A History of Open Source

by Mark B. Rosenthal <mbr@arlsoft.com>

The history of the open source licensing concept begins two decades before the term "open source" was coined.

Let me first explain the meaning of the word "source" in "open source". Computers are designed to run a set of instructions in a language known as "machine code". It's possible for human beings to write programs directly in machine code. I know, because I've done it. But I also know that human beings do not find machine code a particularly hospitable language to write in. So high-level languages were invented to make programmers' lives easier. Early languages included Fortran, Cobol, Lisp, and C. Among the more recent languages are C++, Java, Perl, PHP, and Python. Since a computer cannot execute these languages directly, they have to be translated to something a computer can understand. Originally these translators came in two varieties: compilers which convert the high level language into a file containing machine code to be executed later, and interpreters which run the program by reading the high level language and doing what it says as it's being read. Nowadays, thanks primarily to the widespread adoption of Java, hybrid compiler/interpreter systems have become popular.

Anyway, a compiler reads its input from a source file and writes its output to a destination file. The destination file containing machine language is known as an executable file or a binary. The file containing the human-readable form of the program is known simply as the source file. So the human-readable form of the program came to be called "source code."

Up until sometime in the 1970s, it was common for computer manufacturers to distribute both the source code and the binaries of their software. Customers could fix bugs in the code without having to wait until the vendor decided their problem was important enough to bother fixing. In that era, it was commonplace for programmers to share code. Asking, "Hey, anybody got a good implementation of a hashing algorithm?" was the programmer's equivalent of dropping by your neighbor's house to borrow a cup of sugar. We didn't give it a second thought. But by the late 1970s, companies were starting to assert a proprietary interest in what programmers wrote, which began to make life hell for us. But although we all grumbled about it, none of us had any idea how to hang on to the congenial atmosphere that was gradually being wrested away from us. Well almost none of us. I did know one programmer who came up with an idea for what to do.

In the 1970s, my primary recreational activity was folk dancing in and around Cambridge. One of the people I knew from the local folk-dance community was a fellow by the name of Richard Stallman. Given the slightest opportunity, he would talk my ear off about how "they're taking away our freedom to program." At the time I thought he was nuts. Freedom to program? When I thought of fighting for freedom or fighting for people's rights, I thought of Viet Nam war

protests, Civil Rights marches, Women's Liberation. The idea that anyone would consider it important to fight for our freedom to write whatever code we wanted seemed ludicrous to me. By the late 1980s, as I observed the proprietary direction the industry had moved in, I thought back to these conversations and I realized that, far from being nuts, Stallman was far more able to see what was coming than I had been at the time. And since then, as computers have become more and more integrated into every aspect of our lives, lack of that freedom is directly connected to loss of many other freedoms that we as Americans take for granted.

Back when something could be done about the proprietary direction companies were pushing programmers in, Stallman was the only one of us to realize that source code could be protected by copyright law without the license having to say, "you don't have permission to use this unless you contact me and pay me whatever price I ask." He was the only one to actually put together a license which granted people permission to use his source code on condition that they behaved the way programmers had been used to behaving up until the late 1970s. And being a Lisp programmer, he naturally made his license recursive. This license originally went under the humorous name "copyleft," but is now known as the GPL, or General Public License. Copyleft is now used to refer to the general concepts, and the GPL is one example of a copyleft-compatible license.

To emphasize how critical this idea has become to the very freedoms the U.S. was founded on, let me give just one example. With the advent of computerized voting machines, the only way to verify that a silent coup d'etat isn't taking place is to allow any citizen with the ability and desire to do so, to inspect the source code to the programs that count our votes. But the companies that make the voting equipment have strenuously resisted making their source code public, and the courts have ruled that a company's interest in protecting its trade secrets trumps the public interest in being able to trust our election process. Our very democracy is at stake — and the solution is in Stallman's "copyleft" idea.

I know lots of people find Stallman difficult, myself included. He's adamant and unbending in his beliefs. But it's a double-edged sword. If he weren't adamant and unbending, the huge body of software that the world now lumps together under the single name "Linux" would never have come into existence. The idea of the GPL would not have made any difference if it had not become widely adopted. Stallman also founded the GNU project in Cambridge in the 1980s, and he and the GNU team wrote a great deal of code with the intention of producing a complete GPL-licensed replacement for the Unix operating system. The GNU code was of excellent quality, and since it came with source, you could fix what few bugs there might still be. It rapidly became the preferred set of utilities among programmers. And with its adoption, the idea of the GPL spread.

