5

Academic Licenses

The BSD Gift of Freedom

The first open source license, the original BSD (Berkeley Software Distribution), was designed to permit the free use, modification, and distribution of certain University of California software without any return obligation whatsoever on the part of licensees.

The term *academic freedom* usually means the freedom of a (tenured) professor to speak openly without risking his or her job. This presumably results in a dynamic and diverse community of thought that enriches everyone's academic experience and results in the exploration of new ideas. But that isn't the type of academic freedom that open source deals with.

Academic open source licenses promote a slightly different kind of freedom, relating to the mission of an academic institution to promote education and scholarship. Teachers are encouraged to publish their ideas rather than hide them under a cloak of secrecy. Students are expected to take what they learn and apply it to their own work, creating new ideas in turn. In pursuing this type of academic freedom, universities often forgo an immediate profit motive and instead consider

the bigger benefit to society of releasing their intellectual property to the public. Of course, not all universities practice this ideal all the time.

The University of California decided to use the BSD license to promote this latter type of academic freedom. It apparently concluded that some of its software would be more valuable if it were made freely available for all to copy, modify, and distribute than if the University were to keep it secret or to attempt to sell it privately.

Some suggest that the University could have accomplished this merely by waiving its copyright or dedicating its software to the public domain. Under the copyright law, though, there is no mechanism for waiving a copyright that merely *subsists*, and there is no accepted way to dedicate an *original work of authorship* to the public domain before the copyright term for that work expires. A license is the only recognized way to authorize others to undertake the authors' exclusive copyright rights.

In Chapter 4, I created a simple license to accomplish this goal:

> *Simple License: The copyright owner of this software hereby licenses it to you for any purpose whatsoever.*

(This isn't the BSD license. The grant in the BSD license is longer and more complex, but I'll get to that in a bit. I'm using this one-sentence Simple License for illustrative purposes only.)

The Simple License, if properly accepted, is a *unilateral contract* in which only the copyright owner has offered promises, in particular the promise to let you use the software as you see fit. The licensee has promised nothing but is nevertheless bound to the terms and conditions of the contract if he or she uses the software as licensed.

Such unilateral contracts are formed all the time in daily life, although we don't often think about their terms and conditions when we enter into them. That is because, for many commercial transactions, we leave it to the law to specify the implied terms of contracts that we enter. You can take comfort when you go to a store to buy a toaster that the store will return your money if the toaster is unsatisfactory, or will repair or replace the toaster if it doesn't work as advertised. The only way for a store to avoid its *implied* promises is to *expressly* disclaim them; it may sell you the toaster "AS IS" and "subject to all flaws."

Notice that you, the consumer, don't promise anything to the store in return. You can use the toaster however you want, or not use it at all. The store and you have a *contract* even though only the store has made promises—in this case implied ones—about efficacy and safety.

As with the purchase of a toaster, every other condition in the Simple License could be left to the legal defaults for software licenses—whatever those defaults are. Such a one-sentence license is fully compatible with the Open Source Principles, and in theory at least it could be approved by Open Source Initiative as a valid open source license.

What are the licensor's implied promises in this Simple License? The law prescribes that, at least in the case of a commercial or consumer transaction for *goods*, there are implied promises that the goods will perform as they were advertised to do. If software is *goods*, and if the software turns out to break computers or doesn't perform the way its documentation specifies, the licensor may be responsible to pay damages—even when the license is silent about it.

But software isn't *goods*. The law in many jurisdictions hasn't quite decided what it is. *Implied* promises for software contracts aren't well defined.

The University of California didn't want a dispute about whether software was *goods*. It merely wanted protection from implied promises, and it wanted to avoid having to pay damages if a user was injured in any way by the software. It protected itself with an *express* warranty disclaimer and an *express* liability disclaimer, about which more later. That added two sentences to what otherwise could have been a one-sentence license.

The University of California also wanted to impose a few *conditions* that would be required of every licensee. (Ignore for a moment the particulars of those conditions.) How can a *unilateral contract* impose conditions on licensees if all the promises are made by the licensor? The answer is that, under the law, a *condition* is not a *promise*. In the case of a unilateral contract like the BSD, the conditions must be satisfied by the licensee or it relieves the licensor of his promise to let you have the software.

The BSD license accomplished much more than simply giving a particular piece of software away. By encouraging the contribution of software into a public commons of software available to anyone, it created a growing benefit to the University of California and everyone else. As the theory goes, more and more people will contribute BSD-licensed software to the commons in response to, and as consideration for, earlier contributions. The huge amount of software now available under the BSD license (and similar academic licenses) has proven that this theory works in real life.

