CHAPTER 6

Legal Impacts of Open Source and Free Software Licensing

All of the discussions in earlier chapters have assumed that each of these licenses can be and will be enforced by their licensors, and, ultimately, by the courts. However, two unique problems (in addition to those involved in the enforcement of any contract) affect licensors of software under open source and free software licenses.

First, for each license described in previous chapters, the licensor may not even know who the licensees are. All of these licenses, to varying degrees, put forth the licensed code with an invitation to adopt it and use it, subject to the terms of the respective licenses. These open source and free software licenses do not require notification or other affirmative action to be taken by licensees that would notify the licensor of the fact that the licensee has entered into the contract.* In addition, most of these licenses permit and even encourage the free sublicensing of the licensor's work to other licensees, whose connection to the original licensee can become tenuous as the licensed work moves through multiple generations of licensing before ending up with a particular user.

Second, while some of these licenses require that the licensee engage in some affirmative action to access the licensed work (such as clicking on a button indicating that the licensee agrees to be bound by the terms of the license) prior to permitting access of the licensed work, many of them—like the BSD, MIT, and Apache Licenses—do not. Others, like the GPL and LGPL, do not require such affirmative assent in all cases.

Both of these problems are substantially addressed by the fact that use of the licensed work is contingent on accepting the terms of the license. Unlike other types of contracts, open source and free software contracts impose very few, if any, affirmative obligations (such as the payment of royalties) on licensees, but rather impose restrictions only on the rights granted by the license. This property will operate, most likely, to save the enforceability of these licenses from challenges regarding the absence of mutual consent or consideration that may otherwise arise.

* The SCSL, which is not an open source or free software license, although it incorporates some of their principles, does require some form of notification. The Microsoft Shared Source Initiative operates under totally different custom-negotiation principles, so they know who they are dealing with from the outset.

Entering Contracts

Any contract between two or more persons rests on two fundamental assumptions: one, that there is some mutual obligation created by the agreement, which is known as the *consideration*; and two, that there is mutual consent, or a meeting of the minds, as to the terms of the contract, usually described as the *offer* and the *acceptance*. Once an offer that involves the exchange of consideration has been made and accepted, an enforceable contract is created. This principle is, of course, subject to numerous exceptions.

These concepts are capable of any number of variations and any number of hard cases involing these variations provide the subject matter for first-year law students. Basic principles suffice for our purposes. The idea of consideration turns on the fact that each party is undertaking an obligation, even a very minor one, to the other as part of the transaction. If Robert promises to give Sidney $10,000 in one year, and Sidney does nothing and agrees to do nothing, there is no contract, but only a promised gift. The significance of this is that such a promise is not legally enforceable. If Robert does not pay, Sidney cannot legally compel him to pay. However, if Robert agrees to pay Sidney $10,000 in one year if Sidney forbears from drinking alcohol for that entire time, that creates an enforceable promise: if Sidney fulfills her half of the bargain, she can legally compel Robert to live up to his, even though the consideration (abstinence from alcohol) that she promised (and performed) has at most only a very tangential benefit to Robert.

Even the most unrestrictive open source license imposes at least a minimal obligation ensuring that consideration in the legal sense is exchanged and an enforceable contract is created through the license. The MIT License, described in Chapter 2, imposes the following restriction on licensees:

> The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

While this obligation is not onerous, it is real, and failure to abide by it constitutes a breach of the contract. By extension, the more onerous restrictions imposed by the GPL, the BSD, the Apache, and all of the other open source and free software licenses already described impose sufficient obligations so as not to fail as contracts for lack of consideration. The licensor grants a real benefit, the right to use the licensed software, and the licensee agrees to genuine restrictions, i.e., those that are expressed in the license.

Potentially more problematic is the question of mutual consent. In an ordinary commercial contract, this question rarely, if ever, arises. In general, mutual consent can be attacked only in relatively unusual circumstances. In the classic formulation of a contract, the two parties to the contract have met, negotiated, and reached final agreement, embodied in a formal, signed document. Under those circumstances, the consent of either of the parties can be attacked, essentially in only two ways. First, one of the parties can argue that his consent was induced by *fraud*, i.e., that the other party deceived him as to a fact material to the contract. For example, two parties may agree to a contract that provides for the sale of a document signed by Elvis Presley. The genuineness of the signature is critical to the contract. If the buyer can prove that the signature was a forgery and that the seller knew it, he can void the contract—render it of no legal effect—on the grounds of fraud. Second, mutual consent can be attacked on the basis of *incompetence*. In most jurisdictions, a person under the age of 18 cannot enter into a binding contract. Accordingly, if such a person enters into a contract, she can sue to have the contract voided on the basis that she was incompetent to enter into the contract in the first place.

