# 6

# Reciprocity and the GPL

## The GPL Bargain

The world of software was transformed by the GNU General Public License. The word *GNU* in the license name is a play on words by the license author, Richard Stallman. “The name GNU was chosen following a hacker tradition,” he says, “as a recursive acronym for ‘GNU’s Not Unix.’” Throughout the world, the license is mostly referred to simply as the *GPL.* (The GPL is reprinted in the Appendices.)

The GPL has been enormously influential in creating a large public commons of software that is freely available to everyone worldwide. As the GPL advocates might describe it in political tones, they have prevented much software from being captured by proprietary software interests and converted into restricted private property for personal gain. The GPL is both praised and reviled for that accomplishment.

The bargain created by the GPL can be paraphrased simply as follows: You may have this free software on condition that any derivative works that you create from it and distribute must be licensed to all under the same license.

Here’s how the GPL actually says it:

> *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. (GPL, Section 2.)*

This is the most powerful idea in the GPL and the one that has aroused the most passion in its adherents and its detractors.

Adherents of the GPL suggest that this provision protects free software. It guarantees that all derivative works of GPL-licensed software will also be GPL-licensed software. Licensees cannot selfishly remove their improvements from the public commons. Derivative work software will always be free and open. The result is a dynamic and ever growing collection of GPL-licensed software that can be reused and improved.

Detractors say that this provision creates an island of software from which only GPL-licensed software can escape. The rest of the world cannot share the benefits of the source code of GPL-licensed software unless they are willing to travel to that island and commit to using the GPL license for their works.

Some of these GPL detractors are licensors of *proprietary* software. Their complaints are hypocritical. They too have created islands of software from which nothing can escape.

The only principled complaint about the GPL comes from those who license their software under *academic* open source licenses. Such software can be incorporated into GPL-licensed software but the converse is not true. In one sense, academic licenses are for *generous donors* of software, and the GPL and other reciprocal licenses are for *generous sharers* of software. Because of the GPL we have two—not just one—public commons of free software.

This book is not the place to resolve this ongoing debate. It is enough to say that licensors are free to decide what licensing model suits them best and whether or how to give away rights. Licensees may accept or reject software under the terms of the license, but they don't get to set their own terms. That's what copyright law allows, and the GPL uses that law effectively and brilliantly for its avowed purpose of fostering the creation of free software available to all under a single license.

## Copyleft and Reciprocity

Partly to emphasize the role of copyright law to protect the freedom of GPL-licensed software and partly to create a catchy term to highlight their focus on software freedom, the authors of the GPL coined the term *copyleft* to describe its license bargain. It is both a play on the word *copyright* and an acknowledgment that it promoted a radical (i.e., *left-wing*, perhaps) departure from traditional software licensing models. The role of a *copyleft* software license is to grow the public commons of software rather than allow each owner's *copyright* to pull from that commons.

The Free Software Foundation also describes *copyleft* as a rule that, when redistributing a program, one cannot add restrictions to deny other people the central software freedoms. The word *restriction* is very vague in a licensing context; almost any of the terms and conditions in a license can be described as a restriction of some sort. This limitation on restrictions in the definition of copyleft causes some attorneys, including me, heartburn. We contend it would be helpful to add some restrictions to open source licenses that the GPL's authors didn't think of when they wrote their license. For example, provisions for defense against patent infringement lawsuits or to protect the licensor's trademarks can be very useful; both provisions are missing from the GPL.

In practice, the Free Software Foundation's restriction on adding restrictions has had the effect of allowing them to veto any restriction they find unacceptable—even those that are improvements over the GPL. Their avoidance of restrictions has delayed the adoption of new and useful licensing concepts for open source software. This topic will be addressed again when I discuss license compatibility in Chapter 10.

I find the word *reciprocity* to be less alarming and more descriptive than the word *copyleft*. I particularly like that word because it does not carry with it the reference to *restrictions* espoused by the Free Software Foundation.

> *Reciprocity means a mutual or cooperative interchange of favors or privileges. Something is reciprocal when it is performed, experienced, or felt by both sides. (The American Heritage Dictionary of the English Language, 4th edition.)*

The GPL license is reciprocal, because it is "performed, experienced, or felt" by both sides—the licensor and the licensees both use the GPL.

For these reasons, I refer to *reciprocity* rather than *copyleft*. The term *copyleft*, of course, needn't disappear. It still has great rhetorical value. It is a useful word to toss back at those who mistakenly complain that the GPL destroys copyrights; the GPL requires copyright law to create a copyleft bargain. But I do not find the term useful and I won't use that word again in this book.

Reciprocity provisions are now quite common in open source licenses; the GPL is merely the first and most influential proponent of that particular software bargain. The reciprocity obligations of other open source licenses are subtly different. I shall explore those differences when the individual licenses are discussed. But first, I must explore the policy objectives of the GPL, as much as possible in its authors' own words.