The BSD license even allows software to be taken from that public commons and used in proprietary applications. There is no obligation for the licensee to return anything to the commons. But despite the absence of such an obligation, the BSD "gift of freedom" is being repaid over and over by companies

and individuals who see more value to them in giving software away under an academic license than in keeping it private.

BSD License as Template

The BSD license has been through several revisions. The current version discussed in this chapter has been redesigned to work as a *template* appropriate for software other than the original Berkeley Software Distribution. A copy of the current version of the BSD license is shown in the Appendices.

When a licensor says "I license my software under the BSD license," that licensor is not suggesting that he or she is or represents the University of California, or that the licensed software is or is derived from the original Berkeley Software Distribution. Instead, this sentence means only that the license is in the form of the BSD license, inserting the licensor's own name as the name of the licensor and an original copyright notice instead of the copyright notice for that other university's software.

If you use software that purports to be under the BSD license, look for the license itself somewhere in the source code of the software. The license should be complete, with all blanks filled in. That text is the authoritative version of *your* BSD license, not the version shown in the Appendices.

The BSD License Grant

Here is the actual BSD license grant:

> *Redistribution and use in source and binary forms, with or without modification, are permitted provided.... (BSD license.)*

I will describe the *provided...* clause (what is also called a *proviso*) soon, but first I need to describe just which of the

University of California's intellectual property rights were actually being licensed by the first BSD license.

Almost everyone believes that the redistribution and use clause of the BSD license was intended to include all of the exclusive intellectual property rights the University then owned for something called the "Berkeley Software Distribution." The fact that the BSD license does not expressly list those exclusive rights (e.g., copy, create derivative works, distribute, perform, display, make, use, sell, offer for sale, import) doesn't mean they intended any of those rights to be excluded from the license.

The term *redistribution* means *distribution again*. This necessarily includes the right to make copies, since one cannot distribute software again without making copies. And since the word *modification* later in the sentence implies *derivative work*, I assume that the license allows the copying and distribution of both the original and derivative works. The word *redistribution* in the BSD license appears to encompass all those copyright rights that must be granted to ensure software freedom. The BSD license passes the filter of the Open Source Principles.

The word *use*, on the other hand, is not found among the exclusive rights of *copyright* owners. The *use* of software can be affected by a *patent*, because under the law, a patent owner has the exclusive right to *make, use, and sell* any product in which the patent is embodied. But the University of California made no patent grant in the BSD license. Indeed, later in the license the University specifically used the phrase *this software is provided by the copyright holders and contributors*, suggesting by its absence that there are no patent holders or that those patent holders are not granting anything in this license.

In the absence of an explicit patent grant, but considering the word *use* in the license, can we assume that the BSD license impliedly grants enough of whatever patent rights the University of California then owned that a licensee may use the software as it was originally distributed by the University? Most licensees under the BSD assume it does on the theory that otherwise the copyright license would be of no value. What good, they say, is software that can be copied but not used?

Such a conclusion is not based on the law of licenses. Indeed, a *bare license of copyright* need not include a *bare license of patent* at all. It is only if the BSD is viewed as a contract that we can introduce contract law principles such as *reliance* or *reasonable expectations of the parties*. If software is licensed under the BSD without forming a contract between licensor and licensee, the extent of any patent grant is at best ambiguous.

As to whether an implied grant of patent rights extends to versions of the software *with modifications*, that's an even more complicated question. The BSD license is silent about a patent license for derivative works. So if a licensee improves the original Berkeley Software Distribution in a way that infringes a patent owned by the University of California, there is no easy way of knowing whether an implied BSD patent license includes a patent license for that improvement.

Since courts are likely to construe implied grants of license narrowly, a licensee should consider obtaining separately from the licensor an explicit grant of patent rights that might be needed for modified versions of BSD-licensed software.

Source and Binary Forms of Code

In the late 1980s, when the BSD license was new, software was written in source code and compiled into a binary form

for execution. Those terms now have more complex practical meanings, with computer programs written in a variety of languages and executed by computers in many forms other than binary.

The phrase *source code* is assumed to mean the form of the software in which it was originally written by a human being. Used in this way in the BSD license, the phrase *source code* does not necessarily include any documentation about the program or even instructions on how to modify the source code.

Nothing in the BSD license actually requires the publication of the source code, either by the licensor of the original software or by the licensee of modified versions. Distribution in source form is merely *permitted*. However, any software someone might attempt to distribute under the BSD license without at least making source code available upon a licensee's request would, as a practical matter, merely be ignored by the open source community; it would find no projects willing to accept it.

