld permission to redistribute under terms of GPLv3. In larger aggregate distributions which include GPLv2-only works (such as the kernel named Linux), redistributors must operate as if termination is immediate and permanent, since the technological remove of GPLv2-only works from the larger distribution requires much more engineering work than the negotiation required to seek restoration of rights for distribution under GPLv2-only after permanent termination.

[1] This applies to all programs licensed to you under only GPLv2 ("GPLv2-only"). However, most so-called GPLv2 programs are actually distributed with permission to redistribute under GPLv2 *or any later version of the GPL* ("GPLv2-or-later"). In the latter cases, the redistributor can choose to redistribute under GPLv2, GPLv3, GPLv2-or-later or even GPLv3-or-later. Where the redistributor has chosen v2 explicitly, the v2 termination provision will always apply. If the redistributor has chosen v3, the v3 termination provision will always apply. If the redistributor has chosen GPLv2-or-later, then the redistributor may want to narrow to GPLv3-only upon violation, to take advantage of the termination provisions in v3.

CHAPTER 17

STANDARD REQUESTS

As we noted above, different copyright holders have different requirements for reinstating a violator's distribution rights. Upon violation, you no longer have a license under the GPL. Copyright holders can therefore set their own requirements outside the license before reinstatement of rights. We have collected below a list of reinstatement demands that copyright holders often require.

- **Compliance on all Free Software copyrights**. Copyright holders of Free Software often want a company to demonstrate compliance for all GPL'd software in a distribution, not just their own. A copyright holder may refuse to reinstate your right to distribute one program unless and until you comply with the licenses of all Free Software in your distribution.
- **Notification to past recipients**. Users to whom you previously distributed non-compliant software should receive a communication (email, letter, bill insert, etc.) indicating the violation, describing their rights under the GPL, and informing them how to obtain a gratis source distribution. If a customer list does not exist (such as in reseller situations), an alternative form of notice may be required (such as a magazine advertisement).
- **Appointment of a GPL Compliance Officer.** The software freedom community values personal accountability when things go wrong. Copyright holders often require that you name someone within the violating company officially responsible for Free Software license compliance, and that this individual serve as the key public contact for the community when compliance concerns arise.
- **Periodic Compliance Reports.** Many copyright holders wish to monitor future compliance for some period of time after the violation. For some period, your company may be required to send regular reports on how many distributions of binary and source have occurred.

These are just a few possible requirements for reinstatement. In the context of a GPL violation, and particularly under v2's termination provision, the copyright holder may have a range of requests in exchange for reinstatement of rights. These software developers are talented professionals from whose work your company has benefited. Indeed, you are unlikely to find a better value or more generous license terms for similar software elsewhere. Treat the copyright holders with the same respect you treat your corporate partners and collaborators.

CHAPTER 18

SPECIAL TOPICS IN COMPLIANCE

There are several other issues that are less common, but also relevant in a GPL compliance situation. To those who face them, they tend to be of particular interest.

18.1 LGPL Compliance

GPL compliance and LGPL compliance mostly involve the same issues. As we discussed in § 14.1, questions of modified versions of software are highly fact-dependent and cannot be easily addressed in any overview document. The LGPL adds some additional complexity to the analysis. Namely, the various LGPL versions permit proprietary licensing of certain types of modified versions. These issues are discussed in greater detail in Chapter 10 and 11. However, as a rule of thumb, once you have determined (in accordance with LGPLv3) what part of the work is the "Application" and what portions of the source are "Minimal Corresponding Source", then you can usually proceed to follow the GPL compliance rules that discussed above, replacing our discussion of "Corresponding Source" with "Minimal Corresponding Source".

LGPL also requires that you provide a mechanism to combine the Application with a modified version of the library, and outlines some options for this. Also, the license of the whole work must permit "reverse engineering for debugging such modifications" to the library. Therefore, you should take care that the EULA used for the Application does not contradict this permission.

Thus, under the terms of LGPL, you must refrain from license terms on works based on the licensed work that prohibit replacement of the licensed components of the larger non-LGPL'd work, or prohibit decompilation or reverse engineering in order to enhance or fix bugs in the LGPL'd components.

LGPLv3 is not surprisingly easier to understand and examine from a compliance lens, since the FSF was influenced in LGPLv3's drafting by questions and comments on LGPLv2.1 over a period of years. Admittedly, LGPLv2.1 is still in wide use, and thus compliance with LGPLv2.1 remains a frequent topic you may encounter. The best advice there is careful study of Chapter 10.

However, to repeat a key point here made within that chapter: Note though that, since the LGPLv2.1 can be easily upgraded to GPLv2-or-later, in the worst case you simply need to comply as if the software was licensed under GPLv2. The only reason you must consider the question of whether you have a "work that uses the library" or a "work based on the library" is when you wish to take advantage of the "weak copyleft" effect of the Lesser GPL. If GPLv2-or-later is an acceptable license (i.e., if you plan to copyleft the entire work anyway), you may find this an easier option.

18.2 Upstream Providers

With ever-increasing frequency, software development (particularly for embedded devices) is outsourced to third parties. If you rely on an upstream provider for your software, note that you *cannot ignore your GPL*

compliance requirements simply because someone else packaged the software that you distribute. If you redistribute GPL'd software (which you do, whenever you ship a device with your upstream's software in it), you are bound by the terms of the GPL. No distribution (including redistribution) is permissible absent adherence to the license terms.

Therefore, you should introduce a due diligence process into your software acquisition plans. This is much like the software-oriented recommendations we make in § 14. Implementing practices to ensure that you are aware of what software is in your devices can only improve your general business processes. You should ask a clear list of questions of all your upstream providers and make sure the answers are complete and accurate. The following are examples of questions you should ask:

- What are all the licenses that cover the software in this device?
- From which upstream vendors, be they companies or individuals, did *you* receive your software before distributing it to us?
- What are your GPL compliance procedures?
- If there is GPL'd software in your distribution, we will be redistributors of this GPL'd software. What mechanisms do you have in place to aid us with compliance?
- If we follow your recommended compliance procedures, will you formally indemnify us in case we are nonetheless found to be in violation of the GPL?

This last point is particularly important. Many GPL enforcement actions are escalated because of petty finger-pointing between the distributor and its upstream. In our experience, agreements regarding GPL compliance issues and procedures are rarely negotiated up front. However, when they are, violations are resolved much more smoothly (at least from the point of view of the redistributor).

Consider the cost of potential violations in your acquisition process. Using Free Software allows software vendors to reduce costs significantly, but be wary of vendors who have done so without regard for the licenses. If your vendor's costs seem "too good to be true," you may ultimately bear the burden of the vendor's inattention to GPL compliance. Ask the right questions, demand an account of your vendors' compliance procedures, and seek indemnity from them.

In particular, any time your vendor incorporates copylefted software, you *must* exercise your own rights as a user to request CCS for all the copylefted programs that your suppliers provided to you. Furthermore, you must ensure that CCS is correct and adequate yourself. Good vendors should help you do this, and make it easy. If those vendors cannot, pick a different vendor before proceeding with the product.

18.3 Mergers and Acquisitions

Often, larger companies often encounter copyleft licensing during a Mergers and Acquisitions (M&A) process. Ultimately, a merger or acquisition causes all of the other company's problems to become yours. Therefore, for most concerns, the acquirer "simply" must apply the compliance analysis and methodologies discussed earlier across the acquired company's entire product line. Of course, this is not so simple, as such effort may be substantial, but a well-defined process for compliance investigation means the required work, while voluminous, is likely rote.

A few sections of GPL require careful attention and legal analysis to determine the risk of acquisitions. Those handling M&A issues should pay particular attention to the requirements of GPLv2 §7 and GPLv3 §10–12 — focusing on how they relate to the acquired assets may be of particular importance.

For example, GPLv3§10 clarifies that in business acquisitions, whether by sale of assets or transfers of control, the acquiring party is downstream from the party acquired. This results in new automatic downstream licenses from upstream copyright holders, licenses to all modifications made by the acquired business, and rights to source code provisioning for the now-downstream purchaser. However, despite this aid given by explicit language in GPLv3, acquirers must still confirm compliance by the acquired (even if GPLv3§10 does assert the the acquirers rights under GPL, that does not help if the acquired is out of compliance altogether). Furthermore, for fear of later reprisal by the acquirer if a GPL violation is later discovered in the acquired's product line, the acquired may need to seek a waiver and release of from

additional damages beyond a requirement to comply fully (and a promise of rights restoration) if a GPL violation by the acquired is later uncovered during completion of the acquisition or thereafter.

Finally, other advice available regarding handling of GPL compliance in an M&A situation tends to ignore the most important issue: most essential copylefted software is not wholly copyrighted by the entities involved in the M&A transaction. Therefore, copyleft obligations likely reach out to the customers of all entities involved, as well as to the original copyright holders of the copylefted work. As such, notwithstanding the two paragraphs in GPLv3§10, the entities involved in M&A should read the copyleft licenses through the lens of third parties whose software freedom rights under those licenses are of equal importance to then entities inside the transaction.

18.4 User Products and Installation Information

GPLv3 requires you to provide "Installation Information" when v3 software is distributed in a "User Product." During the drafting of v3, the debate over this requirement was contentious. However, the provision as it appears in the final license is reasonable and easy to understand.