## Policy Objectives

Traditional software licenses serve business needs. Their objective is usually to maximize profit from licensing of the software. The GPL has an entirely different policy objective. It seeks to maximize the amount of free software available in the public commons.

The authors of the GPL point out that placing software into the public commons under an academic open source license doesn't always serve that important purpose. Any licensee under an academic open source license can take that free software, create derivative works from it, and then distribute those derivative works under a proprietary license. The resulting software is not free. The Free Software Foundation politely characterizes these licensees as "uncooperative people."

> *They can make changes, many or few, and distribute the result as a proprietary product. People who receive the program in that modified form do not have the freedom that the original author gave them; the middleman has stripped it away. (From www.fsf.org.)*

The GPL seeks to prevent that situation by imposing a reciprocity obligation on all such middlemen. Licensees must use the GPL as their license if they distribute modified versions of the software. Any resulting derivative works will also be free software.

The GPL also seeks to prevent a software problem that was common in the early 1990s and continues to this day. Many software vendors believe that the only path to profit is through the creation of unique proprietary versions of standard software. This leads to software incompatibility, ultimately locking customers into specific vendors, reducing meaningful choices for consumers, and creating roadblocks to software sharing. The story of UNIX is replete with examples of that. (Eric Raymond's book, *The Art of UNIX Programming,* paints a turbulent history of the various proprietary forks of the UNIX operating system.)

By the time Linux was released under the GPL, there were many versions of UNIX available from many vendors, many of them incompatible with each other. Now, under the GPL, there can be many versions of Linux, but the improvements in any of them can be incorporated back into the rest of them as market forces dictate. There are no longer licensing obstacles to taking the best components of Linux software available anywhere and incorporating them back into anyone else's version of Linux. Compatibility can be created at will by any licensee of Linux. That is guaranteed by the GPL.

A third policy point of the GPL is that free software is an ethical objective, distinct from the practical objective of making the source code of software available to licensees. Free software, they say, is a good in itself.

> *Whatever approach you use, it helps to have determination and adopt an ethical perspective, as we do in the Free Software Movement. To treat the public ethically, the software should be free—as in freedom—for the whole public. (See www.fsf.org.)*

Because this is a book about the law of licensing rather than ethics, I will only make two comments about this. First, contract and copyright law doesn't generally deal with the ethical concerns of private parties; courts are expected to interpret the plain language of their license agreements in accordance with legal principles only. Second, whether you agree or disagree with the ethics of a licensor, accepting software under a license binds you to the terms of that license; you need only concern yourself with doing what you agreed to, not with whatever gods or demons the licensor prays to.

## The Preamble to the GPL

Richard Stallman and Eben Moglen, the authors of the GPL, write eloquently in the GPL's preamble about their primary objective in creating the license:

> *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. (GPL Preamble.)*

Eloquence, by the way, and discussions of public policy, are extremely rare in licenses; attorneys will recall no other such example from their law school courses in technology licensing. That is one feature that stands out about the GPL. It was the obvious intention of the authors of the GPL to arouse licensors and licensees to a higher purpose than the mere distribution of software. Strong and convincing language was called for. It is thus perhaps not surprising that some of the harshest critics of the GPL, and many of its most fervent admirers, point to the preamble to that license when engaging each other in political debate about free software.

The preamble, of course, is not an operative part of the GPL license. It is not among its *terms and conditions*. There is nothing in its words that must be obeyed. It is merely a helpful preface so that you can better understand the GPL in its context.

The preamble proceeds to define *free software*:

*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. (GPL Preamble.)*

You will note that this is a shorthand definition of software freedom, shorter even than the definition on the Free Software Foundation website quoted in the first chapter of this book. This paragraph from the license is not the authoritative definition of free software. Unfortunately, arguments about precisely what *free software* means have engaged the open source community and perplexed the public for some time now. This additional definition in the preamble to the GPL doesn't help.

The next paragraph foretells the reciprocity bargain of the GPL:

*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. (GPL Preamble.)*

This paragraph is particularly interesting. It subtly transforms what had previously been a focus on *freedom* to a statement about *rights*. It suggests to me a number of questions that have no easy answers, at least within the four corners of the GPL license or its preamble: How does a freedom become a right? Whose rights are being protected by the GPL, and from whom? Who is trying to deny you those rights, and who has the authority to forbid them from doing it? Can someone make you surrender a right simply by asking? Do the GPL restrictions effectively protect you from those awful prospects?

Why do you have to incur responsibilities to protect your own rights?

The fact that these questions have no ready answers points out once again why preambles are bad in licenses. Preambles are not helpful, and they potentially confuse. They are too brief and too ambiguous to guide in the interpretation of the license. And to the extent that they raise discomforting questions for potential licensors and licensees—and their attorneys—they discourage the adoption of the license.

Fortunately, the GPL preamble has no legal significance and is not going to matter if the license is ever litigated in court. But it is still worth reading and analyzing to understand the license authors' achievements and possible misconceptions.