Conditions under the BSD

The BSD license includes the following proviso that must be met for source code distributions:

> *Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. (BSD license.)*

The phrase *above copyright notice* is somewhat misleading. Presumably the BSD license really refers to the actual copyright notice that is displayed on the software being distributed rather than the copyright notice shown above in the license, for otherwise this would be a meaningless requirement.

The phrase *this list of conditions* includes three items: the requirement for source code distributions quoted above and two other conditions. The second condition is:

> *Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. (BSD license.)*

The phrase *binary form* is assumed to mean something broader than what that term meant in 1989, what we now more commonly refer to as the *executable* form of the software. BSD-licensed software may be distributed in binary (executable) form alone, without source code.

I assume that the requirement to include *the above copyright notice* in binary distributions means the original copyright notice valid for the work itself rather than the copyright notice shown in the license. And since there is no actual requirement to provide *documentation and/or other materials* with the distribution, it isn't clear that the *above* copyright notice will ever actually be seen by users.

The third BSD license condition relates to the name of the licensor, either *University of California, Berkeley,* for the original Berkeley Software Distribution or, since the BSD license is a template, whatever the BSD licensor's name is:

> *Neither the name of the <ORGANIZATION> nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission. (BSD license.)*

The *name* of a company or individual is not a copyright or patent, but it is nevertheless an important property interest that is protected by law in many countries. It can—and from the perspective of the open source community of contributors *should*—be protected from association with other people's

work or products. (Remember item 5 of the Open Source Definition in Chapter 1, although it is not included as a mandatory feature of open source licenses in the Open Source Principles.)

The BSD license explicitly prevents the name of the licensor or contributors from being used *to endorse or promote products*. This restriction clearly covers marketing activities. It probably doesn't cover otherwise naming the original licensor and contributors, as long as those names aren't used for product endorsement or promotion.

A more comprehensive requirement concerning advertising was present in the original BSD license:

> *All advertising materials mentioning features or use of this software must display the following acknowledgement: "This product includes software developed by the University of California, Berkeley, and its contributors." (Previous version of BSD license.)*

This condition was removed from the BSD license in 1999 after extensive public criticism of that requirement. Many people complained that it is one thing to prohibit the use of the licensor's and contributors' names for publicity purposes (i.e., the third condition already discussed), but it is quite another to require that a specific advertisement for the University be included in all advertising materials for the software or its derivative works. The concern was not merely for the University of California's one-sentence advertisement, but that other licensors using the BSD template could demand even more grandiloquent advertisements that create unacceptable burdens for subsequent creators of derivative works. Such advertising demands are no longer acceptable for open source licenses because they interfere with the freedom to create derivative works.

There are other forms of *reputation* interests of this type, such as property interests in trademarks, which are not mentioned in the BSD license. The BSD license refers only to *names* and doesn't explicitly say that a licensor's trademarks can't be used to endorse products. Even in the absence of a provision relating to trademarks, however, the law of unfair competition, at least in the United States, prevents a licensee from using a licensor's trademark on different but similar goods without the licensor's permission.

Warranty and Liability Disclaimer

The BSD license contains a *warranty and liability disclaimer*. It is reproduced here, but not in the all-capital-letters form of the original license text.

> *This software is provided by the copyright holders and contributors "AS IS" and ANY EXPRESS OR IMPLIED WARRANTIES, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of this software, even if advised of the possibility of such damage.*

The reason such provisions are often shown in all capital letters is that the law requires that these provisions be prominent so licensees will notice and read them. But capital letters are harder to read and are frequently ignored simply because of

the printing. I much prefer to capitalize only very important words, such as the words *AS IS* in the above disclaimer, to highlight what is truly important.

The first sentence of the BSD disclaimer deals with warranties and the second sentence with liability. A disclaimer of warranty is independent of a disclaimer of liability. The BSD warranty disclaimer makes it clear that the licensor promises nothing about the software, and the liability disclaimer makes it clear that the licensor will not pay for any kind of damage, however caused.

A software warranty is a promise relating to such things as the quality, effectiveness, and reliability of software. Under the BSD license, there are no such promises. The licensor only promises to allow the user to practice the licensor's exclusive copyright (and perhaps patent) rights, nothing more.

Contract law and consumer protection laws provide for certain express and implied warranties. The BSD license intends to disclaim absolutely all of them. That is generally what the words *AS IS* means in contract law. Whatever faults or defects exist in the software as licensed, and whatever problems are later encountered while using the software, are not the licensor's concern.