If you put GPLv3'd software into a User Product (as defined by the license) and *you* have the ability to install modified versions onto that device, you must provide information that makes it possible for the user to install functioning, modified versions of the software. Note that if no one, including you, can install a modified version, this provision does not apply. For example, if the software is burned onto an non-field-upgradable ROM chip, and the only way that chip can be upgraded is by producing a new one via a hardware factory process, then it is acceptable that the users cannot electronically upgrade the software themselves.

Furthermore, you are permitted to refuse support service, warranties, and software updates to a user who has installed a modified version. You may even forbid network access to devices that behave out of specification due to such modifications. Indeed, this permission fits clearly with usual industry practice. While it is impossible to provide a device that is completely unmodifiable[1], users are generally on notice that they risk voiding their warranties and losing their update and support services when they make modifications.[2]

GPLv3 is in many ways better for distributors who seek some degree of device lock-down. Technical processes are always found for subverting any lock-down; pursuing it is a losing battle regardless. With GPLv3, unlike with GPLv2, the license gives you clear provisions that you can rely on when you are forced to cut off support, service or warranty for a customer who has chosen to modify.

18.5 Beware The Consultant in Enforcers' Clothing

There are admittedly portions of the GPL enforcement community that function somewhat like the computer security and network penetration testing hacker community. By analogy, most COGEO's consider themselves white hats, while some might appropriately call proprietary relicensing by the name "black hats". And, to finalize the analogy, there are indeed few grey hat GPL enforcers.

Grey hat GPL enforcers usually have done some community-oriented GPL enforcement themselves, typically working as a volunteer for a COGEO, but make their living as a "hired gun" consultant to find GPL violations and offer to "fix them" for companies. Other such operators hold copyrights in some key piece of copylefted software and enforce as a mechanism to find out who is most likely to fund improvements on the software.

A few companies report that they have formed beneficial consulting or employment relationships with developers they first encountered through enforcement. In some such cases, companies have worked with such consultants to alter the mode of use of the project's code in the company's products. More often in these cases, the communication channels opened in the course of the inquiry served other consulting purposes later.

[1] Consider that the iPhone, a device designed primarily to restrict users' freedom to modify it, was unlocked and modified within 48 hours of its release.

[2] A popular t-shirt in the software freedom community reads: "I void warranties.". Our community is well-known for modifying products with full knowledge of the consequences. GPLv3's "Installation Instructions" section merely confirms that reality, and makes sure GPL rights can be fully exercised, even if users exercise those rights at their own peril.

Feelings and opinions about this behavior are mixed within the larger copyleft community. Some see it as a reasonable business model and others renounce it as corrupt behavior. Regardless, a GPL violator should always immediately determine the motivations of the enforcer via documented, verifiable facts. For example, COGEOs such as the FSF and Conservancy have made substantial public commitments to enforce in a way that is uniform, transparent, and publicly documented. Furthermore, since these specific organizations are public charities in the USA, they are accountable to the IRS (and the public at large) in their annual Form 990 filings. Everyone may examine their revenue models and scrutinize their work.

However, entities and individuals who do GPL enforcement centered primarily around a profit motive are likely the most dangerous enforcement entities for one simple reason: an agreement to comply fully with the GPL for past and future products — always the paramount goal to COGEOs — may not suffice as adequate resolution for a proprietary relicensing company or grey hat GPL enforcer. Therefore, violators must consider carefully who has made the enforcement inquiry and ask when and where the enforcer made public commitments and reports regarding their enforcement work and perhaps even ask the enforcer to directly mimic CEOGEO's detailed public disclosures and follow the standard requests for resolution found in this document.

CHAPTER 19

CONCLUSION

GPL compliance need not be an onerous process. Historically, struggles have been the result of poor development methodologies and communications, rather than any unexpected application of the GPL's source code disclosure requirements.

Compliance is straightforward when the entirety of your enterprise is well-informed and well-coordinated. The receptionists should know how to route a GPL source request or accusation of infringement. The lawyers should know the basic provisions of Free Software licenses and your source disclosure requirements, and should explain those details to the software developers. The software developers should use a version control system that allows them to associate versions of source with distributed binaries, have a well-documented build process that anyone skilled in the art can understand, and inform the lawyers when they bring in new software. Managers should build systems and procedures that keep everyone on target. With these practices in place, any organization can comply with the GPL without serious effort, and receive the substantial benefits of good citizenship in the software freedom community, and lots of great code ready-made for their products.

Part III

Case Studies in GPL Enforcement

PREFACE

This one-day course presents the details of five different GPL compliance cases handled by FSF's GPL Compliance Laboratory. Each case offers unique insights into problems that can arise when the terms of the GPL are not properly followed, and how diplomatic negotiation between the violator and the copyright holder can yield positive results for both parties.

Attendees should have successfully completely the course, a "Detailed Study and Analysis of the GPL and LGPL," as the material from that course forms the building blocks for this material.

This course is of most interest to lawyers who have clients or employers that deal with Free Software on a regular basis. However, technical managers and executives whose businesses use or distribute Free Software will also find the course very helpful.

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.

CHAPTER 20

OVERVIEW OF COMMUNITY ENFORCEMENT

The GPL is a Free Software license with legal teeth. Unlike licenses like the X11-style or various BSD licenses, the GPL (and by extension, the LGPL) is designed to defend as well as grant freedom. We saw in the last course that the GPL uses copyright law as a mechanism to grant all the key freedoms essential in Free Software, but also to ensure that those freedoms propagate throughout the distribution chain of the software.

20.1 Termination Begins Enforcement

As we have learned, the assurance that Free Software under the GPL remains Free Software is accomplished through various terms of the GPL: §3 ensures that binaries are always accompanied with source; §2 ensures that the sources are adequate, complete and usable; §6 and §7 ensure that the license of the software is always the GPL for everyone, and that no other legal agreements or licenses trump the GPL. It is §4, however, that ensures that the GPL can be enforced.

Thus, §4 is where we begin our discussion of GPL enforcement. This clause is where the legal teeth of the license are rooted. As a copyright license, the GPL governs only the activities governed by copyright law — copying, modifying and redistributing computer software. Unlike most copyright licenses, the GPL gives wide grants of permission for engaging with these activities. Such permissions continue, and all parties may exercise them until such time as one party violates the terms of the GPL. At the moment of such a violation (i.e., the engaging of copying, modifying or redistributing in ways not permitted by the GPL) §4 is invoked. While other parties may continue to operate under the GPL, the violating party loses their rights.

Specifically, §4 terminates the violators' rights to continue engaging in the permissions that are otherwise granted by the GPL. Effectively, their rights revert to the copyright defaults — no permission is granted to copy, modify, nor redistribute the work. Meanwhile, §5 points out that if the violator has no rights under the GPL, they are prohibited by copyright law from engaging in the activities of copying, modifying and distributing. They have lost these rights because they have violated the GPL, and no other license gives them permission to engage in these activities governed by copyright law.

20.2 Ongoing Violations

In conjunction with §4's termination of violators' rights, there is one final industry fact added to the mix: rarely does one engage in a single, solitary act of copying, distributing or modifying software. Almost always, a violator will have legitimately acquired a copy of a GPL'd program, either making modifications or not,

and then begun distributing that work. For example, the violator may have put the software in boxes and sold them at stores. Or perhaps the software was put up for download on the Internet. Regardless of the delivery mechanism, violators almost always are engaged in *ongoing* violation of the GPL.

In fact, when we discover a GPL violation that occurred only once — for example, a user group who distributed copies of a GNU/Linux system without source at one meeting — we rarely pursue it with a high degree of tenacity. In our minds, such a violation is an educational problem, and unless the user group becomes a repeat offender (as it turns out, they never do), we simply forward along a FAQ entry that best explains how user groups can most easily comply with the GPL, and send them on their merry way.

It is only the cases of *ongoing* GPL violation that warrant our active attention. We vehemently pursue those cases where dozens, hundreds or thousands of customers are receiving software that is out of compliance, and where the company continually offers for sale (or distributes gratis as a demo) software distributions that include GPL'd components out of compliance. Our goal is to maximize the impact of enforcement and educate industries who are making such a mistake on a large scale.

In addition, such ongoing violation shows that a particular company is committed to a GPL'd product line. We are thrilled to learn that someone is benefiting from Free Software, and we understand that sometimes they become confused about the rules of the road. Rather than merely giving us a postmortem to perform on a past mistake, an ongoing violation gives us an active opportunity to educate a new contributor to the GPL'd commons about proper procedures to contribute to the community.

Our central goal is not, in fact, to merely clear up a particular violation. In fact, over time, we hope that our compliance lab will be out of business. We seek to educate the businesses that engage in commerce related to GPL'd software to obey the rules of the road and allow them to operate freely under them. Just as a traffic officer would not revel in reminding people which side of the road to drive on, so we do not revel in violations. By contrast, we revel in the successes of educating an ongoing violator about the GPL so that GPL compliance becomes a second-nature matter, allowing that company to join the GPL ecosystem as a contributor.

20.3 How are Violations Discovered?

Our enforcement of the GPL is not a fund-raising effort; in fact, FSF's GPL Compliance Lab runs at a loss (in other words, it is subsided by our donors). Our violation reports come from volunteers, who have encountered, in their business or personal life, a device or software product that appears to contain GPL'd software. These reports are almost always sent via email to <license-violation@fsf.org>.

Our first order of business, upon receiving such a report, is to seek independent confirmation. When possible, we get a copy of the software product. For example, if it is an offering that is downloadable from a Web site, we download it and investigate ourselves. When it is not possible for us to actually get a copy of the software, we ask the reporter to go through the same process we would use in examining the software.