For example, the first sentence of the next paragraph of the GPL preamble is technically incorrect and the rest of that paragraph is misleading:

> *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. (GPL Preamble.)*

The problem with that first sentence is with the word *all* in the phrase *all the rights that you have*. Neither the free software guidelines nor the open source definition require a licensor to grant *all* his rights; he retains, for example, the right to grant licenses to his own software under different terms than the GPL, and the right to refuse to issue new licenses. Technically, a copyright owner retains all his or her rights and merely grants licenses to others in accordance with certain terms and conditions. The phrase *give the recipients all the rights that you have* is unnecessarily frightening and is not true.

I think what that first sentence intends to say is that, when you sublicense a GPL-licensed work, you must pass along the software under its original license without adding further restrictions. Only in that context does the rest of that paragraph make sense: You received source code when you received the GPL-licensed work, and so you must provide source code when you sublicense it. And to make sure that your sublicensees know that they have the rights to copy, modify and distribute the software and the right to the source code, you must provide them with a copy of the GPL license text, just as you were provided with this copy of the license.

The next paragraphs of the preamble anticipate that the GPL will give licensees "legal permission to copy, distribute, and/or modify the software"; will not provide a warranty; will protect the reputations of the original authors; and will deal effectively with the threat of patents. The actual license terms for this, of course, are not these in the preamble, but those that follow later in the license, under the heading "TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION."

## GPL as Template

The GPL devised an elegant solution to the problem of associating a generic software license with specific software. Instead of placing the name of the software in the license (as is usually and inconveniently done with proprietary and many other open source software licenses), it requires that a notice be placed in the software by the copyright holder saying, "it may be distributed under the terms of this General Public License." Where this notice is to be placed is not specified, but at the end of the GPL, in a separate nonbinding section entitled "How to Apply These Terms to Your New Programs," the

GPL suggests that the notice should be at the start of each source file.

The GPL is thus a template license, applicable to software by any author who chooses to use it. All a licensor has to do to use it is to include a notice in his or her source code saying, in effect, "I'm licensing this software to you under the GPL." But which GPL? There was at least one earlier version and there are promises of another version to come. How can a licensor indicate which version of the GPL applies to the licensed software? Here's how the GPL handles that situation:

> *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. (GPL section 9.)*

Some licensors object to giving anyone, including the Free Software Foundation, the opportunity to change the licensing rules for his or her own software after the software has already been distributed under a specific license. Those licensors, then, should be explicit in the notices they place in the software, being careful to identify that they are "licensing this software to you under the GPL version 2" if that is specifically what they intend.

## The GPL Applies to Programs

Rather than use the generic term *software*, the GPL instead defines the term *Program* as *a program or other work*. We generally understand that a *program* (with lower case *p*) is computer software, but the phrase *other work* is left undefined.

The GPL, in section 0, then defines the phrase *work based on the Program* as either a *Program* or a *derivative work under copyright law*. (Careful readers will remember that, under copyright law a *derivative work* is a work based upon one or more preexisting works.... 17 U.S.C. § 101.) This definition is repeated in a different way in section 1 of the GPL, which says that a *work based on the Program* is formed by modifying the original Program. (Remember that a modification is one of the specific kinds of derivative works mentioned in the copyright law.) Thus far, the GPL is entirely consistent with copyright law definitions, and so it applies to Programs and to derivative works of those Programs.

Unfortunately, the section 0 definition of *work based on the Program* is then broadened beyond what is generally considered in the copyright law to be a derivative work:

> *...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. (GPL section 0.)*

Is a work based on the Program really the same as a work containing the Program or a portion of it?

I have already explained the fundamental difference in copyright law between a *collective work* and a *derivative work*. You will recall generally that the former is a collection of independent works and the latter is a work based upon one or more preexisting works. A work containing another work is a *collective work*. A work based on another work is a *derivative work*. Merging those concepts in the GPL would leave no distinction between a derivative and collective work, an absurd result considering the importance of those two defined terms in copyright law.

The issue is critical for another reason. It is the basis for a long-running dispute about the reach of the GPL to separate unmodified programs that merely link to each other but that are collected into one program for convenience. If, through linking to a program that is included in a collective work, one creates a *derivative work*, how widely does the *reciprocity* obligation of the GPL reach?

## Linking to GPL Software

It is appropriate to look within the four corners of the GPL itself for guidance on this question about program linking.

The word *link* actually occurs only once in the official GPL, way at the end in the last paragraph of a nonbinding section called "How to Apply These Terms to Your New Programs." This paragraph deals with a different license, the LGPL, which I will describe in due course.

There are other provisions of the GPL that refer to *work based on the Program*. Here is the first possibly helpful reference:

> *...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. (GPL section 0.)*

This provision deals with the special case of a Program that generates other programs that contain either verbatim or modified/translated versions of itself. Such an esoteric example of program interdependence is best ignored in a general book like this about open source licensing. It is not likely to be encountered in typical open source applications.