The laws in some jurisdictions override warranty disclaimers in licenses and contracts. For example, in the United States certain warranty disclaimers for a consumer product are ineffective and will generally be ignored by the courts. Software by itself is not a consumer product under this law, but when software is combined into a consumer product such as a PDA or television recorder, the warranties of merchantability and fitness for a particular purpose cannot be disclaimed for that product regardless of what a license says, at least in the United States.

The second sentence of the BSD disclaimer deals with the liability of the licensor to pay damages actually incurred as a result of the use of the software.

The BSD license disclaims liability of any sort. This means that any damages caused by the software, whether to people, to computers, or to the licensee's business, are not going to be paid for by the licensor. Such liability disclaimers may not be legally effective in certain jurisdictions, particularly for consumer products. If a company distributes a consumer product that causes harm to people or property, the distributor may be liable regardless of what a license says.

It is unlikely that a court would extend liability in such a situation all the way up the chain of title to the contributor or distributor of general purpose software that happens to be included in a consumer product, but that is a factual situation that would need to be analyzed by an attorney at the appropriate time. The disclaimer language in the license, the characteristics of the software, and the existence of an agreed contract rather than just a bare license would be among the relevant facts that a judge would consider in determining whether a liability disclaimer is fair, under the circumstances, to an ordinary consumer who is injured by a software-based product.

The MIT License

The lawyers at the Massachusetts Institute of Technology (MIT) created their own version of the BSD license. They cleaned up some of the vague language of the BSD license and made their version simpler to read and understand. A copy of the current version of the MIT license is shown in the Appendices.

The license grant of the MIT license reads as follows:

> *Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions.... (MIT license first paragraph.)*

This improves on the BSD license by specifically mentioning all of the exclusive rights under copyright law and almost all of the exclusive rights under patent law (e.g., "make" is omitted, but that is probably unnecessary given the other verbs in that sentence). No longer are we limited by the BSD's reference to *redistribution and use*. On the other hand, the new phrase *deal in the software* has no precise legal meaning. In light of the longer list of rights in the MIT license grant, it appears not to limit copyright or patent rights in any way.

Like the BSD license that preceded it, the scope of the patent grant in the MIT license is implicit rather than explicit. This means that a licensee cannot be sure that the *implied* patent rights granted by MIT are broad enough to cover derivative works.

The grant in the MIT license extends not just to the software itself but to its *associated documentation files*. It is not clear whether MIT is offering here to provide all documentation in its possession concerning the software or only certain files that are associated in some way with the software.

The phrase *free of charge* means that the licensor (MIT in this case) will not charge a royalty or license fee. But the word *sell* among the list of rights granted means that downstream licensees are not restricted in any way from charging their cus-

tomers royalties or license fees for modified versions of the software.

The MIT license also serves as a license template. It is so short a license that only the copyright notice needs to be changed to fill in the template. Unfortunately, the phrase *this software and associated documentation files* doesn't clearly identify which software the license applies to. The only way to correlate particular software with a particular copy of the MIT license is to physically find the license text in the source code of the software.

The Right to Sublicense

The MIT license also grants the right to sublicense, a word missing entirely from the BSD license grant. Sublicensing is an important concept in open source licensing.

Referring back to the chain of title explanation earlier in Chapter 2, I described how contributions from many people can be combined into collective and derivative works, and how those works can in turn be used by others to create still more collective and derivative works. That is the very premise and promise of open source development. The ever lengthening chain of title is reflective of the robust creative energies of community development. A major open source software program may have a long chain of title by the time it arrives on your computer.

From whom does the person at the end of the chain of title get a license to use, copy, modify, and distribute the software? Does the user receive a set of licenses, one from each of the original authors of each of the contributions all the way along the chain, or is there a single license from the immediate predecessor on which the user can rely?

If a license *is not sublicensable*, then only the owner of the original work can grant licenses. For each nonsublicensable component of a collective or derivative work, each prospective licensee must obtain a license to that component directly from its owner. In principle this requires tracing the entire chain of title, obtaining copies of each copyright or patent license up the chain, but in practice it means nothing so complicated. Leaders of nonsublicensable open source projects take steps to ensure that licenses to components will be available for the asking, but they don't actually expect everyone to ask. The project team merely announces that licenses are available and points to the open source code, with its copyright, patent, and other attribution notices there for all to read, for information about where to get those licenses. If you want to make sure you have a license to each component, they in effect say, go get it yourself; but considering the low risk, most licensees don't bother. This is a reasonable solution for most open source software, but as a legal matter it is risky not to confirm that all licenses up the chain are actually available.