While these circumstances appear in numerous variations and can present difficulties in interpreting contracts and adjudicating disputes that arise from them, they are relatively clear cut assaults on the mutual consent to a contract. However, because of the absence of a writing signed by both parties formally indicating their agreement to a contract, the open source and free software licenses described earlier present a different, and more complex problem.

It has long been accepted that contracts may be formed in the absence of a signed document. Oral contracts, with significant exceptions, are regularly enforced. The familiar "shrinkwrap" license that frequently governs the use of commercial software is more applicable to software contracts. The user purchases the software; the box in which the media containing the software is sold indicates that use of the software is governed by a license; and the purchaser is further informed that breaking the shrinkwrap and opening the box indicates the user's consent to the license agreement. Some courts have upheld the creation of a contract under these terms; other courts have not. A potentially critical distinction, described in more detail later, is the extent to which the purchaser was aware (or could have made himself aware) that the software was provided subject to a license and could have learned the terms of the license that would govern the use of the software.

These questions become more difficult when the product and the license both exist in a virtual space and the offer and acceptance both take place there. There are a number of different contexts in which this kind of offer and acceptance can take place, and small differences can be critical in determining whether a contract is formed. For the following examples, a web site is posited as the locus of the contract, although the same issues could arise as easily with software recorded on a physical medium, such as a CD-ROM.[*]

In the first example, an icon appears on the introductory screen for a piece of software, indicating that that software is being provided subject to the terms of a license.

[*] Readers interested in a more detailed legal analysis should read the opinion of Judge Alvin K. Hellerstein in *Specht v. Netscape Comm. Corp.*, 00 Civ. 4871 (AKS), 2001 WL 755396 (S.D.N.Y. July 5, 2001).Such contracts arise outside the world of software licensing as well. Ticket stubs—such as those received at coatchecks or parking garages—which typically disclaim any liability for checked items, present similar issues.

A user who wants to view the terms of the license can click on a hyperlink that takes him to a page displaying the terms of the license. Another hyperlink links to the site from which the software can be downloaded. This “browsewrap” license may create an enforceable contract: the user (or purchaser) is at least made aware that the software is produced subject to a license, but he is not required to assent to the terms of the license, or even to look at it, before accessing the licensed work. The enforceability of this kind of contract is, however, subject to dispute and this arrangement may not result in a contract that would be enforced.

The second example, the so-called “clickwrap” license, is more likely to create an enforceable contract. In this variation, the user is required to view, however fleetingly, the terms of the license and to take some affirmative action to agree to its terms, such as by clicking a button that says “Yes, I have read this license and I agree to its terms,” before accessing the licensed software. This is the form of license contemplated in some of the licenses described earlier and will generally provide sufficient notice to the user of the terms of the license and require sufficient affirmative action to create an enforceable contract, so long as the other requirements of contract are met, such as the competence of the parties and the absence of fraud.

A variant of the “clickwrap” and “browsewrap” licenses, in which the user only views the license and is not required to take any affirmative action indicating consent to the licensed terms, but where consent is implied from some other action (usually the downloading of the licensed software), may or may not be sufficient to create an enforceable contract. The licensee knows of the license, knows it governs use of the software, and has the opportunity to review it before accessing the software. Nonetheless, the absence of affirmative consent (such as clicking on a text box as required by the “clickwrap” license) is troubling to courts, and correctly so. It seems unfair to enforce terms of a contract to which one of the parties has done nothing to positively affirm.

This issue has obvious application to the open source and free software licenses already discussed. Staying with the MIT License, say, for example, that an ordinary user comes across a piece of code that is subject to this license. The user takes the code and uses it on his personal computer. The user incorporates the code into a program that he is writing. The user distributes the program, either for profit or not. At no point has the user taken any affirmative, symbolic action that would indicate his consent to the terms of the license that is comparable to the act of signing a contract.

Statutory Developments Related to Software Contracts

The Uniform Electronic Transactions Act (UETA), a model law adopted by at least 22 states and under consideration in others, provides as a general matter that a contract may not be denied legal effect simply because the contract is recorded in an electronic medium and not on paper.

E-Sign, a federal law passed on October 1, 2000, operates to a similar effect, in holding that digital signatures on documents are as effective as ordinary written signatures on paper in memorializing an agreement.

Neither UETA nor E-Sign purports to alter ordinary state law governing interpretation of contracts.