The GPL, in section 2, then requires us to analyze the software based not upon how it is *linked* but upon how it is *distributed*. Because it will be helpful to parse this provision carefully, I quote each sentence separately.

> *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. (GPL section 2.)*

According to the first sentence, the entire GPL applies to a "modified work as a whole." Under the copyright law, such a "modified work" is a derivative work. (17 U.S.C. § 101.) So far, there is no hint that linking makes a difference.

The second sentence refers to portions of the work that "are not derived from the Program"—that is, are not derivative works. This necessarily means works that have their own copyrights, their own copyright owners, and potentially their own licenses. So the second sentence is true regardless of whether the independent and separate works are linked in some way to the GPL software. Such works remain "independent and separate works," at least "when you distribute them as separate works," and the GPL cannot possibly apply to them without their copyright owner's consent.

The third sentence refers to those "independent and separate works" when they are distributed "as part of a whole." Once again, we are reminded that the GPL applies to the whole work. But how are we to understand its reference to "the same sections as part of a whole which is a work based on the Program" and later "to each and every part regardless of who wrote it"? Is this a reference to the Copyright Act?

> *The copyright in a compilation or derivative work extends only to the material contributed by the author of such work, as distinguished from the preexisting material employed in the work, and does not imply any exclusive right in the preexisting material. (17 U.S.C. § 103.)*

All that the third sentence of GPL section 2 could possibly mean under the copyright law is that, for a work to be made available under the GPL, its preexisting component parts must be available to all subsequent licensees. The licenses to those components must permit that combination. That much is necessarily true for any software containing components licensed by others. The law makes it clear that the GPL can't affect the licenses to those preexisting component parts. Again, linking doesn't matter.

The GPL then expresses its intent this way:

> *The intent is to exercise the right to control the distribution of derivative or collective works based on the Program. (GPL section 2.)*

That may be the intent, but is that what the GPL actually does? This is a critical example of imprecise phrasing. Who gets "to exercise the right to control" distribution? Certainly the owner of a collective or derivative work gets "to exercise the right to control" those works, and the owner of each contribution gets "to exercise the right to control" his or her contribution. (17 U.S.C. § 103[b].)

Does the phrase *based on the program* refer to both derivative and collective works? That isn't technically correct, at least under the U.S. Copyright Act, because a derivative work is a work based on one or more preexisting works, but a collective work is not. (17 U.S.C. § 101.) There is still no meaningful clue about linkage.

This entire GPL provision in section 2 relating to distribution of the *whole work* is technically trivial to avoid. Some open source projects, trying to stay on the "safe" side of this GPL provision, advise their customers to separately download and install required non-GPL software merely to avoid "distribution as part of a whole." Thus the distinction drawn by this part of GPL section 2 has become an inconvenience rather than a meaningful requirement.

Finally the GPL directly addresses the distribution of collective works, noting that the GPL does not apply to them:

> *...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. (GPL section 2.)*

This sentence seems to mean that only *derivative* works are covered by the GPL reciprocity provision, and that "mere aggregation" of separate works onto common media (or common computer memory?) does not require reciprocity, even if those mere aggregations are distributed in one unit (i.e., "as part of the whole").

We are left with uncertainty—and instructions to contact the author of the Program for guidance:

> *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. (GPL section 10.)*

Some authors have indeed provided that guidance. Linus Torvalds, for example, has set a policy that software that is merely combined with Linux is not subject to the GPL regardless of how that software is linked and distributed.

## Copyright Law and Linking

Why do I spend so much time dealing with issues of software linking? Does this topic really matter to anyone but open source zealots?

Consider the metaphor of the World Wide Web, a huge collection of individually written web pages that anyone can access and display just by linking. Those pages are individually copyrighted works, made available to all by their authors, generally for free (i.e., at zero price). Under the copyright law, you do not create a derivative work of someone's web page by linking to it, nor is it a derivative work of your web page if it links to you.

At most, such linkages create collective works. A web page, for example, that contains links to articles about open source may present those links in an original, copyrightable way. That list of links is a copyrightable *original work of authorship*, and the links operate to create a *collective work*. But the original articles remain the copyrightable works of their own authors.

Not that we can't envision using the Internet to create derivative works of web pages. You can find a web page you like and make changes to it, using the modified version as your own. You can translate a web page from one language to another. You can provide editorial revisions, annotate the web pages, or elaborate upon them. You can then link to your new versions. In doing so, you create *derivative works*. But it is not the linking that made the difference.

I do not want to discourage the creation of *collective works*. To do so would be inconsistent with the goals of free and open source software, just as it would be inconsistent with the goals of a free and open World Wide Web. Are the GPL's *Programs* so different from other copyrightable works that they deserve a narrower range of freedom?