On the other hand, if a license *is sublicensable*, then any distributor has the right to grant a license to the software, including its component parts, directly to third parties. For each sublicensable work that is a component of a collective or derivative work, each prospective licensee obtains a license directly from the owner of the collective or derivative work. Leaders of sublicensable open source projects take steps to ensure that licenses to components are consistent with their own licensing terms and are sublicensable. They then extend sublicenses to their customers sufficient to allow those customers to exercise their rights under the open source licenses.

Note that the license terms for a sublicense must be consistent with—not necessarily the same as—the original license terms. A sublicensor cannot sublicense more rights than have

been granted by the original author. The sublicensors needn't use the identical words as in the earlier license they received, but they cannot override terms and conditions that are mandated by that license.

This subject will be addressed again in Chapter 10 when I discuss how open source projects should in-license contributions and how they can relicense their collective and derivative works when new and better licenses become available despite being bound by the licenses of their contributors.

The fact that the MIT license is sublicensable is an advantage for anyone who wants to distribute copies or derivative works of MIT-licensed works. A distributor can provide to his customers all the rights needed to the entire work without expecting those customers to follow the chain of title to its beginning.

The Warranty of Noninfringement

Another important aspect of the MIT license is its disclaimer of the *warranty of noninfringement*. This concept is entirely missing from the BSD license. Here's how the MIT license says it (converted from uppercase letters):

> *The software is provided "AS IS", without warranty of any kind, express or implied, including but not limited to the warranties of merchantability, fitness for a particular purpose and noninfringement. (MIT license third paragraph.)*

You can *infringe* someone's intellectual property by exercising any of the exclusive rights of the owner of that intellectual property—copyright or patent—without a license to do so. If you copy, modify, or distribute copyrighted software without a license, or if you make, use, sell or offer for sale, or import a patented invention without a license, you are an *infringer*.

Infringement can happen accidentally, but infringers can be penalized even if the infringement is not intentional. If software that you use infringes someone's intellectual property and you have no license to do so, a court may assess damages and order you to stop the infringement, no matter how costly or disruptive that may be to your business.

Infringement may not be the fault of the licensor who distributed the software to you. A patent owned by some third party of whom neither of you were aware may suddenly be asserted against all users of the software, including you and your distributor, and suddenly you can find yourself accused of patent infringement. As for copyrights, to your own and to your software distributor's surprise, some third party may assert that somewhere in the chain of title to the software someone made a mistake or committed a fraud, turning what everyone thought were legitimately licensed copies into infringing copies.

Warranting against infringement is an impossible burden to impose upon an open source licensor who is, after all, giving software away for free. No open source license provides a warranty of infringement. Neither, for that matter, do most proprietary software licensors, because the uncertainty and potential cost of infringement are far too expensive a risk to take. Even those few software companies that do provide a warranty of infringement typically limit their liability to the purchase price of the software; this is a trivial amount considering the potential costs of infringement.

Even for open source licenses that don't mention the warranty of noninfringement, the "AS IS" phrase should warn you that a meaningful warranty of noninfringement is simply not available. Where it is critical to your business that you avoid infringement risk, you must accept the burden to perform your own diligent analysis of the chain of title, or purchase

your own insurance policy to protect you. It is foolish to look to typical software licenses—and certainly to open source software licenses—to eliminate your risk of copyright or patent infringement.

The Apache License

Of the two most widely known and successful open source projects, Linux and Apache, only the latter is licensed under an academic license. That means—as is true for any software licensed under an academic license—that Apache software may be used by anyone, anywhere, for any purpose, including for inclusion in proprietary derivative works, without any obligation to disclose source code.

A copy of the current version of the Apache license is shown in the Appendices.

The first difference between the current Apache license and the BSD license is the following provision:

> *The end-user documentation included with the redistribution, if any, must include the following acknowledgment: "This product includes software developed by the Apache Software Foundation (www.apache.org/)." Alternately, this acknowledgment may appear in the software itself, if and wherever such third-party acknowledgments normally appear. (Apache License section 3.)*

This provision differs significantly from the rescinded advertising clause of the original BSD license. (As a reminder, here's how that provision read: "All advertising materials mentioning features or use of this software must display the following acknowledgment: This product includes software developed by the University of California, Berkeley and its contributors.") The Apache license only requires an acknowledgment in "end-user documentation" or "in the software itself," not in

"all advertising materials." The Apache license does not specify the prominence that must be given to that acknowledgment. The Apache license is consistent with the Open Source Principles because it does not interfere with the freedom to modify or create derivative works of open source software.