Another model law, the Uniform Computer Information Transaction Act (UCITA), does modify ordinary state contract laws relating to transactions in software. Although it is intended to facilitate transactions in information and provide for uniform interpretation of contracts governing such transactions, the UCITA has not been widely adopted. Only two states, Maryland and Virginia, have adopted UCITA; a number of states, however, have adopted anti-UCITA statutes. Because UCITA's effect is currently very limited and does not seem likely to spread in the near future, it is not further addressed here.

The Self-Enforcing Nature of Open Source and Free Software Licenses

There is a "savings" logic present in the MIT License (and others) that preserves the effect of the license even in the absence of an affirmative act of consent. This is because open source and free software licenses do not impose affirmative obligations on licensees but rather impose restrictions on the rights granted under the license: such restrictions can be relatively straightforward, as is the case with the MIT License's requirement of reprinting the copyright and permission notice; or somewhat more complex, as with the far-reaching consequences of licensing under the GPL License.[*]

The GPL License provides a good example of this phenomenon. The typical limitations of proprietary licenses simply do not apply to most applications of GPL-licensed software. For example, installing, using, or even modifying GPL-licensed software implicates no term of that license. Any user is completely free to undertake any of these actions. There are no limitations on the number of installations of the software that a user may undertake and no requirement that the user pay royalties in exchange for use, in sharp contrast to proprietary licenses. Only if the user intends to distribute the original code or modified versions of it does the GPL come into effect.

It is only at this point (and the same is true of the other open source and free licenses already discussed) that questions of enforcement even arise. And it is at this point that the unique strength of these licenses becomes apparent. As already discussed, in the absence of a license, the user would not have even the right to maintain, use, or modify the copyrighted code. Even work that is not specifically identified as being

[*] This section's discussion draws heavily on the essay by Eben Moglen, "Enforcing the GNU GPL" located at *http://www.gnu.org/philosophy/enforcing-gpl.html*.

copyright is protected under the law of the United States and other nations. The user considering challenging the applicability of the license is thus faced with a real dilemma.

On the one hand, the user is free to disclaim the obligations of the license, most likely on the grounds that he never affirmatively agreed to be bound by the license. If he does so, he is not obligated to pay royalties or otherwise conform to any affirmative agreements that the license might require. However, by disclaiming the license—taking the position that no enforceable contract exists between him and the licensor—the user is arguing that the "default" state of copyright exists: that state of protection which applies to any copyrighted work not in the public domain. While free of any restrictions that may derive from the license at issue, such a user finds himself in the unenviable position of lacking all of the fundamental rights granted by the open source or free software license that he wishes to exercise. A user in such a "default" copyright state is barred from distributing or modifying the work (except to the limited extent permitted by fair use), without the permission of the copyright holder, which permission, by disclaiming the license, he has already refused.

If, however, the user wishes to exercise rights under the license, he is compelled to accept with it whatever limitations or restrictions may be contained in the applicable software license. For example, under the GPL, if a user wishes to incorporate GPL-licensed code into his own programs, he is required to license those programs under the GPL and thereby permit the "free" use of them as described in the GPL. As a legal (and a common sense) matter, he may not pick and choose, so as to accept the benefits of the license without its restrictions.

Unlike people who may object to the onerous obligations that could be imposed by "shrinkwrap," "clickwrap," and "browsewrap" licenses (such as, for example, the obligations of paying royalties) and who would disclaim the contract entirely and forego the use of the licensed software if given the choice, users of open source and free software licensed software cannot realistically "walk away." The continued availability of the work that they want to use is contingent on their adherence to the license's terms. While they are free to "walk away," the condition on the abandonment of the restrictions of the license is the surrender of the rights granted by the license.

This feature makes open source and free software licenses remarkably easy to enforce. A licensor can simply tell infringers that infringement vacates their continued rights to the licensed code. As most infringers are aware of the substantial civil and criminal penalties associated with copyright infringement, and desire the rights granted by the license, they will make their behavior conform to the demands of the license. For those infringers unwilling to conform to the terms of the license, even after being put on notice of the license, and who continue to infringe (typically by redistributing the licensed work under an incompatible license, such as a proprietary license), the licensor can directly contact the customers of the illegally licensed software. The original licensor can inform those customers that the same (or substantially similar software) is available under the terms of the original license, which are almost certainly more favorable to that customer. In addition, because the customer is aware of the difficulties and expense associated with relying on software licensed under what is, at best, a highly questionable license, it is probably sufficient to convince such customers to abandon the use of the work distributed in violation of the license. While this involves some degree of administrative and legal sophistication on the part of the licensor, this is generally not a great burden. The Free Software Foundation has policed the GPL License in exactly this fashion for many years with consistent success.