By rough estimation, about 95% of violations at this stage can be confirmed by simple commands. Almost all violators have merely made an error and have no nefarious intentions. They have made no attempt to remove our copyright notices from the software. Thus, given the third-party binary, `tpb`, usually, a simple command (on a GNU/Linux system) such as the following will find a Free Software copyright notice and GPL reference:

```
strings tpb | grep Copyright
```

In other words, it is usually more than trivial to confirm that GPL'd software is included.

Once we have confirmed that a violation has indeed occurred, we must then determine whose copyright has been violated. Contrary to popular belief, FSF does not have the power to enforce the GPL in all cases. Since the GPL operates under copyright law, the powers of enforcement — to seek redress once §4 has been invoked — lie with the copyright holder of the software. FSF is one of the largest copyright holders in the world of GPL'd software, but we are by no means the only one. Thus, we sometimes discover that while GPL'd code is present in the software, there is no software copyrighted by FSF present.

In cases where FSF does not hold copyright interest in the software, but we have confirmed a violation, we contact the copyright holders of the software, and encourage them to enforce the GPL. We offer our good offices to help negotiate compliance on their behalf, and many times, we help as a third party to settle

such GPL violations. However, what we will describe primarily in this course is FSF's first-hand experience enforcing its own copyrights and the GPL.

20.4 First Contact

The Free Software community is built on a structure of voluntary cooperation and mutual help. Our community has learned that cooperation works best when you assume the best of others, and only change policy, procedures and attitudes when some specific event or occurrence indicates that a change is necessary. We treat the process of GPL enforcement in the same way. Our goal is to encourage violators to join the cooperative community of software sharing, so we want to open our hand in friendship.

Therefore, once we have confirmed a violation, our first assumption is that the violation is an oversight or otherwise a mistake due to confusion about the terms of the license. We reach out to the violator and ask them to work with us in a collaborative way to bring the product into compliance. We have received the gamut of possible reactions to such requests, and in this course, we examine four specific examples of such compliance work.

CHAPTER 21

THINKPENGUIN WIRELESS ROUTER: EXCELLENT CCS

Too often, case studies examine failure and mistakes. Indeed, most of the chapters that follow herein will consider the myriad difficulties discovered in community-oriented GPL enforcement for the last two decades. However, to begin, this is a case study in how copyleft compliance can indeed be done correctly.

This example is, in fact, more than ten years in the making. Since almost the inception of for-profit corporate adoption of Free Software, companies have requested a clear example of a model citizen to emulate. Sadly, while community-oriented enforcers have vetted uncounted thousands of "Complete, Corresponding Source" (CCS) candidates from hundreds of companies, this particular CCS release described herein is the first ever declared a "pristine example".

Of course, most CCS examined for the last decade has (eventually) complied with the GPL, perhaps after many iterations of review by the enforcer. However, in the experience of the two primary community-oriented enforcers (Conservancy and the FSF), such CCS results routinely "barely comply with GPL's requirements". To use an academic analogy: while a "C" is certainly a passing grade, any instructor prefers to disseminate to the class an exemplar sample that earned an "A".

Fortunately, thanks in large part to the FSF's "Respects Your Freedom" (RYF) certification campaign[1], a few electronics products on the market meet a higher standard of copyleft compliance. As such, for the first time in the history of copyleft, CCS experts have pristine examples to study and present as exemplars worthy of emulation.

This case study therefore examines the entire life-cycle of a GPL compliance investigation: from product purchase, to source request, to CCS review, and concluding in a final compliance determination. Specifically, this chapter discusses the purchase, CCS provision, and a step-by-step build and installation analysis of a specific, physical, embedded electronics product: the "TPE-NWIFIROUTER" wireless router by ThinkPenguin.[2]

21.1 Consumer Purchase and Unboxing

The process for copyleft compliance investigation, when properly conducted, determines whether users inclined to exercise their rights under a copyleft license will be successful in their attempt. Therefore, at every

[1] RYF is a campaign by FSF to certify products that truly meet the principles of software freedom. Products must meet strict standards for RYF certification, and among them is a pristine example of CCS.

[2] The FSF of course performed a thorough CCS check as part of its certification process. The analysis discussed herein was independently performed by Software Freedom Conservancy without reviewing the FSF's findings. Thus, this analysis is "true to form", and explains the typical procedures Conservancy uses when investigating a potential GPL violation. In this case, obviously, no violation was uncovered.

stage, the investigator seeks to take actions that reasonably technically knowledgeable users would during the ordinary course of their acquisition and use of copyleft-covered products. As such, the investigator typically purchases the device on the open market to verify that distribution of the copylefted software therein complies with binary distribution requirements (such as those discussed earlier in § 5.2 and § 9.9).

Therefore, the investigator first purchased the TPE-NWIFIROUTER through an online order, and when the package arrived, examined the contents of the box. The investigator immediately discovered that ThinkPenguin had taken advice from § 15.1.2, and exercised GPLv2§3(a) and GPLv3§6, rather than using the problematic offer for source provisions. This choice not only accelerated the investigation (since there was no CCS offer to "test"), but also simplified the compliance requirements for ThinkPenguin.

21.2 Root Filesystem and Kernel Compilation

The CD found in the box was labeled "libreCMC v1.2.1 source code", and contained 407 megabytes of data. The investigator copied this ISO to a desktop GNU/Linux system and examined its contents. Upon doing so, the investigator immediately found a file called "README" at the top-level directory:

```
$ dd if=/dev/cdrom of=libreCMC_v1.2.1_SRC.iso
$ mkdir libCMC
$ sudo mount -o loop ./libreCMC_v1.2.1_SRC.iso libCMC
mount: block device /path/to/libreCMC_v1.2.1_SRC.iso
       is write-protected, mounting read-only
$ ls -1 libCMC
bin
librecmc-u-boot.tar.bz2
librecmc-v1.2.1.tar.bz2
README
u-boot_reflash
$ cat libCMC/README
...
```

The investigator therefore knew immediately to begin the CCS check should begin with a study of the contents of "README". Indeed, that file contained the appropriate details to start the build:

> In order to build firmware images for your router, the following needs to be installed:
> gcc, binutils, bzip2, flex, python, perl, make, find, grep, diff, unzip, gawk, getopt, libz-dev and libc headers.
>
> Please use "make menuconfig" to configure your appreciated configuration for the toolchain and firmware. Please note that the default configuration is what was used to build the firmware image for your router. It is advised that you use this configuration.
>
> Simply running "make" will build your firmware. The build system will download all sources, build the cross-compile toolchain, the kernel and all chosen applications.
>
> To build your own firmware you need to have access to a GNU/Linux system (case-sensitive filesystem required).

In other words, the first "script" that investigator "ran" was the above. This was not a software script, rather the processor for the script was the investigator's own brain — like a script of a play. Less glibly stated: instructions written in English are usually necessary for the build and installation operations that demand actual intelligence. In this case, the investigator ascertained the host system requirements for construction of this embedded firmware.

GPL does not give specific guidance on the form or location of "scripts used to control compilation and installation of the executable" and/or "Installation Information". Community-oriented GPL enforcers apply a "reasonableness standard" to evaluate such instructions. If an investigator of average skill in embedded firmware construction can surmise the proper procedures to build and install a replacement firmware, the instructions are likely sufficient to meet GPL's requirements. Fortunately, in this case, the instructions are more abundant and give extra detail.

Nevertheless, these instructions offer more options than the reader typically sees in other CCS candidates. More typically, top-level build instructions name an exact host distribution to use, such as "Debian 7 installed on an amd64 system with the following packages installed". Of course, if the build will fail on any other system, instructions *should* include such details. However, this CCS builds on a wide range of distributions, and thus it was appropriate (and preferred) that the build instructions do not specify a specific distribution.

In this specific case, the developers of the libreCMC project (a Free Software project that forms the base system for the TPE-NWIFIROUTER) have clearly made an effort to ensure the CCS builds on a variety of host systems. The investigator was in fact dubious upon seeing these instructions, since finicky embedded build processes usually require a very specific host system. Fortunately, it seems such doubts were generally unfounded (although the investigator did find a minor annoyance that could be resolved with more detailed instructions).

Anyway, since these instructions did not specify a specific host system, the investigator simply used his own amd64 Debian GNU/Linux 6 desktop system. Before beginning, the investigator used the following command:

```
$ dpkg --list | egrep '^iii' | less
```

to verify that the required packages listed in the README were installed[3].

Next, the investigator then extracted the primary source package with the following command:

```
$ tar --posix -jxpf libCMC/librecmc-v1.2.1.tar.bz2
```

The investigator did notice an additional source release, entitled "librecmc-u-boot.tar.bz2". The investigator concluded upon simple inspection that the instructions found in "u-boot_reflash" were specific instructions for that part of the CCS. This was a minor annoyance, and ideally the "README" would so-state, but the CCS filesystem layout met the reasonableness standard; the skilled investigator determine the correct course of action with a few moments of study.

The investigator then noted the additional step offered by the "README", which read:

> Please use "make menuconfig" to configure your appreciated configuration for the toolchain and firmware. Please note that the default configuration is what was used to build the firmware image for your router. It is advised that you use this configuration.

This instruction actually exceeds GPL's requirements. Specifically, the instruction guides users in their first step toward exercising the freedom to modify the software. While the GPL does contain requirements that facilitate the freedom to modify (such as ensuring the CCS is in the "preferred form ... for making modifications to it"), GPL does not require specific instructions explaining how to undertake modifications. This specific instruction therefore exemplifies the exceptional quality of this particular CCS.