Protecting Trademarks

The most important feature of the Apache license that distinguishes it from the BSD and MIT licenses is that it specifically protects the Apache trademark. This is an acknowledgment that trademarks are important assets of open source projects.

Here's what the license says:

> *The names "Apache" and "Apache Software Foundation" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact apache@apache.org. (Apache License section 4.)*
>
> *Products derived from this software may not be called "Apache", nor may "Apache" appear in their name, without prior written permission of the Apache Software Foundation. (Apache License section 5.)*

On the surface, this is similar to the BSD provision preventing the University of California name from being used "to endorse or promote products." But the Apache license goes even further when it states that a derivative work may not use "Apache" as part of its name.

Trademarks are brand names of products. You will recall that a *trademark* is a word, name, symbol, or design used to identify a company's products and to distinguish those products from the competition. Trademarks are a form of intellectual property.

Open source software poses some difficult marketing problems. The licenses under which such products are distributed require the distribution of source code and permit the creation and distribution of derivative works. It is difficult for a distributor of such products to compete on price alone, because almost any knowledgeable company can undercut the price by simply copying the original software.

Trademarks can be particularly useful in this kind of environment. A company can demonstrate that its software is of high quality, reliable, efficient, feature-rich, and user-friendly. It can promise continual enhancements, product support, user groups, and other goodwill activities. Then over time, through those marketing efforts, that company's customers will begin to associate its trademarks with that software. New or repeat customers will pay for software they perceive to be worth the price even though there may be cheaper competitive products. Customers will select products whose trademarks they identify.

It would not be fair to allow a licensee who receives free software to also receive a license to the valuable trademarks of his licensor. The Apache license makes it clear that the Apache trademark isn't licensed along with the software.

The Apache Contributor License Agreement

The Apache Software Foundation (ASF) has recently begun to require its contributors to submit a signed Contributor License Agreement. This agreement is copied in the Appendices.

The Apache Contributor License Agreement is intended to convey to ASF all necessary rights to the contributor's intellectual property so that ASF can do what it wishes with those Contributions. The agreement itself asserts that the goal of the Contributor License Agreement is to protect the Contributor:

> *This license is for your protection as a Contributor of software to the Foundation and does not change your right to use your own contributions for any other purpose. (Apache Contributor License Agreement, initial paragraph.)*

In fact, the main purpose of the Apache Contributor License Agreement is to protect ASF in two important ways:

1. It allows ASF to license its collective and derivative works including the *Contribution* under any license it chooses. That gives the ASF flexibility regarding relicensing. (Relicensing is discussed more fully in Chapter 10.) The Apache Contributor License Agreement does not constrain ASF's licensing options for collective and derivative works in any way.
2. It allows ASF to assert that each *Contribution* is actually owned by its *Contributors*, and that third party licenses and restrictions known to the *Contributors* have been divulged. This will make it possible for future Apache licenses to convey a *warranty of provenance*. (That term is described in Chapter 9; the OSL and AFL licenses contain an express *warranty of provenance*.)

Contributor agreements such as the Apache Contributor License Agreement are *licenses*, in both name and effect. They convey copyright and patent rights, as do all the other open source licenses described in this book. But these contributor agreements are not submitted to Open Source Initiative for its review and approval, and so there is no established process for verifying that those agreements are compatible with the Open Source Principles.

This book is also not the place to do that analysis. I will suggest, however, that this contributor agreement, in use by the Apache Software Foundation, is truly open source, based upon my own reading of its terms. Whether that is true for the contributor agreements demanded by other projects remains an open question. Contributors should seek their own legal advice before signing such contributor agreements.

Contributor agreements are relatively new to open source software projects, but they are not new to other industries. Musicians, journalists, photographers, and other contributors of intellectual property have often been asked to sign contracts with their publishers under which they grant broad intellectual property rights. Through the passage of time, some of those works have dramatically increased in value, and the publishers have sometimes failed to share their profits.

While that is not a likely result when the publisher is a nonprofit open source project such as the Apache Software Foundation, not all open source projects are (or will remain) benign; not all projects serve the public interest. Each contributor should decide for himself or herself whether to sign a contributor agreement.

Nor is a contributor agreement always necessary. If an open source contribution is submitted under a *compatible* open source license, no other contributor agreement is necessary. Chapter 10 discusses open source license compatibility.

