3

Distribution of Software

Contributors and Distributors

Open source software is written by computer programmers who generously distribute it to their friends, employers, or customers. Often these programmers work for companies that aggregate code written by many programmers into a functional whole; those companies then distribute the aggregated work to the world. Important computer software is usually too big and complicated to be written by one person acting alone—although each component of software always starts with one person acting alone—and it almost always requires collaboration and joint development.

This is not a unique process to open source. Commercial software has long been created and distributed collaboratively. What is unique about the open source process is that once software has been licensed under an open source license, the collaborative process is no longer tied to a single individual or company. Because software freedom is promised by every open source license, users are free to take control of the software and do whatever they want with it. Everyone is free to become a contributor to or distributor of open source software, starting from anyone's open source software. At least that is the promise, although incompatibilities between open source licenses are preventing that goal from being completely met. License compatibility is discussed in Chapter 10.

Distribution

I have thus far used the word *distribution* as if it had obvious meaning in the software world. Certainly it means selling or giving copies of software away to others. It also may include such arrangements as incorporating software into consumer or industrial products and selling those products to others. For some software, it may also include making the software available across a network for execution by others.

In the proprietary software world, before a company may become a distributor it must negotiate a formal business arrangement with the owner of the software. These contracts typically establish marketing arrangements, territorial limitations, pricing structures, and other business terms.

None of this is needed for open source software. Because of the objective to provide software freedom as specified in the open source definition, the distribution of open source software cannot be restricted in those ways. (See Open Source Principles # 2 and 3.) An open source license must grant everyone permission to make copies, to create derivative works, and to distribute those copies and derivative works. Anyone, anywhere, for any reason, may become a distributor of open source software.

There may be no time, place, or manner limitations on distribution in an open source license—but this does not mean that there may be no conditions on distribution at all. Open source licenses may condition the distribution of derivative works on reciprocity of licensing, an important device first used in the GPL. (Reciprocal licenses are introduced in Chapter 6.) Certain open source licenses include an obligation to provide a reference implementation of derivative works that are distributed, so that standards can be enforced. (The Sun Industry Standards License is discussed in Chapter 13.) And finally, open source licenses can use their own definitions of the term *distribution* to include or exclude network execution of software, the so-called *application service provider* exception. (The OSL and AFL licenses described in Chapter 9 have such a provision.) These qualifications and limitations to the term *distribution* are explained in due course when specific open source licenses are described.

Open Source Collaboration

Open source software is distinguished from most other commercial software because its development frequently takes place collaboratively among many individual developers, working alone or for different companies, without contracts or other formal arrangements among them. Worldwide communities of software engineers dynamically form and grow on the Internet. Participants discuss among themselves what needs to be implemented; allocate the design, programming, and documentation tasks to those who volunteer to do them; and eventually publish one or more working programs for all to use. That is how major open source programs like the Linux operating system and the Apache web server were initially developed.

In the case of Linux, that open source development project is coordinated by an overall project leader, Linus Torvalds. The Linux team and Torvalds evaluate the quality of contributions they receive from around the world, and they decide whether to include those contributions as a part of Linux. The Linux project has formal mechanisms for evaluating and testing contributions, and there is a collective rather than dictatorial decision process, as befits the importance of Linux to the computing community and the collaborative bent of the project leaders.

Torvalds continues to lead the Linux development project. He effectively controls the main intellectual property of the Linux operating system, such as the Linux trademark, although many thousands of programmers and companies are always deeply involved in its development and distribution.

In contrast, a board of directors coordinates the development activities of the Apache Software Foundation, a nonprofit corporation that is the distributor of the Apache web server and many other open source packages. Many of the leaders of the Apache project work for software companies that donate their employees' time and software to the Apache Foundation. Important decisions relating to Apache are decided by open vote and consensus.

These are only two of a wide variety of successful open source development models. Many open source projects are now managed by private companies that have found ways to turn software freedom into profitable enterprises, and by nonprofit foundations that serve the "public interest." But that remarkable and evolving story is not the subject of this book. Open source business models are topics for other books entirely.

Contributors to open source software can be individuals or companies. Their contributions are combined at the project level with the contributions of other individuals and companies into larger works. Those larger open source works, with their many contributions, are then distributed to the public. Some companies take software distributed by open source projects and aggregate it still further into their own open source products, which they then distribute. A single operating system like Linux, a single web server like Apache, or a single commercial product like a cell phone or a television recorder that includes Linux and Apache may be the result of many contributions by many original authors and distributors along the way.

It is not always easy to distinguish between a contributor and a distributor of open source software, because people aggregate software into larger systems at each step of the development and distribution process. A distributor becomes a contributor to the next higher level of the food chain, just as fish in the ocean become food for larger fish.

The roles and rules for contributors and developers, sometimes the same and sometimes different, are important topics for open source licensing to which I shall return frequently.

Contributor Agreements

Why do contributors contribute? There are certainly as many answers to that question as there are contributors. But one thing is certain: People contribute to open source projects whose goals they share. There is usually camaraderie among project members, whether the project is structured as a loose confederation, a formal nonprofit corporation, or a corporate-sponsored activity. When camaraderie fails—for either technical or personal reasons—projects may fork into rival projects. Open source contributors are free to join either fork or leave altogether. Such forks, by the way, have proven to be very rare in open source projects.