However, for purposes of the CCS verification process, typically the investigator avoids any unnecessary changes to the source code during the build process, lest the investigator err and cause the build to fail through his own modification, and thus incorrectly identify the CCS as inadequate. Therefore, the investigator proceeded to simply run:

```
$ cd libCMC
$ make
```

and waited approximately 40 minutes for the build to complete[4]. The investigator kept a full log of the build, which is not included herein due its size (approximately 7.2K of text).

Upon completion of the "make" process, the investigator immediately found (almost to his surprise) several large firmware files in the "bin/ar71xx" directory. Typically, this step in the CCS verification process is harrowing. In most cases, the "make" step will fail due to a missing package or because toolchain paths are not setup correctly.

In light of such experiences, the investigator speculated that ThinkPenguin's engineers did the most important step in self-CCS verification: test one's own instructions on a clean system. Ideally, an employee with similar skills but unfamiliar with the specific product can most easily verify CCS and identify problems before a violation occurs.

[3]The "dpkg" command is a Debian-specific way of finding installed packages.

[4]Build times will likely vary widely on various host systems.

However, upon completing the "make", the investigator was unclear which filesystem and kernel images to install on the TPE-NWIFIROUTER hardware. Ideally, the original "README" would indicate which image is appropriate for the included hardware. However, this was ultimately an annoyance rather than a compliance issue. Fortunately, the web UI (see next section) on the TPE-NWIFIROUTER performs firmware image installation. Additionally, the router's version number was specified on the bottom of the device, which indicated which of the differently-versioned images we should install. The investigator would prefer instructions such as those found at `http://librecmc.org/librecmc/wiki?name=Tp+MR3020`instructions similar to these in the README itself; however, application of the reasonableness standard here again indicates compliance, since a knowledgeable user can easily determine the proper course of action.

21.3 U-Boot Compilation

The investigator then turned his attention to the file, "u-boot_reflash". These instructions explained how to build and install the bootloader for the device.

The investigator followed the instructions for compiling U-Boot, and found them quite straight-forward. The investigator discovered two minor annoyances, however, while building U-Boot:

- The variable `$U-BOOT_SRC` was used as a placeholder for the name of the extracted source directory. This was easy to surmise and was not a compliance issue (per the reasonableness standard), but explicitly stating that fact at the top of the instructions would be helpful.
- Toolchain binaries were included and used by default by the build process. These binaries were not the appropriate ones for the investigator's host system, and the build failed with the following error:

  ```
  mips-librecmc-linux-uclibc-gcc.bin: /lib/libc.so.6:
      version `GLIBC_2.14' not found
      (required by mips-librecmc-linux-uclibc-gcc.bin)
  ```

  (The complete log output from the failure is too lengthy to include herein.)

  This issue is an annoyance, not a compliance problem. It was clear from context that these binaries were simply for a different host architecture, and the investigator simply removed "toolchain/bin" and created a symlink to utilize the toolchain already built earlier (during the compilation discussed in § 21.2):

  ```
  $ ln -s \
    ../../staging_dir/toolchain-mips_34kc_gcc-4.6-linaro_uClibc-0.9.33.2/bin \
    toolchain/bin
  ```

  After this change, the U-Boot build completed successfully.

The full log of the build is not included herein due its size (approximately 3.8K of text). After that, the investigator found a new U-Boot image in the "bin" directory.

21.4 Root Filesystem and Kernel Installation

The investigator next tested installation of the firmware. In particular, the investigator connected the TPE-NWIFIROUTER to a local network, and visited `http://192.168.10.1/`, logged in, and chose the option sequence: "System ⇒ Backup / Flash Firmware".

From there, the investigator chose the "Flash new firmware image" section and selected the "librecmc-ar71xx-generic-tl-wr841n-v8-squashfs-sysupgrade.bin" image from the "bin/ar71xx" directory. The investigator chose the "v8" image upon verifying the physical router read "v8.2" on its bottom. The investigator chose the "sysupgrade" version of the image because this was clearly a system upgrade (as a firmware already came preinstalled on the TPE-NWIFIROUTER).

Upon clicking "Flash image...", the web interface prompted the investigator to confirm the MD5 hash of the image to flash. The investigator did so, and then clicked "Proceed" to flash the image. The process

took about one minute, at which point the web page refreshed to the login screen. Upon logging in, the investigator was able to confirm in the "Kernel Log" section of the web interface that the newly built copy of Linux had indeed been installed.

The investigator confirmed that a new version of "busybox" had also been installed by using SSH to connect to the router and ran the command "busybox", which showed the newly-compiled version (via its date of compilation).

21.5 U-Boot Installation

The U-Boot installation process is substantially more complicated than the firmware update. The investigator purchased the optional serial cable along with the TPE-NWIFIROUTER, in order to complete the U-Boot installation per the instructions in "u-boot_reflash" in its section "Installing u-boot to your router", which reads:

> 1. Install and configure any TFTP server on your PC (tftp-hpa). Set a fixed IP address on your PC ... and connect it to the router, using RJ45 network cable ...
> 2. Connect USB to UART adapter to the router and start any application to communicate with it, like PuTTY. ...
> 3. Power on the router, wait for a line like one of the following and interrupt the process of loading a kernel:
>
> ```
> Autobooting in 1 seconds
>      (for most TP-Link routers, you should enter tpl at this point)
> Hit ESC key to stop autoboot: 1 (for 8devices Carambola 2, use ESC key)
>  Hit any key to stop autoboot: 1 (for D-Link DIR-505, use any key)
> ```
>
> 4. Set ipaddr and serverip environment variables:
>
> ```
> hornet> setenv ipaddr 192.168.1.1
> hornet> setenv serverip 192.168.1.2
> ```

The investigator used the purchased serial cable, which was a USB serial adapter with a male USB type A connector to 4 female jumper wires. The female jumper wires were red, black, white, and green.

The instructions here were slightly incomplete, since they did not specify how to connect the wires to the router. However, the investigator found general information available online at `http://wiki.openwrt.org/toh/tp-link/tl-wr841nd#serial.console` which described the proper procedure. While the "power" and "ground" cables were obvious, some trial and error was necessary to find the RX and TX cables, but this was easily determined since miswiring TX and RX yields no I/O and proper wiring yields the output as expected. Using the pin gender changer included with the adapter, the investigator was able to stably wire the pins for use once the proper RX and TX connections were determined.

The investigator then used the recommended 115200 8N1 for serial console settings, leaving all flow control off, and tested both with the `minicom` and `screen` commands. The investigator found that if all 4 wires were connected on the USB serial adapter that the router would start without additional power and the console would receive the startup messages. The investigator could replicate the same behavior by omitting the power cable from the USB serial adapter (red wire) and connecting the main power adapter to the router instead.

At this point, the on-screen messages as described in the installation instructions appeared, but the investigator found that no key events sent via the serial port appeared to reach the U-Boot console. In other words, while the investigator saw both U-Boot and kernel boot messages in the serial console, the investigator was unable to interrupt the boot process as instructed by "u-boot_reflash". Hitting a key simply did *not* interrupt the boot process and yield the `hornet>` prompt.

After additional trial and error over a period of hours, the investigator had finally to consider this question for the first time during the process: "Has ThinkPenguin violated the GPL?" More specifically, the immediate question was: "Given this failure, has the distributor met the requirements for 'scripts used to control ... installation of the executable' (GPLv2) and necessary 'Installation Information' (GPLv3)?"

The appropriate answer to the question (at this specific stage) is "possibly, but more information is needed". Embedded installation and configuration is a tricky and complex technical process. While the GPL requires documentation and clear instructions for this process, the investigator did not immediately blame the distributor for what may be an honest, correctable mistake, or an issue legitimately explained by feasible alternative theories.

In this case, upon remembering the issues of wiring, the investigator wonder if perhaps the power pin was accidentally connected for a moment to the RX pin while live. Such action could easily fry the RX pin, and yield the observed behavior. Since the investigator could not rule out the possibility of accidental connection of power to the RX pin mentioned, the investigator had to assume the instructions would work properly if he had not made that error.

That conclusion, while correct, also left the investigator with only two option to complete the final verification of the CCS:

- Purchase a new router and cable anew, and reattempt the installation process while taking extra care not to miswire any cables.
- Seek assistance from the libreCMC community to find an alternative method of installation.

The investigator chose the latter and then contacted a libreCMC developer familiar with the product. That developer, who agreed the the RX pin was likely ruined, described an alternative method for console access using the `netcat`. The libreCMC developer described the process as follows:

- Change the IP address of the router to 192.168.1.1.
- Change the IP address of a desktop GNU/Linux system to 192.168.1.2.
- Power on the router while holding the reset button for 7 seconds.
- Use the `netcat` command (as below) on the desktop, and press enter to receive U-Boot's prompt:

```
$ nc -u -p 6666 192.168.1.1 6666
uboot>
```

Upon following this procedure, the investigator was able to confirm the (original) shipped version of U-Boot was still installed:

```
$ nc -u -p 6666 192.168.1.1 6666
uboot> version
U-Boot 1.1.4 (Jul 28 2014)
```

Thereafter, the investigator followed the instructions from "u-boot_reflash". Specifically, the investigator configured a TFTP server and placed the newly built firmware into `/srv/tftp`. The investigator also followed the remaining instructions in "u-boot_reflash", but used the `netcat` console rather than the serial console, and used U-Boot's `reset` command to reboot the router.