The Artistic License

The Artistic License was the first open source license to protect the rights of software authors to attribution and integrity. In the U.S. Copyright Act, those protections apply, as a matter of right, for authors of works of visual art. The law provides that:

> *...The author of a work of visual art (1) shall have the right (A) to claim authorship of that work and (B) to prevent the use of his or her name as the author of any work of visual art which he or she did not create; (2) shall have the right to prevent the use of his or her name as the author of the work of visual art in the event of a distortion, mutilation, or other modification of the work which would be prejudicial to his or her honor or reputation.... (17 U.S.C. § 106A.)*

Software is not a work of visual art, however, so it is not subject to this provision of the law. But a license expresses the *law of the contract*, and in the case of the Artistic License, the law of *this* contract protects software authors' rights to attribution and integrity. It does what the copyright law doesn't do—protect the rights of software artists.

The ways in which the Artistic License does this are interesting and effective, albeit legally confusing. But before I deal with this, I need to comment on the structure of that license—a preamble about preambles.

License Preambles

The Artistic License is the first of the academic licenses to consider its message important enough to warrant a license preamble. A copy of the current version of the Artistic License is shown in the Appendices. Its preamble starts as follows:

> *The intent of this document is to state the conditions under which a Package may be copied, such that the Copyright Holder maintains some semblance of artistic control over the development of the package, while giving the users of the package the right to use and distribute the Package in a more-or-less customary fashion, plus the right to make reasonable modifications. (Artistic License preamble.)*

Preambles to open source licenses are occasionally written in strident political or philosophical terms (although the preamble to *this* Artistic License is not stridently political), intended to convince others of the rightness of the licensor's position rather than to inform licensees of the rules they are to follow. Many lawyers believe license preambles are a bad place to make a political or philosophical statement. There are two reasons for that:

1. Licenses establish terms and conditions governing the relationship between two parties, in our case a licensor and a licensee. The preamble is not a term or condition. It is merely a statement. Therefore, it has no positive legal effect and it is not binding on a court. As such, it is surplusage.
2. The preamble may subtly conflict with the actual rules, or may be stronger or more conciliatory than the actual license provisions. There is no absolute rule that tells a judge that, in the event of a conflict between the preamble and the license terms, the license terms prevail. How a court will rule in the event of an actual conflict between the license and the preamble is difficult to predict.

While preambles and other philosophical arguments should not be used to qualify or modify the terms of software licenses, the points they make are important to some licensors. The software artists who wrote the Artistic License (and, as I will soon describe, the free software activists who wrote the GPL) spent a lot of time crafting their preambles. Those preambles should be read as general statements of the licensor's intent rather than as legally binding terms and conditions.

When Amateurs Write Licenses

The same programmers who cringe when a lawyer attempts to write high-quality software feel no qualms about writing their own open source licenses. Their goal, it appears, is to craft something that sounds like a license, to define a form of software freedom with reasonable terms and conditions, and then wait for the community to adopt the license and distribute software under it. This technique sometimes works. Some members of the open source community are more concerned with making a philosophical statement, getting free software distributed to the world, and letting license enforcement take care of itself somehow in the future. That can be a commendable goal, but from a lawyer's perspective, it is amateurish and risky.

The Artistic License is one such amateur license. It is a license that a lawyer would have difficulty explaining and that a judge would probably not be able to understand. I will incautiously invoke the wrath of the authors of that license by candidly expressing my concerns about it. In this, I don't mean to be harsh to them personally; I'm really trying to make a point about the art of license drafting. I know what those authors were trying to say, and I support their goals of artistic attribution and integrity, but I believe they made a legal mess of it.

Here are a few examples from the definitions in the Artistic License:

> *"Package" refers to the collection of files distributed by the Copyright Holder, and derivatives of that collection of files created through textual modification. (Artistic License definitions.)*

This definition of *Package* assumes that a licensor is distributing only one collection of files; assumes that the phrase *collection of files* has a clear meaning; confuses the terms *derivative works* and *collective works* by referring to *derivatives of that col-*

lection; and then describes the process by which derivative works are created as involving something called *textual modification* (what other kinds of modifications are possible?).

> *"Standard Version" refers to such a Package if it has not been modified, or has been modified in accordance with the wishes of the Copyright Holder. (Artistic License definitions.)*

The law has little to do with *wishes*. The law of contracts has nothing to do with enforcing the *wishes* of a party, or even determining what those wishes are. Precatory language about *wishes* creates what in law are called *illusory rights and obligations*; such language is unenforceable.

> *"You" is you, if you're thinking about copying or distributing this Package. (Artistic License definitions.)*

The law has little to do with what people *think*. A person does not become a licensee of intellectual property merely by *thinking* about it.

