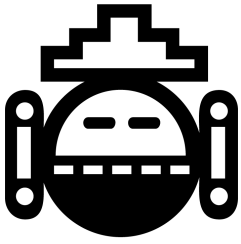




SoftLaw



Larry Graham

Legal Implications of Operating Systems

Though I don't expect much sympathy, Linux does nothing for me. Sure, it offers many technical advantages, and you can't beat the price, but it provides practically no opportunities for litigation—how can you sue others for software piracy when the software is given away in the first place? As an intellectual-property lawyer, I'd have to look for another line of work if everyone gave their products away and disclaimed all rights under the patent and copyright laws.

FREE VERSUS OPEN

Setting my job security aside, Linux is a wonderful product of the legal system. Its many technical attributes notwithstanding, perhaps its most significant benefit is its freedom from legal claims. While some have referred to Linux as "open" software, it is important to distinguish between the terms "free" and "open." "Open" is most often associated with "open architecture," referring to a set of hardware or software specifications that anyone may use to develop products according to those specifications. Frequently industry or government organizations produce open architectures to try to facilitate standardization and interoperability. While "open" refers to the right to use a set of specifications, "free" more narrowly refers to the right to use a single program. A "free" program typically comes with an unrestricted ability to use, copy, modify, and distribute the software without fear of violating the author's intellectual-property rights. In many cases, "free" means that the software is in the public domain and in this sense refers to freedom from intellectual-property claims, rather than the absence of payment requirements. You may have to purchase your physical copy of the "free" software.

The Open Source Initiative (OSI) recently blurred the distinction between these two terms by coining the term "open source" to refer to software that is essentially the same as "free" software. Although not intended to be limited to "open source code," access to source code is an important element of open-source software. Because OSI is using the term "open source" as a certification mark, the term "free" should probably be used for software that generally has the attributes of free software, reserving the term "open source" for programs authorized by OSI to use that designation.

Both "free" and "open" software are relatively new phenomena. As recently as the early 1970s, software was commonly believed to be unprotected by intellectual-property laws. Prior to 1976, there was considerable doubt whether software was a "work of authorship" that could qualify for protection under the copyright laws. Likewise, the US Patent and Trademark Office and the courts generally disapproved of software patents until the early 1990s.

OPERATING SYSTEMS AND COPYRIGHTS

In 1976, the Copyright Act was amended to extend copyright protection to computer programs by defining a computer program as a "set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result." Nevertheless, as recently as the early 1980s many believed that certain categories of software, including operating systems, would not be protected under the copyright laws. Perhaps more than other types of software, operating systems are functional, residing at the computer's core and performing much of the "heavy lifting" transparent to

EDITOR: Larry Graham • Christensen O'Connor Johnson & Kindness • graham@cojk.com



the user. Because the copyright laws protect only the expressive, nonfunctional aspects of software and other works of authorship, many believed that operating systems were "too functional" to fit under copyright laws.

In the early 1980s, Franklin Computer Corporation apparently believed that operating systems could not be protected under the copyright laws when it copied the Apple II operating system. Apple sued Franklin and easily proved that Franklin had copied its operating system: Franklin's master disk holding the Ace 100 operating system contained the word "AppleSoft" and the name of James Huston, an Apple systems programmer who wrote much of the Apple II operating system. Although the federal trial court in Philadelphia ruled in Franklin's favor, the Federal Court of Appeals reversed the trial court's decision in 1983, concluding that operating systems were protectable under the 1976 and 1980 amendments to the Copyright Act.

During the early 1980s, many other court decisions further defined the scope of copyright protection for computer software. For the most part, the courts favored programmers, extending copyright protection to virtually all aspects of computer software: operating systems; object and source code; micro-code; program structure, sequence, and organization; and look-and-feel. The courts also broadly interpreted the strength of copyright protection, severely limiting competitors' ability to evaluate software when developing competing programs.

DEVELOPING LINUX

Several organizations formed to lobby against this trend of increasing legal protection for software. Others organized to develop a body of free software. While these groups have failed to reduce or eliminate legal protection for software, they have made respectable progress in producing a library of free software, including Linux.

Leading the charge for free software is the Free Software Foundation. Richard Stallman founded FSF in 1983 to develop software that could be freely exchanged without restrictions on the ability to make copies or modifications. In 1984, FSF began the GNU project to develop a complete, free software system that would be upwardly compatible with Unix. The GNU system (GNU, pronounced with a hard "G", stands for "GNU's Not Unix") is intended to be free. By "free," FSF means users enjoy three specific freedoms:

1. to copy the program and give it away,
2. to modify the program by having full access to the source code, and
3. to distribute any modified versions of the software.

"Free" expressly does not refer to price. FSF and other distributors will gladly take your money for copies of their free software.

Working with donated equipment, financial contributions, and voluntary programming assistance, by the 1990s FSF had either found or written all the major components of its Unix-like operating system except the kernel. FSF is still working on its kernel, called Hurd; meanwhile, Linus Torvalds developed his kernel, called Linux. When Linux was combined with other free components, including much of the GNU system, it resulted in a complete Unix-like operating system.