Upon reboot, the serial console (still connect with working output) showed the message `U-Boot 1.1.4 (Oct 17 2014)`, and thus confirmed a successful reflash of the U-Boot image built by the investigator.

21.6 Firmware Comparison

Next, to ensure the CCS did indeed correspond to the firmware original installed on the TPE-NWIFIROUTER, the investigator compared the built firmware image with the filesystem originally found on the device itself. The comparison steps were as follows:

1. Extract the filesystem from the image we built by running find-firmware.pl on "bin/ar71xx/librecmc-ar71xx-generic-tl-wr841n-v8-squashfs-factory.bin" and then running bat-extratools' "squashfs4.2/squashfs-tools/bat-unsquashfs42" on the resulting morx0.squash, using the filesystem in the new squashfs-root directory for comparison.

2. Login to the router's web interface (at `http://192.168.10.1/`) from a computer connected to the router.

3. Set a password using the provided link at the top (since the router's UI warns that no password is set and asks the user to change it).

4. Logged into the router via SSH, using the root user with the aforementioned password.

5. Compared representative directory listings and binaries to ensure the set of included files (on the router) is similar to those found in the firmware image that the investigator created (whose contents are now in the local squashfs-root directory). In particular, the investigator did the following comparisons:

   (a) Listed the /bin folder ("ls -l /bin") and confirm the list of files is the same and that the file sizes are similar.

   (b) Checked the "strings" output of "/bin/busybox" to confirm it is similar in both places (similar number of lines and content of lines). (One cannot directly compare the binaries because the slight compilation variations will cause some bits to be different.)

   (c) Repeated the above two steps for "/lib/modules", "/usr/bin", and other directories with a significant number of binaries.

   (d) Checked that the kernel was sufficiently similar. The investigator compared the "dmesg" output both before and after flashing the new firmware. As the investigator expected, the kernel version string was similar, but had a different build date and user@host indicator. (The kernel binary itself is not easily accessible from an SSH login, but was retrievable using the U-Boot console (the start address of the kernel in flash appears to be 0x9F020000, based on the boot messages seen in the serial console).

21.7 Minor Annoyances

As discussed in detail above, there were a few minor annoyances, none of which were GPL violations. Rather, the annoyances briefly impeded the build and installation. However, the investigator, as a reasonably skilled build engineer for embedded devices, was able to complete the process with the instructions provided.

To summarize, no GPL compliance issues were found, and the CCS release was one of the best ever reviewed by any investigator at any community-oriented enforcement organization. However, the following annoyances were discovered:

- Failure to explain how to extract the source tarball and then where to run the "make" command.
- Failure to explain how to install the kernel and root filesystem on the device; the user must assume the web UI must be used.
- Including pre-built toolchain binaries that don't work on all systems, and failure to copy and/or symlink built toolchain binaries in the right location.
- Failure to include information in the U-Boot installation instructions for wiring the serial cable.
- Ideally, the U-Boot installation instructions would also include the `netcat` method of installation.
- Finally, the instructions should note that the new U-Boot firmware should be placed into `/srv/tftp` when using TFTP on most GNU/Linux desktops.

Thus, no CCS is absolutely perfect, but GPL violation investigators always give the distributors the benefit of any doubts and seek to work with the vendors and improve their CCS for the betterment of their users, customers, and the entire software freedom community.

21.8 Lessons Learned

Companies that seek to redistribute copylefted software can benefit greatly from ThinkPenguin's example. Here are just a few of the many lessons that can be learned here:

1. Even though copyleft licenses have them, **avoid the offer-for-source provisions**. Not only does including the CCS alongside binary distribution make violation investigation and compliance confirmation substantially easier, but also (and more importantly) doing so completes the distributor's CCS compliance obligations at the time of distribution (provided, of course, that the distributor is otherwise in compliance with the relevant copyleft license).

2. **Include top-level build instructions in a natural language (such as English) in a clear and conspicuous place.** Copyleft licenses require that someone reasonably skilled in the art can reproduce the build and installation. Typically, instructions written in English are necessary, and often easier than writing programmed scripts. The "script" included can certainly be more like the script of a play and less like a Bash script.

3. **Write build/install instructions to the appropriate level of specificity**. The upstream engineers in this case study clearly did additional work to ensure functionality on a wide variety of host build systems; this is quite rare. When in doubt, include the maximum level of detail build engineers can provide with the CCS instructions, but also double-check to investigate if a more generalized solution (such as other host systems) work just as well for the build.

4. **Seek to adhere to the spirit of copyleft, not just the letter of the license**. Encouragement of users to improve and make their devices better is one of ThinkPenguin's commercial differentiators. Copyleft advocates that other companies have undervalued the large and lucrative market of users who seek hackable devices. By going beyond the mere minimal requirements of GPL, companies can immediately reap the benefits in that target market.

5. Community-oriented enforcement organizations do not play "gotcha"[5] with distributors regarding GPL violations. The goal in the GPL enforcement process is to achieve compliance and correct mistakes and annoyances. Such organizations therefore take an "innocent until proven guilty → assume guilty due to honest error rather than malicious action " approach. The goal is compliance (in direct contrast with the discussion in § 12.2 about the proprietary relicensing business model).

[5]For lack of a better phrase.

CHAPTER 22

BORTEZ: MODIFIED GCC SDK

In our first case study, we will consider Bortez, a company that produces software and hardware toolkits to assist OEM vendors, makers of consumer electronic devices.

22.1 Facts

One of Bortez's key products is a Software Development Kit ("SDK") designed to assist developers building software for a specific class of consumer electronics devices.

FSF received a report that the SDK may be based on the GNU Compiler Collection (which is an FSF-copyrighted collection of tools for software development in C, C++ and other popular languages). FSF investigated the claim, but was unable to confirm the violation. The violation reporter was unresponsive to follow-up requests for more information.

Since FSF was unable to confirm the violation, we did not pursue it any further. Bogus reports do happen, and we do not want to burden companies with specious GPL violation complaints. FSF shelved the matter until more evidence was discovered.

FSF was later able to confirm the violation when two additional reports surfaced from other violation reporters, both of whom had used the SDK professionally and noticed clear similarities to FSF's GNU GCC. FSF's Compliance Engineer asked the reporters to run standard tests to confirm the violation, and it was confirmed that Bortez's SDK was indeed a modified version of GCC. Bortez had ported to Windows and added a number of features, including support for a specific consumer device chipset and additional features to aid in the linking process ("LP") for those specific devices. FSF explained the rights that the GPL afforded these customers and pointed out, for example, that Bortez only needed to provide source to those in possession of the binaries, and that the users may need to request that source (if §3(b) was exercised). The violators confirmed that such requests were not answered.

FSF brought the matter to the attention of Bortez, who immediately escalated the matter to their attorneys. After a long negotiation, Bortez acknowledged that their SDK was indeed a modified version of GCC. Bortez released most of the source, but some disagreement occurred over whether LP was also derivative of GCC. After repeated FSF inquiries, Bortez reaudited the source to discover that FSF's analysis was correct. Bortez determined that LP included a number of source files copied from the GCC code-base.

Once the full software release was made available, FSF asked the violation reporters if it addressed the problem. Reports came back that the source did not properly build. FSF asked Bortez to provide better build instructions with the software, and such build instructions were incorporated into the next software release.

At FSF's request as well, Bortez informed customers who had previously purchased the product that the source was now available by announcing the availability on its Web site and via a customer newsletter.

Bortez did have some concerns regarding patents. They wished to include a statement with the software release that made sure they were not granting any patent permission other than what was absolutely required by the GPL. They understood that their patent assertions could not trump any rights granted by the GPL. The following language was negotiated into the release:

> Subject to the qualifications stated below, Bortez, on behalf of itself and its Subsidiaries, agrees not to assert the Claims against you for your making, use, offer for sale, sale, or importation of the Bortez's GNU Utilities or derivative works of the Bortez's GNU Utilities ("Derivatives"), but only to the extent that any such Derivatives are licensed by you under the terms of the GNU General Public License. The Claims are the claims of patents that Bortez or its Subsidiaries have standing to enforce that are directly infringed by the making, use, or sale of an Bortez Distributed GNU Utilities in the form it was distributed by Bortez and that do not include any limitation that reads on hardware; the Claims do not include any additional patent claims held by Bortez that cover any modifications of, derivative works based on or combinations with the Bortez's GNU Utilities, even if such a claim is disclosed in the same patent as a Claim. Subsidiaries are entities that are wholly owned by Bortez.
>
> This statement does not negate, limit or restrict any rights you already have under the GNU General Public License version 2.

This quelled Bortez's concerns about other patent licensing they sought to do outside of the GPL'd software, and satisfied FSF's concerns that Bortez give proper permissions to exercise teachings of patents that were exercised in their GPL'd software release.

Finally, a GPL Compliance Officer inside Bortez was appointed to take responsibility for all matters of GPL compliance inside the company. Bortez is responsible for informing FSF if the position is given to someone else inside the company, and making sure that FSF has direct contact with Bortez's Compliance Officer.

22.2 Lessons

This case introduces a number of concepts regarding GPL enforcement.