The Global Scope of Open Source and Free Software Licensing

Another issue for open source and free software licenses is their enforcement in jurisdictions outside the United States. The global nature of commerce and the generally free travel of software across national boundaries implicates the enforcement of open source and free software licenses in a number of jurisdictions, not only those in the United States.

International enforcement of copyright laws is frequently lax. While many countries are signatories of treaties that provide for the international enforcement of copyright protection (such as the Berne Convention), such treaties are frequently disregarded. The proliferation of "pirated" DVDs and CDs is a testament to that. The use of file-sharing software frustrates enforcement of copyright even within the United States. Within such a framework, it may seem impossible to enforce the terms of open source and free software licenses, which depend, as just noted, on the foundations of copyright for enforcement across national boundaries.

In many countries, particularly in the "developed" world where most software creation takes place, the enforcement of copyright is routine. While the unauthorized distribution of copyrighted material is commonplace, it is nonetheless difficult for any established company or person to reasonably hope to profit from the illegal distribution of copyrighted material. This is particularly true of software. Users of software, at least commercial users, are generally more concerned with reliable performance and support than with the incremental cost of software. Users expect to be able to rely on a software maker's products and to receive support for that software's application going forward. Providing this reliability and these services requires the existence of a stable, aboveground organization—exactly the kind of organization that is subject to suit and accordingly to the legal enforcement of copyright law.

The question thus becomes whether open source and free software licenses can reasonably be expected to be enforced, as a legal matter, outside the United States. The answer to this question is a slightly qualified yes. Many countries are signatories to the Berne Convention, which provides for copyright protection more stringent in many respects than that provided by United States copyright law. Moreover, as has been the case with the enforcement of proprietary licenses, the existence of some amount of "pirating" or distribution outside the boundaries of a given license, such as with the unauthorized distribution of music, is not fatal to the successful distribution of the licensed work. Even if a certain, substantial, percentage of distribution of work is through illegal channels, the machinery distributing that work is still capable of thriving, creating, modifying, and delivering new work.

This is likely to be particularly the case with open source and free software licensed work, for the reasons already discussed. "Pirating" work generally means nothing more than the distribution of the work itself without the payment of royalties (or other applicable forms of payment) to the creator of the work. "Pirating," in this sense, thus does not violate the restrictions applicable to most open source and free software licenses, which generally do not limit the free (i.e., without charge) distribution of unmodified versions. Only the distribution of modified work in a way inconsistent with the terms of the applicable license really "counts" as a violation of the license.

"Pirating" in this sense is also limited by the fact that the major markets in which software or any other kind of work can be sold at a profit are subject to legal constraints and the enforcement of law. In addition, practical constraints are more likely to limit the extent of such piracy with regards to software than with regards to other forms of expression, such as CDs or DVDs. While a consumer may be willing to take a chance on a five dollar bootleg CD or DVD that she intends to use just for her personal entertainment, such a consumer is much less likely to take such a chance on software, the stability and functionality of which she really must rely on.

These dynamics probably explain the relatively small amount of litigation spawned by open source and free software licenses. While these licenses certainly can be (and are) infringed upon, market forces and social dynamics tend to limit the extent of such infringement, even in the absence of vigorous legal enforcement of the license by the licensor.

The "Negative Effects" of Open Source and Free Software Licensing

Another effect of open source and free software licensing that has already been touched upon is the obstacle that violations of applicable licenses create for the violator of that license. Such violators will find that their own ability to enforce copyrights that arise out of or are related to infringements of the terms of an open source or free software license is seriously compromised. Violations of such licenses put the violators at risk of surrendering the benefits of any actual, copyrightable work that they may have invested in modifying or improving a licensed program.

Taking again one of the least restrictive examples of open source licenses as an example, it becomes apparent that violation of its terms undermines any future copyright enforcement relating to the modified work. The MIT License, described in Chapter 2, imposes the following restriction on licensees:

> The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

This example is equally applicable to the other open source and free software licenses already described in this book, although, obviously, what constitutes such a violation of the license will vary.

Assume that XYZ Corporation develops software based on a program called Duchess, licensed under a license with terms identical to the MIT license. XYZ incorporates large amounts of the Duchess code into its own program, called Vulcan, which is a use clearly permitted by the Duchess license. For purposes of promotion, however, XYZ decides that it would be better served in marketing Vulcan without acknowledging the efforts of the creators of Duchess and launches Vulcan into the market under a proprietary license without including the required copyright and permission notices. After all, XYZ reasons, Vulcan will be released under a proprietary license, without giving anyone else access to Vulcan's source code. The operations and appearance of Vulcan are sufficiently distinct from those of Duchess that it is not apparent that Vulcan is based on Duchess and the functions it performs are dissimilar to those of Duchess. At the time of the software's launch, it seems remote at best that it will ever come to light that XYZ has infringed upon the Duchess copyright by ignoring the MIT License's requirement that the copyright and permission notice be included in Vulcan.