Rosen_ch06 Page 120 Wednesday, June 23, 2004 9:59 AM

One could, of course, ask the authors of the GPL how to interpret their license provisions, and they have indeed spoken out about this topic on their website, *www.fsf.org*, and in other public venues. But it is legally unnecessary to know what the drafter of a license—usually just an attorney with no stake in the matter—meant to say. That is why I can legally ignore the advisory notice that is published with the GPL after its terms and conditions have ended:

> *This General Public License does not permit incorporating your program into proprietary programs. (See "How to Apply These Terms to Your New Programs.")*

Under the law, only the common understanding of a licensor and his licensees matters, as reflected in the written terms and conditions of the license agreement between them. It is Linus Torvalds, and the thousands of other licensors under the GPL, who have standing under the law to assert their interpretations of the GPL, not the Free Software Foundation (except for that software for which *they* own the copyrights). And it is a judge who would ultimately decide such an issue if it reaches that level of conflict.

One final warning: If there is an ambiguity or uncertainty of interpretation in a license, the license will generally be interpreted *against* the licensor regardless of what the license drafter meant to say. It is up to the authors of the GPL to make their license clear, not up to licensees to seek outside guidance to interpret it. I explore that issue further in Chapter 12.

I won't give legal advice of a general nature to the readers of this book. So you can take with a grain of salt my belief that these interrelated sections of the GPL quoted earlier will ultimately be read by the courts to mean that *derivative works* are subject to the GPL's reciprocity provision, but *collective works* are not. And as I shall argue again more fully in the discussion of derivative works litigation in Chapter 12, the legal analysis of what constitutes a derivative work simply doesn't depend upon the style or mechanism of inter-program linking.

This, by the way, is also the only interpretation that is consistent with item 5 of the Open Source Principles listed in Chapter 1, that allows licensees freely to combine open source and other software.

## The LGPL Alternative

Originally called the *Library GPL*, this special version of the GPL directly addresses the linking question. It is now called the *Lesser General Public License*, or *LGPL* for short. Advisory text at the end of the published GPL license (but not one of its terms and conditions) encourages the use of the LGPL for certain applications:

> *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. (GPL, "How to Apply These Terms to Your New Programs" following GPL Terms and Conditions.)*

The LGPL is an important, widely used open source license in its own right. The complete text of the terms and conditions of that license, leaving out the extraneous preamble and postscripts, is shown in the Appendices.

The LGPL is for the distribution of software libraries.

> *A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs ... to form executables. (LGPL section 0.)*

This definition suggests that a *library* is designed with a goal in mind: It is "to be conveniently linked with application programs to form executables." The important characteristics of a library are not the form of linkage used by the members of that collection, nor the specific functions and/or data that are prepared. The LGPL is, after all, a general purpose license intended for adoption by software in many technological forms.

Here is how that same definition might be rephrased in copyright law terms: A "Library" is an original work of authorship that is intended to be incorporated into other works through some form of linkage.

The LGPL then grants a license for the Library to be used in its intended way:

> *The act of running a program using the Library is not restricted.... (LGPL section 0.)*

The LGPL repeats this same point a second time:

> *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. (LGPL section 5.)*

Both license grants are consistent with copyright law, of course, and nobody could reasonably suggest that mere invocation of a Library, however the linkage takes place, is a *derivative work*.

Modifications of a *Library* itself, of course, are derivative works, subject to the LGPL's reciprocity provision, just as modifications to any *Program* are subject to the GPL's reciprocity provision when you distribute those modifications:

> *You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License. (LGPL section 2[c].)*

Two other requirements from LGPL section 2, however, are not so clear:

> *The modified work must itself be a software library. (LGPL section 2[a].)*

Is this a definition or a requirement? How is it to be satisfied by a diligent licensee? And later:

> *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 2[d] 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.) (LGPL section 2[d].)*

And still later:

> *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. (LGPL section 5.)*

These sections of the LGPL are an impenetrable maze of technological babble. They should not be in a general-purpose software license. The LGPL even concedes that "the threshold for this to be true is not precisely defined by law." (LGPL section 5.) A licensee under these provisions won't have a clue how extensive his or her *good faith efforts* must be when creating a *derivative work* in accordance with sections 2(d) and 5 of the LGPL.

In any event, a careful comparison of the text of the GPL and LGPL licenses (far too detailed and specific to attempt here) reveals that, if the process of adding or deleting library functions creates a *derivative work* of the Library, then the LGPL functions identically to the GPL.

The LGPL concedes that the GPL is a better, more appropriate license, and it allows any licensees to convert to the GPL at their option:

> *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. (LGPL section 3.)*

The LGPL, therefore, is an anomaly—a hybrid license intended to address a complex issue about program linking and derivative works. It doesn't solve that problem but merely directs us back to the main event, the GPL license itself.