The GNU software, which comprises about 28 percent of a typical Linux operating system, is distributed under what FSF calls the General Public License. The GPL takes advantage of copyright laws to ensure that free software remains free. Under the GPL, anyone may freely copy, distribute, and modify it. More important, the GPL requires that any subsequent distributions, including distributions of modified software, must also be made under the terms of the GPL. Accordingly, when others distribute modifications of free software, these are also free. The GPL itself may not be modified by adding or subtracting terms.



Boston University Department of Electrical and Computer Engineering

If you have significant academic or industrial experience in developing large software systems, and you are committed to improving the practice of software engineering, we want to talk to you about a faculty appointment at Boston University in Computer Systems Engineering. Our graduate program teaches the engineering skills necessary for the effective development of large-scale computer systems in which software provides essential functionality. We teach students to apply engineering principles to the design of a full range of computer products, from embedded systems, to data communication networks, to software products.

Three types of appointments are available in the Department of Electrical and Computer Engineering starting in September 1999:

- Research oriented tenure-track and tenured appointments.
- Non-tenure-track positions, which require extensive experience in practicing software engineering.
- Adjunct (part-time) positions for Boston-area experts who are interested in teaching their specialty at the graduate level.

All positions require a commitment to excellence in teaching at the undergraduate and graduate levels.

To learn more about opportunities for a non-tenure track OR adjunct positions, please e-mail: eceseach@ecenet.bu.edu and a faculty member will call to discuss our opportunities. For tenure-track OR tenured appointments, send your Curriculum Vita to: Professor Bahaa E.A. Saleh, Chair, Department of Electrical and Computer Engineering, Boston University, 8 St. Mary's Street, Boston, MA 02215

**Boston University is an Equal
Opportunity/Affirmative Action
employer.**



I am excited to resume this legal column after more than two years' hiatus. Many of today's front-page headlines come squarely from the intersection of software and law. The Microsoft antitrust lawsuit, the Java litigation, and the Y2K problem—just to name a few—give us plenty of issues to sort out.

I hope to accomplish three principal objectives in this column. First, I will aim to bridge the gap between software and the law. In many (or perhaps most) cases, the law is a sketchy and hazy collection of statutes and court decisions that can't be deciphered without a key. This column will address legal issues affecting the software industry in a manner that cuts through the legalese as much as possible.

Second, this space will provide general tips regarding sound legal practices in the software industry. Because the laws affecting the software industry evolve constantly, those who pay close attention will be best positioned to use them to their advantage—and more likely to avoid catastrophic consequences.

Finally, this column will offer a lawyer's-eye view of many of today's hot issues. Some important recent developments affect, for example, Internet domain names, database protection, shrink-wrap licensing, the year 2000 problem, and many others. From time to time, this column will report on major developments in an effort to decipher the legal reasoning behind them and address how they will affect the industry.

Because this column is intended to serve the readers, I openly solicit your feedback and suggestions for future issues. I may decline to fill a column with the best lawyer jokes I know, but any other legal topic of interest is fair game. I hope you will find the articles in this column to be interesting, understandable, and useful.

—Larry Graham, *SoftLaw* editor

Countless examples exist of patents that prevent others from selling software without patent infringement. One notable example of a patent that took the industry by surprise is Compton's multimedia patent. If it had not been invalidated by the patent office, virtually all multimedia products would have been subject to royalties under Compton's patent. In another example, certain Unisys and IBM patents constitute roadblocks for the free software community. Unisys and IBM hold patents covering the LZW compression algorithm, which is used for making GIF files, hence to sell software capable of making GIF files requires a license. Consequently, FSF has been unable to create free software capable of making GIF files.

As the other articles in this issue attest, Linux is laudable for its features and technical attributes. But Linux is also a legal marvel. Using donated time, money, and equipment, programmers developed a complete operating system that is free in the legal sense. Whatever your personal opinions regard-

ing intellectual-property laws (and I'll admit I'm biased), Linux should be applauded. Although I—and a lot of programmers—would have to find another line of work if there were no laws preventing software piracy, a few more programs like Linux wouldn't hurt. ❖

PATENT CONCERNS

It isn't always easy—or possible—to create software that is certain to be free of legal claims. Because copyright infringement requires copying, a noninfringing work can be produced simply by avoiding any access to other copyrighted software.

Avoiding patent infringement is not so easy. One infringes a patent by making, using, offering for sale, or selling a product that is within the scope of the patent's claims, regardless of whether there was any awareness of the patent. Worse yet, a patent may issue tomorrow from an application filed several years ago—software that is free today may infringe a patent tomorrow. Thus, while it is possible to produce software that is free from copyright infringement, it is not always possible to create software that is free from patent infringement.

Larry Graham practices intellectual property law as an attorney at the law firm of Christensen, O'Connor, Johnson & Kindness in Seattle, Washington. Contact Graham at graham@cojk.com.