Years pass, and XYZ prospers thanks to sales of Vulcan. One day, another software company, ABC Corp., brings to market a new program, Virgo, that fulfills the same functions as Vulcan but at a lower price. This Virgo software is also based on the Duchess code, but it complies with the Duchess license's requirement that it provide the copyright and permission notice. Virgo, however, has several features that mirror those in Vulcan—strongly suggesting to XYZ that a substantial portion of Virgo's code was taken directly from Vulcan. Moreover, approximately a year before Virgo's release, ABC had hired several of XYZ's programmers who had access to Vulcan's source code.

XYZ now hires lawyers and seriously considers bringing a copyright infringement suit against ABC for infringing its copyright to Vulcan. Seeing Vulcan's market share erode rapidly to Virgo, XYZ begins drafting a complaint against ABC, the first step in initiating litigation. But in the midst of this process, XYZ's lawyers discover XYZ's failure to comply with the Duchess license. They advise XYZ not to bring the lawsuit.

XYZ asks why. The answer is simple. By failing to comply with the requirements of the Duchess license, XYZ has seriously compromised its ability to enforce its copyright to those portions of Vulcan that really are XYZ's own work. Moreover, upon the discovery of XYZ's violation of the Duchess license, Duchess's creators, could sue XYZ for infringement, and one of the potential measures of damages in such a case would be all, or a substantial portion, of the profits that XYZ had realized through sales of Vulcan.

The first result, the compromising of XYZ's ability to enforce its own copyright claims, comes from the equitable doctrine of *unclean hands*. This doctrine holds that a party seeking relief from a court should have engaged in the transaction from which the lawsuit derives fairly and equitably. Following this doctrine, federal courts have held that a copyright claim can be defeated if that copyright was obtained unfairly or inequitably.* While not a foregone conclusion, if XYZ brought such an infringement suit, it would almost certainly be discovered that XYZ itself had infringed on the Duchess copyright by distributing Vulcan without complying with the license. This could result in the invalidation of XYZ's copyright to Vulcan. Having lost the copyright, XYZ would lose its exclusive right to distribute Vulcan.

The second result, following naturally from the first, is that upon the disclosure of XYZ's violation of the Duchess license, Duchess's creators could sue XYZ for infringing the Duchess copyright. Having disregarded the terms of the license, XYZ is in the same position as any other infringer. One possible remedy for such a violation is a measure of damages called *unjust enrichment*. This measure would award in damages those profits that could reasonably be said to flow from XYZ's infringement of the Duchess copyright; a measure that could result in XYZ having to pay over a substantial portion of the profits it earned since it had begun to distribute Vulcan. Again, while such a result is not a foregone conclusion, it is an outcome that XYZ would have to consider in deciding whether to bring a lawsuit.

Given the reasonable possibility that one or both of these results would flow from the lawsuit, either of which would be sufficient to put XYZ out of business, and given the uncertainty involved in bringing a copyright infringement action under even the best of circumstances, the lawyers see no alternative to foregoing the lawsuit. XYZ simply must compete in the marketplace the best it can with the potentially infringing Virgo program. XYZ's lawyers would also likely recommend that XYZ quietly add the permission and copyright notices required by the MIT license to avoid future infringement.

Thus, the failure to comply with the Duchess license, while providing potentially significant short-term benefits to XYZ, ultimately threatened the viability of XYZ's ability to continue an ongoing operation. While such license violations may never be directly discovered, they significantly compromise, as just described, the violator's ability to enforce its own copyright, with potentially dire consequences.

* Wrongful action taken in securing a copyright can invalidate that copyright. See *Lasercomb Am., Inc. v. Reynolds*, 911 F.2d 970, 977-79 (4th Cir. 1990). In at least one case, Ashton-Tate had sued Fox Software and the Santa Cruz Operation, alleging that the defendants had infringed upon its dBase line of programs with the sale of their competing FoxBase software, a federal court found that Ashton-Tate had obtained its own copyright deceptively by failing to inform the Copyright Office that its own software was based in significant part on JPLDIS, a public domain program. As a result, the court voided Ashton-Tate's copyright and dismissed the suit. While the court soon reversed itself, the potential for such a severe sanction is real.