## GPL Grant of License

The first place in its terms and conditions that the GPL mentions its license grant is in the negative:

> *Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. (GPL section 0.)*

Thus are the first three exclusive rights of a copyright owner from 17 U.S.C. § 106 introduced. (Refer to the discussion of the exclusive rights of copyright owners in Chapter 2.) The license grant is stated in an affirmative way later in the GPL:

> *You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided.... (GPL section 1.)*
>
> *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.... (GPL section 2.)*
>
> *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.... (GPL section 3.)*

These, plus the source code grant discussed in the next section, are the required grants to comply with the Open Source Principles listed in Chapter 1.

You may have noted that the GPL does not grant *all* the rights under copyright; missing are licenses to perform the work or to display the work publicly. For most software, that's not important.

There are more interesting things than that missing from the GPL's license grant. The first and most important is a patent grant. The GPL does not expressly grant rights to make, use, sell or offer for sale, or import software that embodies the licensor's patents. This omission is important for a bare license like the GPL, because nothing in the law requires the licensor of copyrights to also license his patents. Bare patent licenses are not implied.

The GPL attempts to solve this problem by including the following condition:

> *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. (GPL section 7.)*

In other words, a licensor cannot distribute software under the GPL while simultaneously demanding royalties for his patents. His act of distributing the software implies a royalty-free license.

As to the scope of such an implied patent license, can we assume that it extends to the creation of derivative works since the GPL contemplates that licensees will create derivative works? That is possible, but there's nothing in the law of bare licenses that requires that result. Any company intending to create and distribute derivative works under the GPL ought to obtain separately the patent licenses it needs.

The second item that is missing is a statement of what other intellectual property rights, if any, are intentionally excluded from the license grant. For example, suppose a GPL-licensed program bears a trademark and that trademark is printed out by the program in some initial welcome message. Does a licensee under the GPL have the right to apply that trademark to his or her own derivative works? Must the licensee remove the trademark from executable versions of this derivative work? The GPL is silent on that point.

The GPL is also silent about the scope and duration of the licenses it does grant. One can assume that the license is *worldwide*, consistent with the open source definition. One can also assume that the license is *perpetual*, since there is no mechanism for terminating the license as long as the licensee complies with the terms of the license:

> *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. (GPL section 4.)*

This GPL section 4, with its negative wording, is also the only place that references the right to *sublicense*. One might assume from the way GPL section 4 is worded that the right to sublicense was intended in sections 1 (right to copy), 2 (right to modify) and 3 (right to distribute) as well. However, section 6 implies that there are no sublicenses but instead a direct license from each up-stream contributor:

Rosen_ch06 Page 128 Wednesday, June 23, 2004 9:59 AM

*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.... (GPL section 6.)*

As to sublicensing, then, the GPL is ambiguous. I refer you to the discussion in Chapter 5 of sublicensing in the MIT license. Sublicensing rights can be very important to open source distributors for dealing properly with the chain of title to contributions. In practice, most software projects ignore the issue completely and assume that, for GPL software, only the most recent license in the chain of title matters. They assume that GPL licensed software is sublicenseable, but the GPL isn't clear about that.

## Access to Source Code

The GPL allows licensees to copy and distribute the source code:

*You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided.... (GPL section 1.)*

Source code is defined as follows:

*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. (GPL section 3.)*

This is a broad definition and its intent is obviously to ensure that usable source code is available for licensed software. Deliberate obfuscation of the source code (as has been rumored to have been done by some GPL licensors) is potentially actionable as bad faith.

The GPL then offers a curious *special exception* for software that is normally distributed with the operating system on which the Program runs:

> *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. (GPL section 3.)*

The licensor of the operating system, and not the licensor of the Program, is the only one who can elect to publish his or her own source code. The GPL cannot possibly grant that permission or provide an exception relating to it. As was discussed at length in the previous section, the fact that software is merely distributed with the Program doesn't bring it under the GPL. This "special exception" is irrelevant if one accepts that only derivative works, and not collective works, are brought under the GPL.

As one of the conditions for distributing a Program or a derivative work of the Program in object code form, the licensor must also commit to 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.) (GPL section 3.)*

A licensor under the GPL is expected to distribute or make available the source code for software he or she writes. That is what items a) and b) of GPL section 3 require. But what is the licensor's obligation regarding the source code of GPL-licensed software that he or she merely distributes, perhaps as a component of a GPL-licensed collective or derivative work? Must the licensor undertake to distribute the source code to all the contributions of the entire collective work, including components he or she didn't write? Item c) would appear to solve this problem, but only for *noncommercial distribution*. I believe that, in practice, most distributors under the GPL provide source code for their entire collective works, not just the portions they themselves write, regardless of this limitation to *noncommercial* licensors.