GNU was always intended to be a complete replacement for Unix, including a kernel. There was no need to write a windowing system, because the X Window System had been developed by MIT's Project Athena with government funding, which meant that the source code developed with that government funding had to be made available to the public.

By the late 1980s, the GNU project had produced a fairly complete set of replacements for Unix tools. But for reasons I've never understood, it took them more than a decade after that to release a working kernel.

Anyway, by the early 1990s the GNU tools were stable, they were pretty much the favorite development tools among Unix programmers, and they could easily be ported to the Intel architecture. The X Window System port to the Intel architecture (known as XFree86) was pretty stable by then too. The one missing piece was the kernel. Enter a Finnish student by the name of Linus Torvalds. In 1990, Stallman had given a speech in Helsinki, which Torvalds had attended. So in 1991, when Torvalds wrote a first implementation of a Unix-like kernel to run on his IBM PC, he used the GNU utilities. Torvalds decided to license his kernel under Stallman's GPL in order to give something back to the community whose work he had depended upon. Without the GNU replacement for the Unix utilities, Torvalds' kernel by itself would have been of little interest.

Off the top of my head, I'd guess that early distributions of Linux (by which I mean the complete system, not just the kernel) were no more than 10% Torvalds' kernel, the remaining 90% being GNU and X Windows code. There is now a huge collection of software licensed under either the GPL or other copyleft-compatible licenses. The GNU project started the whole thing, and tens of thousands of programmers world-wide contributed their time and effort freely, thus adding countless useful tools. But the media seems to prefer to tell stories about brilliant whiz-kids who invent something single-handedly, rather than stories about huge bodies of work created by innumerable participants. Thus "Linux" (a contraction of "Linus' Unix") has come to be applied to the entire body of work. And far too many people, including some programmers who ought to know better, believe that everything in Linux was written by Linus Torvalds.

If it were not for Stallman's vision and single-mindedness in striving for that vision, all software today would be closed-source. Stallman uses the term "Free Software" to emphasize the freedom software recipients have when software providers distribute source code with their software. Unfortunately, the double meaning of the word "free" means that he constantly has to explain that he means "free" as in "free speech," not as in "free beer."

By the mid- to late-1990s, programmers knew that this collection of software was far more reliable than proprietary software. But corporate decision-makers generally lack a technical background, and they are easily scared off by the term

"free". They wonder how good something can possibly be if it's free, and reject it in favor of a less reliable proprietary solution. The confusion of "free beer" vs. "free speech" was impeding the adoption of "free (as in freedom) software".

In 1998, a group of interested software developers decided to start publicizing the body of software under the name "open source" instead of "free software". They have been very successful in this effort. Although Stallman objects that placing the emphasis on pragmatism de-emphasizes the ideals he's striving for, most people today know the software under the name "open source."

But whether you call it "free software" or "open source," the concept has ramifications far beyond the software world. There are many other forms of creative work besides programming. The GPL and other similar licenses are designed specifically with software in mind. The first non-software application the GNU project encountered was technical documentation. So they came up with the GNU Free Documentation License (http://www.gnu.org/licenses/fdl.html). This license is applicable to written works that are not technical documentation. When Sam Williams' biography of Richard Stallman, "Free as in Freedom," was published in 2002, it was a copyrighted work licensed under the terms of the GFDL. The worldwide collaborative encyclopedia known as Wikipedia (http://www.wikipedia.org/) is also licensed under the GFDL. Although calling works like this "open source" is something of a misnomer, as the distributed content usually is the "source" itself, the name has stuck.

In the past few years, an organization known as Creative Commons has come into existence under the leadership of Stanford law professor Lawrence Lessig and others. At their website (http://creativecommons.org/), you can choose a set of conditions for distribution of your work, and they'll generate a license for you which expresses those conditions in appropriate legal language. Some combinations of conditions are intended to comply with open source licensing; some are not. In their FAQ, they state, "Note that no Creative Commons license has been certified as open source by the Open Source Initiative and any license containing the NonCommercial or NoDerivatives properties definitely does not qualify as open source, as they violate the Open Source Definition's No Discrimination Against Endeavor and Derived Works criteria respectively."

Although the future is impossible to predict, it seems certain that proprietary companies will continue trying to lock up everything in sight, and the Open Source movement will continue to protect the public interest by creating and defending innovative forms of licensing.