The consequences that flow therefrom can be even more serious depending on the license being violated. Under a "copyleft" license like the GPL, a company like XYZ would be in an even more tenuous position. As described in Chapter 3, "copyleft" is a variety of the generational limitation described in Chapter 1, which requires that derivative works be subject to the terms of the GPL and only the terms of the GPL. This requirement is embodied in Section 2(b) of the GPL.

> 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:
>
> [. . .]
>
> 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.

The GPL explicitly provides that failure to comply with the terms of the license voids any rights granted by the license.

> 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.

Assuming that the Duchess program from the previous example were licensed under the GPL instead of the MIT License, these provisions of the GPL License would place a company in the position of XYZ in an even more precarious position. If XYZ takes the position that it is not bound by the GPL License, it has no right to incorporate code derived from it in its own program, Vulcan. If XYZ takes the position that it is bound by the GPL License, it must cease distributing Vulcan under anything but the GPL License and must also concede that its previous distributions under a non-compliant license constituted an infringement of the Duchess copyright. In such a scenario, XYZ is in an even worse position that it would be in the MIT scenario. Unlike the MIT License, there is no "quiet" way for XYZ to ensure compliance with the terms of the Duchess license in the future. XYZ's lawyers are in the difficult position, once the infringement has come to light, of informing XYZ that it must either cease distribution of Vulcan or immediately release it under the GPL (and only the GPL) License. Because criminal as well as civil penalties attach to copyright infringement, the continued distribution of Vulcan under a proprietary license could potentially involve XYZ's lawyers in XYZ's own wrongdoing, a result most lawyers seek to avoid.[*]

[*] XYZ's lawyers are not obligated to inform anyone of XYZ's wrongdoing and in most jurisdictions would be barred, by the attorney-client privilege, from doing so. However, continuing to aid an ongoing criminal violation is both unethical and dangerous.

While only examples using the MIT and GPL Licenses are described, similar results would follow from distributions of licensed software inconsistent with the terms of the applicable open source or free software license.

Community Enforcement of Open Source and Free Software Licenses

The open source and free software communities are also critical to the practical enforcement of open source and free software licenses. While the discussion so far has focused on the legal and practical reasons why open source and free software licenses tend to be complied with, there is a more fundamental reason why most programmers comply with such licenses. Non-compliance, or at least knowing non-compliance with the terms of these licenses, is simply wrong.

The world of open source and free software licensing is still a relatively small one. As has already been described in previous chapters, the code written under these licenses is mostly the work of volunteers who have dedicated huge amounts of time, and, in many cases, significant parts of their lives to the development and distribution of good code for the benefit of as many people as possible. In the course of writing this code and supporting these projects, these programmers have foregone significantly more lucrative opportunities offered by commercial software companies. Behind the black and white terms and restrictions of these licenses, which have taken up the bulk of this book, is a real principle. Free code, however free may be defined, is a social good in itself. This is the goal that is being pursued. However that goal may be reached, whatever avenue of development is followed, this principle is held above all others.

This principle is deeply felt by this community. The gross violation of it by taking someone else's work and distributing it as one's own is unthinkable. This moral principle is, by itself, responsible for the largest part for the enforcement of open source and free software licenses, not the texts of the licenses themselves, and not the courts that enforce those licenses.*

Even those who have not internalized this principle have good reason to abide by the norms of this community. Violating those norms will incur, at the least, the displeasure of this community. Given the number of people in this community and, perhaps more importantly, the knowledge and capabilities of its members, such a violation can result in the ostracism of the violator. Such a person might find his emails remaining unanswered, being ignored or flamed in usegroups, and being

* For more discussion of this principle, see the essay *Homesteading the Noosphere* in *The Cathedral & The Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*, Eric S. Raymond (O'Reilly 2001) (rev. ed.), and the chapter *The Art of Code* in *rebel code: inside linux and the open source revolution*, Glyn Moody (Perseus Publishing 2001).

excluded from projects, whether under the open source or free software banner, that involve members of this community.[*]

This does not mean there is a univerally shared view as to the purpose of open source and free software licensing or the best way to realize that purpose. As noted earlier, there are real ideological differences between, for example, the “open source” community and the “free software” community. That said, there is considerable common ground. One principle, which is universally accepted, is that taking someone else’s work and modifying or distributing it in disregard of the intent of its creator is wrong.

This should not be confused with the “cross-over” of programmers (and their code) from an open source project to a proprietarily licensed projects. As described at the end of Chapter 2, prominent open source programmers such as Bill Joy and Eric Allman moved from open source to proprietary projects. In Allman’s case, he maintained both open source and proprietary distributions of his popular Sendmail program in a way consistent with both the terms and the principles of the original license. Such movement does not (and should not) result in any ill feeling against such individuals.