The GPL also gives licensors the option to distribute source code through the Internet, although it acknowledges that licensees *are not* compelled (i.e., they *cannot be* compelled) to accept the source code if they don't want it:

> *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. (GPL section 3.)*

This provision relates only to software that is downloaded, and is needed only because items a-c) of GPL section 3 relate to distribution on a physical medium. Almost all open source software is now distributed electronically on the Internet. A more modern open source license would probably condense these complex source code rules in the GPL into a few brief sentences to require that the licensor make source code available online.

The GPL is thus consistent with the source code requirements of the Open Source Principles listed in Chapter 1.

## "At No Charge"

There are three words in the GPL's reciprocity provision that I saved until now. Here's how the GPL reads:

*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. (GPL section 2.)*

The GPL, unlike most other licenses, requires that derivative works be licensed as a whole *at no charge* to third parties under the terms of this License.

The open source principles listed in Chapter 1 allow reciprocity conditions, under which a licensor can insist that licensees operate on the exact same playing field as the licensor does. The GPL licensors distributed their software for free, and they insist that their licensee's derivative works also be *zero price, as a reciprocal condition for being allowed to create derivative works in the first place*.

Earlier in the license, however, the GPL left a big escape hatch for those who want to recover their costs of distributing software. It provides:

> *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. (GPL section 1.)*

Anyone familiar with business accounting will recognize that it is relatively simple to allocate costs to the "physical act of transferring a copy" when it is really a cost of getting copies ready to be transferred.

Regardless of this loophole, the laws of economics dictate that customers will only pay for value received. If they are free to make copies without paying anyone royalties for those copies, then the price that distributors can charge for copies of original software or derivative works will soon approach the marginal cost of production and distribution regardless of whether the GPL mandates a zero price.

There is also a problem that may prevent enforcement of the GPL's *at no charge* provision. It may be an illegal restraint of trade in some countries. Ordinarily, companies are allowed to set their own prices, and it is improper for a GPL licensor to constrain that in any way.

Most other reciprocal licenses do not require that derivative works be distributed at zero price. Their reciprocity obligation extends simply to requiring that the source code be published and that derivative works be distributed *under the terms of this License*. The price of the derivative work software is left for market forces to determine.

## Other Obligations in the GPL

The GPL doesn't grant unconditional licenses. Those who copy and distribute verbatim copies of a Program are required to:

> *…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. (GPL section 1.)*

Those who create and distribute derivative works are also required to:

> *a) Cause the modified files to carry prominent notices stating that you changed the files and the date of any change. (GPL section 2)*
>
> *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.) (GPL section 2.)*

These provisions protect the integrity and reputation of the original authors and ensure that subsequent licensees know that the GPL applies to that software.

## The GPL and Patents

Nobody is quite sure what effect software and business method patents will have on open source software. That is because patent problems often arise from unexpected quarters. A person nobody heard of may claim that software infringes his or her patent. Suddenly software embodying that patent cannot be made, used, or sold absent a license from the patent owner—unless, of course, the patent can be designed around and similar functionality accomplished in a different way.

It may thus happen that open source software that was previously free is no longer so. But that conclusion is here just a vague abstraction. Which software and which patent, and what effect on software freedom, is a mystery until it actually happens.

The GPL deals with such potential patent claims in a philosophically consistent way. If and when a valid patent claim by a third party prevents a GPL licensor from making, using, or selling the software, such software will no longer be *free* (in the GPL's sense of that word) and the software can no longer be distributed under the GPL. Here is the provision:

> *If, as a consequence of a court judgment or allegation of patent infringement..., 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.... (GPL section 7.)*

This leaves undefined just what "pertinent obligations" one might incur as a "consequence of court judgment," and leaves to later analysis what "obligations under this License" might be contradicted by the court judgment. The provision clearly means, though, that it will take more than the threat of patent infringement to invoke this provision. An actual patent dispute has to be alleged and either litigated or settled.

At the end of section 7, the GPL describes this patent provision not as a new provision but as "a consequence of the rest of this License." And so it is important to ask whether this form of self-imposed restriction on licensing in the face of a third party patent claim is an inevitable consequence of open source licensing in general or something unique to the GPL. It is particularly instructive that only the GPL has this provision, and that many other important open source licenses have very different patent defense provisions that don't require subsequent licensees to forgo their rights to create and distribute derivative works.

The only obligations a licensee accepts under the GPL are (1) the reciprocity obligation and (2) obligations regarding the integrity of the original authors. It is difficult to see how a court judgment regarding a patent would prevent either of these two obligations from continuing to be met.

I will describe in more detail in a later chapter on open source litigation that there are really only two significant consequences of civil litigation about a software license: an *injunction* or an *economic penalty*. As to injunction, whatever the injunction you must obey it; if a court orders you to stop making, using, or selling a patented invention, you must do so. As to economic penalties, if a court orders you to start paying royalties or to pay royalties for past infringement, you must do so. (You agree to accept those risks; the GPL's warranty provision, similar to those in most other open source licenses, provides no warranty of noninfringement.) The risk from patent infringement is the same whether you use the GPL, any other open source license, or indeed any proprietary license.