Contributors may leave a project but their contributions remain. Once software is made available to a project under an open source license, the project may continue to copy the software, create derivative works from it, and distribute it even after the contributor's participation ends. That is because open source licenses are perpetual, even though most licenses don't expressly say so. As long as the project continues to honor the terms of the licenses under which it received contributions, the licenses continue in effect. There is one important caveat: Even a perpetual license can be revoked. See the discussion of bare licenses and contracts in Chapter 4.

For most projects, receiving contributions under an appropriate open source license from the contributor provides more than enough authority to do what they need to incorporate the contribution into the project's software. That, after all, is what the Open Source Principles stand for.

As long as each contributor's license is compatible with the project's open source license used for its distributions, then the contributor/distributor food chain evolves as I described in the previous section. This is always the case when identical licenses are used for contributions and for the project's derivative works. For example, if a project accepts contributions under the BSD license, it can then license derivative works under the BSD license; if it accepts contributions under the GPL, it can then license derivative works under the GPL.

But compatibility encompasses much more than simply identical licenses. A contributor license for his contribution is compatible with a project license for its collective or derivative work if the contributor's license contains no terms or conditions that would conflict with the terms and conditions of the project's license. Determining whether two licenses have conflicting terms and conditions requires a provision by provision comparison of the two licenses.

That comparison must be analyzed separately in each direction. For example, as I shall describe later, a contributor license like the BSD license is compatible with the other project licenses in this book, including the GPL, but the converse is not true; contributions licensed under GPL cannot be used in BSD-licensed projects. Incompatibility may exist in both directions; GPL-licensed contributions cannot be used by the

Apache project and Apache-licensed contributions cannot be used by GPL projects. I will have much more to say about license compatibility in Chapter 10.

What happens if a project decides that it wants to use a contribution in a way that is incompatible with the terms of the contributor's license? The answer is obvious: The project is bound by the terms of the licenses under which it receives contributions. In general, if the contributor's license is incompatible with the project's open source license, then the project cannot use the contribution.

Open source projects are usually not the owners of the copyrights in the contributions to them, and they have no right to change those licensing terms on their own. Sometimes, to ensure that they have freedom to choose licensing terms, open source projects seek to own the copyrights in contributions made to them, or to enter into written agreement with contributors that expressly allows the projects to decide license terms for contributions. These contributor agreements take the form of copyright and patent assignments that actually transfer ownership of the intellectual property, or broad license grants much more comprehensive than the open source licenses in this book. License compatibility is not an issue for projects that are copyright and patent owners, because the contributors no longer have any right to refuse the projects' licensing decisions for contributions the contributors no longer own.

What happens, then, if an open source project faces an actual relicensing decision but it doesn't own the copyrights and patents in its contributions? For compatible relicensing, no additional license is necessary. But it must obtain the agreement of the contributors to any relicensing that is incompatible with the terms of the license it received from its contributors.

Who should have the right to make future licensing decisions about contributions, the project or the contributor? There is no single answer to this question in the open source community. In fields other than software, this issue has long been a fruitful source of litigation. Musicians and artists have often fought against their own publishers, to whom they once willingly assigned their copyrights, trying to regain those valuable rights for other markets. In recent years, contributors to newspaper articles fought against their own publishers for the rights to republish their articles in new online forums. These cases often turn on the interpretation of contributor agreements. Of course, had they been handled as copyright or patent assignments, no rights would remain and the musicians, artists, and newspaper writers would have been without recourse regardless of what decisions their publishers made.

I personally don't want to give up too much control to my publisher. When the words are mine, I want to own them. I will license them to everyone under an appropriate open source license, but I will not give them away to someone else who can then elect to take them private or license them in ways of which I don't approve. This is true no matter how much I like my publisher, and no matter how much I want to save my publisher from having to worry about future relicensing problems.

This is obviously just my own opinion about an issue of copyright policy. Each contributor of intellectual property to a project or to a publisher must decide for himself how many rights—and therefore how much control—to give away. Beyond this I will not advise and will merely proceed to explain the various kinds of open source licenses that projects adopt. If you intend to contribute to an open source project and it presents you with a contributor agreement different from an open source license, make sure you read it carefully and consult an attorney if you are unsure what you're being asked to give away.

What about Users?

I will begin in Chapter 4 to explain the broad categories of open source licenses—particularly academic and reciprocal licenses—that are available today. I follow that in Chapters 5 through 9 with detailed license descriptions of the major open source licenses.

Fortunately for *users* of open source software, none of the distinctions between academic and reciprocal licenses, or among the various project and company licenses described in this book, matter much. Individual *users* don't often have to concern themselves with the intricate conditions of these licenses, or warranties, or patent defenses, or other esoteric legal issues. Users of open source software typically do not create and distribute derivative works, so a reciprocity provision does not apply to them.

For these reasons, mere users of open source software can safely ignore the rest of this book. Open source software is completely free for users. All open source software, whether licensed under academic or reciprocal licenses, can be freely used by anyone, anywhere, for any purpose whatsoever. Copies of that software can be made without payment of additional royalties to the licensor and, for the most part, without concern about the specific license terms.