In sum, while contracts and courts are fundamental to protecting the principles of open source and free software licensing, the real guardians of these principles are programmers (and users) themselves.

Compatible and Incompatible Licensing: Multiple and Cross Licensing

In writing code, a programmer may find that he wants to fuse elements from two or more programs into a new program. The two programs are under different licenses. The question arises: is it possible to take this code, under different licenses, and combine them in one work without violating the terms of either of the two licenses?

While two licenses may appear to be compatible, programmers must ensure that they are, in fact, compatible. Apparently innocuous terms in one or both licenses may make them incompatible with each other. Distribution or modification of programs, including incompatibly licensed code, will result in copyright infringement.

Those undertaking this analysis should note that with some exceptions (such as the GPL and the SCSL Licenses) the licenses described in this book are frequently templates for individual licenses, and their language may not be exactly the same as that

[*] Those interested in the enforcement of social norms that parallel legal restrictions should read *Order Without Law: How Neighbors Settle Disputes*, Robert C. Ellickson (Harvard, 1991). While this book addresses primarily the enforcement of social norms among cattle ranchers in Shasta County, California, its analysis is no less applicable to “virtual” communities such as the open source and free software communities.

described here. Simply because something is described as a "BSD-style" license, for example, does not mean that it is written just like the BSD License or contains exactly the same terms. In every case, a user considering combining works licensed under different licenses should read the licenses at issue very carefully.

It is much easier to describe those licenses that are incompatible than to assert with any assurance that two licenses are compatible. There are several scenarios in which the answers are obvious. If either one of the works is licensed under a proprietary license, the code cannot be combined with work under another license (except through cross-licensing, described later). As a general matter, under a classic proprietary license, the user has no rights to the work other than to use a single copy of it. Ordinarily, she may not even examine the code, much less modify or distribute part of it as a section of another work. Two works under proprietary licenses, even if they are the same license, cannot be modified or distributed together without violating the license(s).

Another example susceptible to quick analysis is the GPL License. GPL-licensed code is incompatible with code licensed under most licenses. As noted in Chapter 3, the second sentence of Section 6 of the GPL reads as follows:

> You may not impose any further restrictions on the recipients' exercise of the rights granted herein.

By combining GPL-licensed code with code under any but the most unrestrictive licenses, the creator of the putative "new program" is imposing a restriction (compliance with the terms of that license) that is not present in the GPL License and which accordingly violates the GPL. (The LGPL is not restrictive but is compatible with the GPL by design.) As noted previously in this chapter, one of the fundamental building blocks of any contract—and all licenses are a form of a contract—is consideration, i.e., the imposition of some obligation on each party. In the open source and free software licenses discussed in this book, the typical transaction involves the licensor agreeing (becoming obligated) to permit certain uses of the licensed work in exchange for the licensee agreeing and becoming obligated to comply with certain restrictions. Accordingly, any license worthy of the name will impose some obligation: if that obligation does not precisely parallel an obligation in the GPL, that obligation is a "further restriction" as far as the GPL is concerned and cannot be imposed on licensees of the GPL code.*

* The LGPL operates in exactly the same way, excluding, obviously, the less restrictive limitations imposed on "hitchhiker" programs that use, and may be distributed with, the LGPL-licensed library.

For a list of licenses that the Free Software Foundation considers to be compatible with the GPL, see *www.gnu.org/philosophy/license-list.html*.

It should be noted that Section 2(b) and Section 6 of the GPL might be read to be so restrictive as to make the GPL incompatible even with those licenses described by the Free Software Foundation as compatible. In the event the GPL-licensed work is not copyrighted by the Free Software Foundation, a person interested in combining such a work with a work under a "compatible" license may wish to take additional precautions, such as contacting the copyright holder of the GPL-licensed work.

This brings us to another realm of quick analysis. Some works are not subject to any license at all. Their creator has either consigned them to the *public domain*—for example, by attaching a Public Domain Dedication to the work, as described in Chapter 4—or the work has lost its copyright protection and entered the public domain by the lapse of time.* Because such public domain works impose no obligations on their users, code that is in the public domain may be combined with code licensed under any license, so long as that license's terms are complied with. Accordingly, a programmer may incorporate public domain code with GPL licensed code, and modify and distribute the resulting work without fear, so long as he complies with the GPL.

After these three straightforward examples, the question of the mutual compatibility of licenses becomes substantially more complicated. In the following situations, programmers (or preferably their attorneys) should examine each of the applicable licenses cautiously.