If a court order requires that you stop distributing derivative works unless you pay a license fee to a patent holder, you may elect to stop distributing derivative works or to pay the fee. The *at no charge* language of the GPL's reciprocity provision may prevent you from recovering that cost, but by itself it doesn't prevent you from continuing distribution if you're willing to do so at your own cost.

Patents are a local problem. Patents are awarded nationally; what is patented in one country may be free to use in other countries. The GPL acknowledges this by allowing licensors to continue to license their works in the geographical regions where the patents don't apply:

> *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.... (GPL section 8.)*

## Accepting the GPL

Under contract law, a contract is not properly formed unless the parties to the contract manifest their assent to being bound by it. Such assent is traditionally manifested by signatures on a license agreement, a technique that is not appropriate for mass-marketed software distributed at retail stores or over the Internet.

For software downloaded from the Internet, distributors generally require a *click-wrap* form of assent. Before they can download the software, prospective licensees are presented with the license and are given a chance to "Click to Agree." Only those who manifest their assent by clicking are allowed to download the software. Courts have also blessed this procedure under contract law.

For retail store purchases, distributors often use *shrink-wrap* agreements. A license is placed in the package along with the software. By careful packaging (usually in a shrink-wrap plastic), licensors can ensure that prospective licensees have an opportunity to review the license agreement before they gain access to the copy of the software they have purchased. By proceeding to open the software, licensees are presumed to have seen and agreed to the license. Opening the inner software package is deemed to be an appropriate manifestation of assent. Licensees who don't assent to the license have the opportunity to return the copy of software, unopened, for a full refund of their purchase price. Courts have generally blessed this procedure as satisfying the *manifestation of assent* requirements of contract law.

Some software implements a *click-wrap* procedure that occurs as the copy of software is actually installed on a computer rather than when it is purchased. Regardless of when the assent is requested, any purchaser of a copy who does not assent must be given an opportunity to return the copy for a full refund.

Courts don't generally care whether prospective licensees actually read the license agreements as long as there is a reasonable opportunity to do so, and as long as their intent to assent is manifested.

Of course, since most consumers don't actually read license agreements, and since most license agreements are complicated legal documents with largely unintelligible legal language, courts will also protect consumers from being surprised by unfair or unexpected provisions, even if they have manifested their assent. For now, I assume that most people reading this book accept that open source licenses—and the Open Source Principles upon which they are based—are fair.

The GPL relies on an entirely different set of legal principles, based on copyright law rather than contract law, to ensure that the license terms are accepted. It does not require—indeed its authors seek to prevent attempts to obtain—a manifestation of assent to GPL license terms. The GPL license acceptance provision reads as follows:

> *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. (GPL section 5.)*

Copyright law says that an author has exclusive rights to make copies of, to modify, or to distribute copyrighted software. Nobody can make a copy without a license from the author to do so. The mere exercise of someone else's exclusive rights without a license is an illegal *copyright infringement*. It is not necessary to prove that a defendant intended to infringe.

If you modify and distribute software without a license, the GPL suggests, you are presumed to know that your actions are illegal and, even if you don't know that you're breaking the law, the copyright law still makes you a copyright infringer. Don't do it, or you will expose yourself to potentially substantial penalties under the copyright law. (Possible penalties for copyright infringement include injunctions, the impounding and destruction of infringing articles, actual damages and profits, statutory damages, costs, and attorneys' fees.)

While this GPL reliance entirely on copyright law for license enforcement is legally sound, it has two shortcomings.

1. Only a copyright owner, not a distributor under a nonexclusive license, has standing to sue to enforce the GPL copyright license. On the other hand, if a contract is formed through a manifestation of assent, then contract law allows the distributor to enforce a license even if he or she doesn't own the copyrights in the underlying works. This means that if you use the GPL to distribute software but you don't own the copyrights to parts of that software, you can't sue under copyright law to protect those parts from infringement even if they were copied from *your* distribution. If you can prove that the licensee assented to a contract, however, you can protect your version of the entire work, and its component parts, against license violations.
2. At least in the United States, copyright disputes are heard only in federal court. Contract claims, on the other hand, can be heard in state and local courts, or in federal court if the amount in dispute is large enough and if the parties are not in the same state. If you use the GPL, you are limiting your litigation options to federal court.

If you are an open source licensor, I encourage you to obtain a proper manifestation of assent to your open source software licenses so that your enforcement options match your business strategies. If you want the option to pursue contract litigation and obtain contract law remedies, you probably don't want to use the GPL.

All open source licenses rely, at heart, upon the copyright law, as the GPL says in its section 5. But then, once a license is granted, that license may be interpreted under contract law provisions. Open source licenses should be clean, well-written contracts, or they may not be enforced by the courts.

This is the direction taken by all the licenses in the rest of this book. The GPL is the only license whose authors insist that it be treated as a bare copyright license but not a contract.