> *"Reasonable copying fee" is whatever you can justify on the basis of media cost, duplication charges, time of people involved, and so on. (You will not be required to justify it to the Copyright Holder, but only to the computing community at large as a market that must bear the fee.) (Artistic License definitions.)*

The courts don't care about matters that the parties to the license admit is not important enough to *justify* to the copyright owner. The only point of this definition of r*easonable copying fee* is for the authors to describe a law of economics, namely that the marketplace determines whether a price is reasonable. It has no legal significance whatsoever.

At various places the Artistic License refers to the *public domain*. (The *public domain* was explained earlier in Chapter 2 when I discussed the duration of copyright and patent.) The

use of that term in the Artistic License is misleading. For example:

> *You may apply bug fixes, portability fixes and other modifications derived from the Public Domain or from the Copyright Holder. A Package modified in such a way shall still be considered the Standard Version. (Artistic License section 1.)*

What the authors of this license may have meant was that modifications derived *from other open source works*, because there is so little software actually available in the *public domain*. It is not clear how works licensed under different licenses will interact legally with works licensed under the Artistic License. I will discuss the complex issue of license compatibility later in this book.

I understand that the authors of the Artistic License wanted to retain some control over subsequent derivative and collective works. In this, they subtly cross the line that distinguishes academic and reciprocal licenses. An academic license, remember, imposes no burdens or obligations on the creator and distributor of collective and derivative works. However, the Artistic License imposes burdens and obligations that require the licensee "to place ... modifications in the public domain or otherwise make them freely available" (§ 3[a]) and "to rename any non-standard executables" (§ 3[c]). It requires distributors of executable versions of the licensed software to "accompany the distribution with the machine-readable source of the package with ... modifications" (§ 4[b]) and to "document clearly the differences" between the standard version and the modified version (§ 4[c]). There is one other option, to "make other distribution arrangements with the Copyright Holder" (§§ 3[d] and 4[d]). All of these requirements can be avoided, however:

> *You may distribute this Package in aggregate with other (possibly commercial) programs as part of a larger (possibly commercial) software distribution provided that you do not advertise this Package as a product of your own. (Artistic License section 5.)*

Given the confusing language in the Artistic License, I suggest that the best way to deal with it is to treat it as an academic license granting broad freedom to copy, modify, distribute, make, use, and sell the original software. If you distribute copies or derivative works of software licensed under the Academic License, you are obligated to attribute the original software to the original author, and to make it clear to your licensees that you—and not the original author—are responsible for your derivative works. Because of the ambiguity, in legal terms, of the terms *aggregate* and *larger*, this is an easy out. With the broad exception provided in this section 5, it appears, the other strictures in the Artistic License can be easily avoided simply by being careful not to advertise the software as a product of your own.

Big Picture of Academic Licenses

As you have seen, academic open source licenses are typically short and to-the-point. Often less than a page in length, academic licenses intend to grant to everyone all the copyrights and patent rights needed to exercise software freedom. There are few conditions in such licenses. A licensee, at most, needs to accept the absence of warranty or liability and to acknowledge the contributions of the original authors.

The brevity of most academic licenses is encouraging to users but somewhat perplexing to attorneys. Before open source licenses, it was not unusual to see multi-page licenses, with lots of terms and conditions that clearly defined the expectations of the parties. But with open source academic licenses, licensors have no expectations for what happens with

their works. In a form of generosity not typical for major software companies, those licensors are entirely comfortable giving up any vestiges of control over what happens to their works after they are released to the world.

A different kind of academic license, the Academic Free License, handles the academic open source bargain in a more comprehensive way. I will defer commenting on that license until Chapter 9, after I describe the GPL and other reciprocal licenses in the next few chapters.

Apache License Version 2.0

While I was finishing the final edits for this book, the board of directors of the Apache Software Foundation approved version 2.0 of the Apache License. I debated with myself whether to insert a review of that license here. I was reminded of a cat chasing its tail. If I delay the publication of this book for every new license that comes along, I'll never finish.

The Apache License version 2.0 is a much more robust open source license than the other academic licenses already discussed in this chapter. It deserves careful analysis, perhaps a chapter all its own like the GPL, MPL, CPL, and OSL/AFL licenses in the chapters that follow this one. It is a very good open source license, a dramatic improvement over its predecessor. I decided that, rather than try to catch that Apache License for this book, I will use it only as an object lesson: Open source licensing is part of a dynamic, fast-moving world. New licenses and licensing strategies are introduced constantly. Companies that intend to play seriously in the open source marketplace will want to dedicate some effort to remaining current. This book unfortunately doesn't have all the answers.