In general, the "research style" licenses described in Chapter 2 are compatible with each other. Accordingly, a program licensed under the BSD License may be combined with a program licensed under the MIT License, and both released under the licensee's proprietary license so long as each license's restrictions are complied with. In this example, the BSD License would require that its list of conditions be included in the software distribution and that all advertising materials note that the software includes work made by the University of California at Berkeley; the MIT License would require that its copyright and permission notices be distributed with the resulting software, and the proprietary license governing the combined work would contain whatever additional restriction that licensee chose to impose.

"Research style" licenses are also generally compatible with licenses that do not bar the imposition of additional restrictions on the code to be licensed. For example, the Q Public License permits the distribution of modified forms of the licensed work in the form of the original work plus patches. If a user wishes to draw code for a patch from an MIT-licensed program, he is free to do so and to distribute that patch in a manner consistent with both licenses, so long as he complies with the other terms of the Q Public License and encloses the permission and copyright notices required by the MIT License.

Beyond these general observations, it is difficult, if not impossible, to provide precise guidance about what licenses may or may not be compatible with each other. As already noted, many licenses described in this book are really templates and are subject to significant variations in their terms in practice. Programmers who are considering combining code governed by two or more different licenses should proceed cautiously.

* As noted in Chapter 4, there are reasonable questions about the binding effect of public domain dedications, such as the one put forward by Creative Commons.

Fortunately, there is another solution, generally available, which is both easier and more reliable than comparing the arcane terms of two separate licenses. This is the phenomenon of cross-licensing.

As the creator of a work, the original licensor retains all of the rights associated with that copyright, subject only to the sale or licensing of those rights to others. The open source and free software licenses described in this book do not require that the creator of a work surrender all of his rights to another. Rather, in each case, the license reflects only a specific, one-time, grant of certain specified rights based on the compliance of the licensee with specified conditions. The licensor does not agree only to license the work under those terms or to those licensees.

Accordingly, such a licensor retains the power and the discretion to license his work under terms other than those contained in the original license. This is cross-licensing. ABC Corp. licenses its program, Mudd, under the pre-1999 BSD License. Several years later, John Smith wishes to incorporate some of the Mudd code into his ongoing free software project, the GPL-licensed Pond, which is based on code from an earlier GPL-licensed program, River, created by Audrey Strauss. Smith understands that the GPL and the pre-1999 BSD Licenses are incompatible. He can resolve this dilemma if either ABC or Strauss is willing to cross-license their programs—i.e., make the program (or a version of it) available under a license other than that which the program was originally provided under. In this case, Smith could go to ABC and ask them to license a version of Mudd under a GPL License so that he can use it in a GPL licensed new version of Pond. Smith could also go to Strauss and ask her to license a version of her River program under the pre-1999 BSD (or other compatible) license so that he can incorporate it in a BSD licensed version of Pond.

While it may seem somewhat presumptious to approach an author of a work, who likely has given at least some thought to the license applicable to the work, to reconsider that decision, open source and software programmers are generally open to the idea of cross-licensing. Given the ethic in the open source and free software communities to favor free distribution of work and to avoid duplication of effort, most programmers would be inclined to give such requests a favorable hearing, at the very least. This possibility is explicitly laid out in some licenses, including the GPL. Section 10 of the GPL, for example, provides as follows:

> 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.

If this option succeeds, all the difficulties and potential uncertainty associated with different licenses pass away. Whatever license is mutually agreeable to everyone involved will control and the coding can go on without fear of future legal problems.

Some open source groups will not cross-license works copyrighted by them. The Apache Software Foundation, for example, does not cross-license its works.

There are also some situations in which cross-licensing is simply not a practical alternative. Some project structures, such as the "bazaar" structure described in the next chapter, permit input into projects by hundreds and possibly thousands of programmers. The Linux (or GNU/Linux) operating system is the quintessential example. Linux is licensed under the GPL and includes the works of thousands of people who made contributions to the project with the belief (assuming that they took the time to develop one) that the resulting work would be licensed under the GPL. In such cases, there is no one person capable of relicensing the work of all these people under a different license, not even Linus Torvalds. Because of the rigidity of the monopoly granted by copyright laws, each one of those contributors could argue, legally, that their contribution can only be used in ways consistent with the terms upon which they agreed to participate in the project. Cross-licensing such projects, while not impossible, is impractical in all but the most unusual situations.

Most open source and free software projects, however, do not present such logistical difficulties. They are maintained by a small number of people, frequently just one person, whose permission to distribute that work under another license can often be gained for no more than the cost of asking.