1. **Enforcement should not begin until the evidence is confirmed.** Most companies that distribute GPL'd software do so in compliance, and at times, violation reports are mistaken. Even with extensive efforts in GPL education, many users do not fully understand their rights and the obligations that companies have. By working through the investigation with reporters, the violation can be properly confirmed, and **the user of the software can be educated about what to expect with GPL'd software**. When users and customers of GPL'd products know their rights, what to expect, and how to properly exercise their rights (particularly under §3(b)), it reduces the chances for user frustration and inappropriate community outcry about an alleged GPL violation.

2. **GPL compliance requires friendly negotiation and cooperation.** Often, attorneys and managers are legitimately surprised to find out GPL'd software is included in their company's products. Engineers sometimes include GPL'd software without understanding the requirements. This does not excuse companies from their obligations under the license, but it does mean that care and patience are essential for reaching GPL compliance. We want companies to understand that participating and benefiting from a collaborative Free Software community is not a burden, so we strive to make the process of coming into compliance as smooth as possible.

3. **Confirming compliance is a community effort.** The whole point of making sure that software distributors respect the terms of the GPL is to allow a thriving software sharing community to benefit and improve the work. FSF is not the expert on how a compiler for consumer electronic devices should work. We therefore inform the community who originally brought the violation to our attention and ask them to assist in evaluation and confirmation of the product's compliance. Of course, FSF coordinates and oversees the process, but we do not want compliance for compliance's sake; rather, we wish to foster a cooperating community of development around the Free Software in question, and encourage the once-violator to begin participating in that community.

4. **Informing the harmed community is part of compliance.** FSF asks violators to make some attempt — such as via newsletters and the company's Web site — to inform those who already have the products as to their rights under the GPL. One of the key thrusts of the GPL's §1 and §3 is to *make sure the user knows she has these rights*. If a product was received out of compliance by a customer, she may never actually discover that she has such rights. Informing customers, in a way that is not burdensome but has a high probability of successfully reaching those who would seek to exercise their freedoms, is essential to properly remedy the mistake.

5. **Lines between various copyright, patent, and other legal mechanisms must be precisely defined and considered.** The most difficult negotiation point of the Bortez case was drafting language that simultaneously protected Bortez's patent rights outside of the GPL'd source, but was consistent with the implicit patent grant in the GPL. As we discussed in the first course of this series, there is indeed an implicit patent grant with the GPL, thanks to §6 and §7. However, many companies become nervous and wish to make the grant explicit to assure themselves that the grant is sufficiently narrow for their needs. We understand that there is no reasonable way to determine what patent claims read on a company's GPL holdings and which do not, so we do not object to general language that explicitly narrows the patent grant to only those patents that were, in fact, exercised by the GPL'd software as released by the company.

CHAPTER 23

BRACKEN: A MINOR VIOLATION IN A GNU/LINUX DISTRIBUTION

In this case study, we consider a minor violation made by a company whose knowledge of the Free Software community and its functions is deep.

23.1 The Facts

Bracken produces a GNU/Linux operating system product that is sold primarily to OEM vendors to be placed in appliance devices used for a single purpose, such as an Internet-browsing-only device. The product is almost 100% Free Software, mostly licensed under the GPL and related Free Software licenses.

FSF found out about this violation through a report first posted on a Slashdot[1] comment, and then it was brought to our attention again by another Free Software copyright holder who had discovered the same violation.

Bracken's GNU/Linux product is delivered directly from their Web site. This allowed FSF engineers to directly download and confirm the violation quickly. Two primary problems were discovered with the online distribution:

- No source code nor offer for source code was provided for a number of components for the distributed GNU/Linux system; only binaries were available
- An End User License Agreement ("EULA") was included that contradicted the permissions granted by the GPL

FSF contacted Bracken and gave them the details of the violation. Bracken immediately ceased distribution of the product temporarily and set forth a plan to bring themselves back into compliance. This plan included the following steps:

- Bracken attorneys would rewrite the EULA to comply with the GPL and would vet the new EULA through FSF before use
- Bracken engineers would provide source side-by-side with the binaries for the GNU/Linux distribution on the site (and on CD's, if ever they distributed that way)
- Bracken attorneys would run an internal seminar for its engineers regarding proper GPL compliance to help ensure that such oversights regarding source releases would not occur in the future

[1] Slashdot is a popular news and discussion site for technical readers.

- Bracken would resume distribution of the product only after FSF formally restored Bracken's distribution rights

This case was completed in about a month. FSF approved the new EULA text. The key portion in the EULA relating to the GPL read as follows:

> Many of the Software Programs included in Bracken Software are distributed under the terms of agreements with Third Parties ("Third Party Agreements") which may expand or limit the Licensee's rights to use certain Software Programs as set forth in [this EULA]. Certain Software Programs may be licensed (or sublicensed) to Licensee under the GNU General Public License and other similar license agreements listed in part in this section which, among other rights, permit the Licensee to copy, modify and redistribute certain Software Programs, or portions thereof, and have access to the source code of certain Software Programs, or portions thereof. In addition, certain Software Programs, or portions thereof, may be licensed (or sublicensed) to Licensee under terms stricter than those set forth in [this EULA]. The Licensee must review the electronic documentation that accompanies certain Software Programs, or portions thereof, for the applicable Third Party Agreements. To the extent any Third Party Agreements require that Bracken provide rights to use, copy or modify a Software Program that are broader than the rights granted to the Licensee in [this EULA], then such rights shall take precedence over the rights and restrictions granted in this Agreement solely for such Software Programs.

FSF restored Bracken's distribution rights shortly after the work was completed as described.

23.2 Lessons Learned

This case was probably the most quickly and easily resolved of all GPL violations in the history of FSF's Compliance Lab. The ease with which the problem was resolved shows a number of cultural factors that play a role in GPL compliance.

1. **Companies that understand Free Software culture better have an easier time with compliance.** Bracken's products were designed and built around the GNU/Linux system and Free Software components. Their engineers were deeply familiar with the Free Software ecosystem, and their lawyers had seen and reviewed the GPL before. The violation was completely an honest mistake. Since the culture inside the company had already adapted to the cooperative style of resolution in the Free Software world, there was very little work for either party to bring the product into compliance.

2. **When people in key positions understand the Free Software nature of their software products, compliance concerns are as mundane as minor software bugs.** Even the most functional system or structure has its problems, and successful business often depends on agile response to the problems that do come up; avoiding problems altogether is a pipe dream. Minor GPL violations can and do happen even with well-informed redistributors. However, resolution is reached quickly when the company — and in particular, the lawyers, managers, and engineers working on the Free Software product lines — have adapted to Free Software culture that the lower-level engineer already understood

3. **Legally, distribution must stop when a violation is identified.** In our opinion, Bracken went above and beyond the call of duty by ceasing distribution while the violation was being resolved. Under GPL §4, the redistributor loses the right to distribute the software, and thus they are in ongoing violation of copyright law if they distribute before rights are restored. It is FSF's policy to temporarily allow distribution while compliance negotiations are ongoing and only in the most extreme cases (where the other party appears to be negotiating in bad faith) does FSF even threaten an injunction on copyright grounds. However, Bracken — as a good Free Software citizen — chose to be on the safe side and do the legally correct thing while the violation case was pending. From start to finish, it took less than a month to resolve. This lapse in distribution did not, to FSF's knowledge, impact Bracken's business in any way.

4. **EULAs are a common area for GPL problems.** Often, EULAs are drafted from boilerplate text that a company uses for all its products. Even the most diligent attorneys forget or simply do not know that a product contains software licensed under the GPL and other Free Software licenses. Drafting a EULA that accounts for such licenses is straightforward; the text quoted above works just fine. The EULA must be designed so that it does not trump rights and permissions already granted by the GPL. The EULA must clearly state that if there is a conflict between it and the GPL, with regard to GPL'd code, the GPL is the overriding license.

5. **Compliance Officers are rarely necessary when companies are educated about GPL compliance.** As we saw in the Bortez case, FSF asks that a formal "GPL Compliance Officer" be appointed inside a previously violating organization to shepherd the organization to a cooperative approach to GPL compliance. However, when FSF sees that an organization already has such an approach, there is no need to request that such an officer be appointed.

CHAPTER 24

VIGORIEN: SECURITY, EXPORT CONTROLS, AND GPL COMPLIANCE

This case study introduces how concerns of "security through obscurity" and regulatory problems can impact GPL compliance matters.

24.1 The Facts

Vigorien distributes a back-up solution product that allows system administrators to create encrypted back-ups of file-systems on Unix-like computers. The product is based on GNU tar, a backup utility that replaces the standard Unix utility simply called tar, but has additional features.

Vigorien's backup solution added cryptographic features to GNU tar, and included a suite of utilities and graphical user interfaces surrounding GNU tar to make backups convenient.

FSF discovered the violation from a user report, and determined that the cryptographic features were the only part of the product that constituted a derivative work of GNU tar; the extraneous utilities merely made shell calls out to GNU tar. FSF requested that Vigorien come into compliance with the GPL by releasing the source of GNU tar, with the cryptographic modifications, to its customers.

Vigorien released the original GNU tar sources, but kept the cryptographic modifications proprietary. They argued that the security of their system depending on keeping the software proprietary and that regardless, USA export restrictions on cryptographic software prohibited such a release. FSF disputed the first claim, pointing out that Vigorien had only one option if they did not want to release the source: they would have to remove GNU tar from the software and not distribute it further. Vigorien rejected this suggestion, since GNU tar was an integral part of the product, and the security changes were useless without GNU tar.

Regarding the export control claims, FSF proposed a number of options, including release of the source from one of Vigorien's divisions overseas where no such restrictions occurred, but Vigorien argued that the problem was insoluble because they operated primarily in the USA.

The deadlock on the second issue was resolved when those cryptographic export restrictions were lifted shortly thereafter, and FSF again raised the matter with Vigorien. At that point, they dropped the first claim and agreed to release the remaining source module to their customers. They did so, and the violation was resolved.

24.2 Lessons Learned

1. **Removing the GPL'd portion of the product is always an option.** Many violators' first

response is to simply refuse to release the source code as the GPL requires. FSF offers the option to simply remove the GPL'd portions from the product and continue along without them. Every case where this has been suggested has led to the same conclusion. Like Vigorien, the violator argues that the product cannot function without the GPL'd components, and they cannot effectively replace them.

Such an outcome is simply further evidence that the combined work in question is indeed a modified version of the original GPL'd component. If the other components cannot stand on their own and be useful without the GPL'd portions, then one cannot effectively argue that the work as a whole is not a based on the GPL'd portions.

2. **The whole product is not always covered.** In this case, Vigorien had additional works aggregated. The backup system was a suite of utilities, some of which were the GPL and some of which were not. While the cryptographic routines were tightly coupled with GNU tar and clearly made a whole new combined work of both components, the various GUI utilities were separate and independent works merely aggregated with the distribution of the GNU-tar-based product.

3. **"Security" concerns do not exonerate a distributor from GPL obligations, and "security through obscurity" does not work anyway.** The argument that "this is security software, so it cannot be released in source form" is not a valid defense for explaining why the terms of the GPL are ignored. If companies do not want to release source code for some reason, then they should not base the work on GPL'd software. No external argument for noncompliance can hold weight if the work as a whole is indeed a modified version of a GPL'd program.

   The "security concerns" argument is often floated as a reason to keep software proprietary, but the computer security community has on numerous occasions confirmed that such arguments are entirely specious. Security experts have found — since the beginnings of the field of cryptography in the ancient world — that sharing results about systems and having such systems withstand peer review and scrutiny builds the most secure systems. While full disclosure may help some who wish to compromise security, it helps those who want to fix problems even more by identifying them early.

4. **External regulatory problems can be difficult to resolve.** The GPL, though grounded in copyright law, does not have the power to trump regulations like export controls. While Vigorien's "security concerns" were specious, their export control concerns were not. It is indeed a difficult problem that FSF acknowledges. We want compliance with the GPL and respect for users' freedoms, but we certainly do not expect companies to commit criminal offenses for the sake of compliance. We will see more about this issue in our next case study.

CHAPTER 25

HAXIL, POLGARA, AND THESULAC: MERGERS, UPSTREAM PROVIDERS AND RADIO DEVICES

This case study considers an ongoing (at the time of writing) violation that has occurred. By the end of the investigation period, three companies were involved and many complex issues arose.

25.1 The Facts

Haxil produced a consumer electronics device which included a mini GNU/Linux distribution to control the device. The device was of interest to many technically-minded consumers, who purchased the device and very quickly discovered that Free Software was included without source. Mailing lists throughout the Free Software community erupted with complaints about the problem, and FSF quickly investigated.

FSF confirmed that FSF-copyrighted GPL'd software was included. In addition, the whole distribution included GPL'd works from hundreds of individual copyright holders, many of whom were, at this point, up in arms about the violation.

Meanwhile, Haxil was in the midst of being acquired by Polgara. Polgara was as surprised as everyone else to discover the product was based on GPL'd software; this fact had not been part of the disclosures made during acquisition. FSF contacted Haxil, Polgara, and the product managers who had transitioned into the "Haxil division" of the newly-merged Polgara company. Polgara's General Counsel's office worked with FSF on the matter.

FSF formed a coalition with the other primary copyright holders to pursue the enforcement effort on their behalf. FSF communicated directly with Polgara's representatives to begin working through the issues on behalf of itself and the Free Software community at large.

Polgara pointed out that the software distribution they used was mostly contributed by an upstream provider, Thesulac, and Haxil's changes to that code base were minimal. Polgara negotiated with Thesulac to obtain the source, although the issue moved very slowly in the channels between Polgara and Thesulac.

FSF encouraged a round-table meeting so that high bandwidth communication could occur between FSF, Polgara and Thesulac. Polgara and Thesulac agreed, and that discussion began. Thesulac provided nearly complete sources to Polgara, and Polgara made a full software release on their Web site. At the time of writing, that software still has some build problems (similar to those that occurred with Bortez, as described in Section 22.1). FSF continues to negotiate with Polgara and Thesulac to resolve these problems, which have a clear path to a solution and are expected to resolve.

Similar to the Vigorien case, Thesulac has regulatory concerns. In this case, it is not export controls —

an issue that has since been resolved — but radio spectrum regulation. Since this consumer electronic device contains a software-programmable radio transmitter, regulations in (at least) the USA and Japan prohibit release of those portions of the code that operate the device. Since this is a low-level programming issue, the changes to operate the device form a single combined work with the kernel named Linux. A decade later, this situation remains largely unresolved.

25.2 Lessons Learned

1. **Community outrage, while justified, can often make negotiation more difficult.** FSF has a strong policy never to publicize names of GPL violators if they are negotiating in a friendly way and operating in good faith toward compliance. Most violations are honest mistakes, and FSF sees no reason to publicly admonish violators who genuinely want to come into compliance with the GPL and to work hard staying in compliance.

   This case was so public in the Free Software community that both Haxil's and Polgara's representatives were nearly shell-shocked by the time FSF began negotiations. There was much work required to diffuse the situation. We empathize with our community and their outrage about GPL violations, but we also want to follow a path that leads expediently to compliance. In our experience, public outcry works best as a last resort, not the first.

2. **For software companies, GPL compliance belongs on a corporate acquisition checklist.** Polgara was truly amazed that Haxil had used GPL'd software in a major new product line but never informed Polgara during the acquisition process. While GPL compliance is not a particularly difficult matter, it is an additional obligation that comes along with the product line. When planning mergers and joint ventures, one should include lists of GPL'd components contained in the products discussed.

3. **Compliance problems of upstream providers do not excuse a violation for the downstream distributor.** To paraphrase §6, upstream providers are not responsible for enforcing compliance of their downstream, nor are downstream distributors responsible for compliance problems of upstream providers. However, engaging in distribution of GPL'd works out of compliance is still just that: a compliance problem. When FSF carries out enforcement, we are patient and sympathetic when the problem appears to be upstream. In fact, we urge the violator to point us to the upstream provider so we may talk to them directly. In this case, we were happy to begin negotiations with Thesulac. However, Polgara still has an obligation to bring their product into compliance, regardless of Thesulac's response.

4. **It behooves upstream providers to advise downstream distributors about compliance matters.** FSF has encouraged Thesulac to distribute a "good practices for GPL compliance" document with their product. Polgara added various software components to Thesulac's product, and it is conceivable that such additions can introduce compliance. In FSF's opinion, Thesulac is in no way legally responsible for such a violation introduced by their customer, but it behooves them from a marketing standpoint to educate their customers about using the product. We can argue whether or not it is your coffee vendor's fault if you burn yourself with their product, but (likely) no one on either side would dispute the prudence of placing a "caution: hot" label on the cup.

5. **FSF enforcement often avoids redundant enforcement cases from many parties.** Most Free Software systems have hundreds of copyright holders. Some have thousands. FSF is in a unique position as one of the largest single copyright holders on GPL'd software and as a respected umpire in the community, neutrally enforcing the rules of the GPL road. FSF works hard in the community to convince copyright holders that consolidating GPL claims through FSF is better for them, and more likely to yield positive compliance results.

   A few copyright holders engage in the "proprietary relicensing" business, so they use GPL enforcement as a sales channel for that business. FSF, as a community-oriented, not-for-profit organization, seeks only to preserve the freedom of Free Software in its enforcement efforts. As it turns out, most of the community of copyright holders of Free Software want the same thing. Share and share alike is a simple

rule to follow, and following that rule to FSF's satisfaction usually means you are following it to the satisfaction of the entire Free Software community.

Generally, from the experience of GPL enforcement, we glean the following general practices that can help in GPL compliance for organizations that distribute products based on GPL'd software:

- Talk to your software engineers and ask them where they got the components they use in the products they build. Find out if GPL'd components are present.
- Teach your engineering staff to pay attention to license documents. Give them easy-to-follow policies to get approval for using a Free Software component.
- Build a "Free Software Licensing" committee that handles requests and questions about the GPL and other Free Software licenses.
- Add "What parts of your products are under the GPL or other Free Software licenses?" to your checklist of questions to ask when you consider mergers, acquisitions, or joint ventures.
- Encourage your engineers to participate collaboratively with GPL'd software development. The more knowledge about the Free Software world your organization has, the better equipped it is to deal with this rapidly changing field.
- When someone points out a potential GPL violation in one of your products, do not assume the product line is doomed. The GPL is not a virus; merely having GPL'd code in one part of a product does not necessarily mean that every related product must also be GPL'd. And, even if some software needs to be released that was not before, the product will surely survive. In FSF's enforcement efforts, we have not yet seen a product line die because source was released to customers in compliance with the GPL.

Part IV

Appendices

APPENDIX A

CITATIONS OF INCORPORATED MATERIAL FROM OTHER PUBLISHED WORKS

As a public, collaborative project, this Guide is primarily composed of the many contributions received via its public contribution process. Please review its Git logs for full documentation of all contributions.

Below is a list of CC-By-SA-licensed works, with specific titles and publication dates, from which any material was incorporated into this Guide. The specific methods and details of incorporation are fully documented in the Git logs of the project.

- *GPLv3 First Discussion Draft Rationale*, published by the Free Software Foundation on 2006-01-16.
- *GPLv3 Second Discussion Draft Rationale*, published by the Free Software Foundation circa 2006-07.
- *GPLv3 Third Discussion Draft Rationale*, published by the Free Software Foundation on 2007-03-28.
- *GPLv3 Discussion Draft 3 FAQ*, published by the Free Software Foundation on 2007-03-28.
- *GPLv3 Final Discussion Draft Rationale*, published by the Free Software Foundation on 2007-05-31.
- *GPLv3 Final Rationale*, written and published by the Free Software Foundation on 2007-06-29.
- *A Practical Guide GPL Compliance*, written by Bradley M. Kuhn, Aaron Williamson and Karen Sandler and published by the Software Freedom Law Center on 2008-08-20.
- *Software Freedom Law Center Guide to GPL Compliance, 2nd Edition*, written by Eben Moglen and Mishi Choudhary and published by the Software Freedom Law Center on 2014-10-31.
- *Detailed Analysis of the GNU GPL and Related Licenses*, written by Bradley M. Kuhn, Daniel B. Ravicher, and John Sullivan and published by the Free Software Foundation for its CLE courses on 2004-01-20, 2004-08-24, and 2014-03-24.
- *Enforcement Case Studies*, written by Bradley M. Kuhn and published by the Free Software Foundation for its CLE courses on 2004-01-20, 2004-08-24, and 2014-03-24.

Please note, however, that this list above does not include nor adequately represent the substantial contributions from those who directly contributed to this Guide using its Git (and formerly, CVS) repository. Rather, this is a list of third-party published works from which any text was herein included under their

CC-By-SA licensing. Thus, as the reader might expect, the version control logs contain the only true and accurate view available of who has contributed which portions of this project.

The remaining appendices include a full copy of GPLv2, GPLv3, LGPLv2.1, LGPLv3, and AGPLv3. These are the most commonly used licenses in the GPL family of licenses.

APPENDIX B

THE GNU GENERAL PUBLIC LICENSE, VERSION 2

Version 2, June 1991

59 Temple Place - Suite 330, Boston, MA 02111-1307, USA

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 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

11. Because the program is licensed free of charge, there is no warranty for the program, to the extent permitted by applicable law. Except when otherwise stated in writing the copyright holders and/or other parties provide the program "as is" without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The entire risk as to the quality and performance of the program is with you. Should the program prove defective, you assume the cost of all necessary servicing, repair or correction.

12. In no event unless required by applicable law or agreed to in writing will any copyright holder, or any other party who may modify and/or redistribute the program as permitted above, be liable to you for damages, including any general, special, incidental or consequential damages arising out of the use or inability to use the program (including but not limited to loss of data or data being rendered inaccurate or losses sustained by you or third parties or a failure of the program to operate

WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

End of Terms and Conditions

Appendix: How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it Free Software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

> one line to give the program's name and a brief idea of what it does.
> Copyright (C) yyyy name of author
>
> This program is Free Software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.
>
> This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.
>
> You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

Also add information on how to contact you by electronic and paper mail.
If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

> Gnomovision version 69, Copyright (C) yyyy name of author
> Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type 'show w'.
> This is Free Software, and you are welcome to redistribute it under certain conditions; type 'show c' for details.

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.

APPENDIX C

THE GNU LESSER GENERAL PUBLIC LICENSE, VERSION 2.1

Version 2.1, February 1999

59 Temple Place - Suite 330, Boston, MA 02111-1307, USA

[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 applicable law. Except when otherwise stated in writing the copyright holders and/or other parties provide the library "as is" without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The entire risk as to the quality and performance of the library is with you. should the library prove defective, you assume the cost of all necessary servicing, repair or correction.

16. In no event unless required by applicable law or agreed to in writing will any copyright holder, or any other party who may modify and/or redistribute the library as permitted above, be liable to you for damages, including any general, special, incidental or consequential damages arising out of the use or inability to use the library (including but not limited to loss of data or data being rendered inaccurate or losses sustained by you or third parties or a failure of the library to operate with any other software), even if such holder or other party has been advised of the possibility of such damages.

End of Terms and Conditions

How to Apply These Terms to Your New Libraries

If you develop a new library, and you want it to be of the greatest possible use to the public, we recommend making it Free Software that everyone can redistribute and change. You can do so by permitting redistribution under these terms (or, alternatively, under the terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

> one line to give the library's name and a brief idea of what it does.
> Copyright (C) year name of author
>
> This library is Free Software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.
>
> This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.
>
> You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the library, 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 1990
> Ty Coon, President of Vice

APPENDIX D

THE GNU GENERAL PUBLIC LICENSE, VERSION 3

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program–to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. 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 them 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 prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. 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.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other

domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

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

Terms and Conditions

0. Definitions.

   "This License" refers to version 3 of the GNU General Public License.

   "Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

   "The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

   To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

   A "covered work" means either the unmodified Program or a work based on the Program.

   To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

   To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

   An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

   The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

   A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

   The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

   The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface

definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

   All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

   You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

   Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

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

   No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

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

4. Conveying Verbatim Copies.

   You may convey 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; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

   You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

   You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

   (a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

   (b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

(c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

(d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

(a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

(b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

(c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

(d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

(e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of

the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

(a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

(b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

(c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

(d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

(e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

(f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

   You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

   However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

   Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

   Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

   You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

    Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

    An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

    You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

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

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If 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 convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

    Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

    The Free Software Foundation may publish revised and/or new versions of the GNU 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 that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

    If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

    Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

    THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

    IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

    If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

End of Terms and Conditions

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief idea of what it does.>

Copyright (C) <textyear>  <name of author>

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.
```

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

```
<program>  Copyright (C) <year>  <name of author>

This program comes with ABSOLUTELY NO WARRANTY; for details type 'show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type 'show c' for details.
```

The hypothetical commands `show w` and `show c` should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see `http://www.gnu.org/licenses/`.

The GNU 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 Lesser General Public License instead of this License. But first, please read `http://www.gnu.org/philosophy/why-not-lgpl.html`.

APPENDIX E

THE AFFERO GENERAL PUBLIC LICENSE, VERSION 3

Version 3, 19 November 2007

Preamble

The GNU Affero General Public License is a free, copyleft license for software and other kinds of works, specifically designed to ensure cooperation with the community in the case of network server software.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, our General Public Licenses are intended to guarantee your freedom to share and change all versions of a program–to make sure it remains free software for all its users.

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 them 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.

Developers that use our General Public Licenses protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License which gives you legal permission to copy, distribute and/or modify the software.

A secondary benefit of defending all users' freedom is that improvements made in alternate versions of the program, if they receive widespread use, become available for other developers to incorporate. Many developers of free software are heartened and encouraged by the resulting cooperation. However, in the case of software used on network servers, this result may fail to come about. The GNU General Public License permits making a modified version and letting the public access it on a server without ever releasing its source code to the public.

The GNU Affero General Public License is designed specifically to ensure that, in such cases, the modified source code becomes available to the community. It requires the operator of a network server to provide the source code of the modified version running there to the users of that server. Therefore, public use of a modified version, on a publicly accessible server, gives the public access to the source code of the modified version.

An older license, called the Affero General Public License and published by Affero, was designed to accomplish similar goals. This is a different license, not a version of the Affero GPL, but Affero has released a new version of the Affero GPL which permits relicensing under this license.

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

Terms and Conditions

0. Definitions.

"This License" refers to version 3 of the GNU Affero General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

   All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

   You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

   Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

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

   No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

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

4. Conveying Verbatim Copies.

   You may convey 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; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

   You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

   You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

   (a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

   (b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

   (c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

   (d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

(a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

(b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

(c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

(d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

(e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

(a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

(b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

(c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

(d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

(e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

(f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

   You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

   However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

   Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

   Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

   You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

    Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

    An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

    You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

    A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

    A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of

this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

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

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

    If 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 convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

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

    Notwithstanding any other provision of this License, if you modify the Program, your modified version must prominently offer all users interacting with it remotely through a computer network (if your version supports such interaction) an opportunity to receive the Corresponding Source of your version by providing access to the Corresponding Source from a network server at no charge, through some standard or customary means of facilitating copying of software. This Corresponding Source shall include the Corresponding Source for any work covered by version 3 of the GNU General Public License that is incorporated pursuant to the following paragraph.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the work with which it is combined will remain governed by version 3 of the GNU General Public License.

14. Revised Versions of this License.

    The Free Software Foundation may publish revised and/or new versions of the GNU Affero 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 that a certain numbered version of the GNU Affero General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU Affero General Public License, you may choose any version ever published by the Free Software Foundation.

    If the Program specifies that a proxy can decide which future versions of the GNU Affero General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

    Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

    THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

    IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

    If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

End of Terms and Conditions

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief idea of what it does.>

Copyright (C) <textyear>  <name of author>

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.
```

Also add information on how to contact you by electronic and paper mail.

If your software can interact with users remotely through a computer network, you should also make sure that it provides a way for users to get its source. For example, if your program is a web application, its interface could display a "Source" link that leads users to an archive of the code. There are many ways you could offer source, and different solutions will be better for different programs; see section 13 for the specific requirements.

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU AGPL, see `http://www.gnu.org/licenses/`.