J. Webster (ed.), *Wiley Encyclopedia of Electrical and Electronics Engineering*

FREE AND OPEN SOURCE SOFTWARE

The word *free*, used in connection with software, has two entirely different meanings. On one hand, it means free of payment, or without charge—for example, gratis software or freeware. On the other, it implies freedoms (liberties) associated with social, political, and religious agendas—for example, free or open source software.(1). This article is concerned with software belonging to the latter rather than the former category.

The availability of an application's source code is the fundamental value behind the free and open source movement. Source code, in contrast with machine or object binary code, is the collection of programming statements that adhere to the structure and syntax of a formal language, such as C, C++, Pascal, Cobol, or Fortran. In most cases, it can be easily viewed and edited with a text editor before being submitted to a compiler for conversion to machine or object code. It is the machine or object code that is actually read and interpreted by a computer, but it is not easily viewed or edited.

This article introduces the philosophy and practice of the free and open source software movements. It begins with a brief history behind the movements, and then discusses key intellectual property and licensing concepts that are central to software development. Some well-known free and open source applications are described and followed by a brief discussion on the merits of the movement and criteria for evaluating it. The article concludes by referring the reader to highly regarded Web pages for more in-depth and up-to-date coverage of the free and open source software movement.

Historical Background

The concept of free software is anything but new. In fact, there has been free software as long as computers have been in existence. Before entrepreneurs saw the business opportunities for information technology, engineering and computer scientists in academia were collaborating and sharing their programming knowledge with others at conferences, symposia, and society meetings. As personal computers like the Apple II, Commodore 64, Kaypro, Tandy, and Timex Sinclair surfaced in the 1970s, programming became a hobby for enthusiasts who manually entered example programs that appeared in vendor documentation and magazines (2).

By the time the personal computer appeared in the 1970s, computer scientists, engineers, and students from academia and industry had been collaborating for about ten years on technologies like artificial intelligence, ARPAnet (the predecessor to the Internet), Unix, and the C programming language. By 1974, Unix had been ported to several different computing platforms, and its user population was growing.

By the early 1980s, however, things changed dramatically. First, PC users observed the mass commercialization of the industry. Second, the Unix community faced the uncertainty of an operating system becoming proprietary. Finally, the ARPAnet and artificial intelligence developers witnessed the decline of an aging computing platform and migration of users to other endeavors (3).

The individual most credited for pioneering the concept of free software is Richard Stallman. Dismayed by being denied access to the source code for a device driver while working at the Artificial Intelligence Lab at MIT, Stallman began writing free software. In 1985, he established the Free Software Foundation (*FSF*)

as a tax-exempt charity to support development of free software (4). The FSF (http://www.fsf.org/) continues to support the philosophy and practices of free software and maintains a wealth of resources through an impressive Web presence.

When he started writing free software in 1983, Richard Stallman's goal was to develop a free replacement for Unix. By 1985, Stallman's text editor, Emacs, had been ported to many minicomputers (5). Other tools and utilities that emulated their proprietary Unix counterparts followed. By 1994, the GNU project finally had a Unix-like kernel, with the release version 1.0 of Linux. Started as a hobby by Linus Torvalds, then a student from Helsinki, Linux is now one of the most widely ported operating systems for personal computers (6).

While the Free Software Foundation remained firm in its values of full disclosure of source code with software applications, others felt exceptions were necessary. They argued that certain cases warranted the packaging of some binary code without its corresponding source, such as combining open source with its own proprietary products (7). So the Open Source Initiative (http://www.opensource.org/) was established in 1998 as an advocate for both the commercial interests of legitimate software businesses and users of free software.

One of the earliest and most successful organized efforts to distribute free software was the Kermit project at Columbia University. The Kermit protocol was developed as a noncommercial utility to transfer files between computers (8). Released in 1981 for a handful of operating systems, Kermit is now available for virtually any platform. Interestingly, while many commercial vendors now tout the portability of their software, the Kermit project has been quietly active in this practice since its inception. Until recently, all Kermit distributions were freely available with source code and included liberal licensing policies that allowed copying and redistribution. In 1995, Kermit released its first commercial offering, Kermit 95, with more restrictive licensing terms. This move was made, for the most part, because its revenue stream was diminishing. Although the Kermit project is still funded in part by sale of its publications, its income from distributing software on tape and other media diminished as Internet availability increased during the 1990s.

Intellectual Property Issues

The liberties associated with free and open source are centered on the concept of software as intellectual property. The application of intellectual property laws (copyright and trademark) to software parallels the historical events of the free and open source movements outlined earlier. For instance, software generally did not qualify as works of authorship under U.S. copyright laws until 1976. Furthermore, the U.S. Patent and Trademark Office did not regularly approve applications for computer programs until the early 1990s (9).

Copyright. Copyright protects the intellectual property of the author by restricting the duplication, modification, and redistribution of a work, whether it is a computer program, musical composition, work of art, or literary composition. Copying, modifying, and redistributing copyright works are allowable under *fair use* exceptions to the Copyright Act. Fair use is a broad term that covers nonprofit activities like personal use, study, scholarship, or research. Works that explicitly relinquish any copyright protection, or are produced by U.S. government employees as part of their regular duties, are considered in the *public domain*. Although morally and ethically irresponsible, it would be legally possible to sell software in the public domain as if one owned it (10). On the other hand, virtually all software labeled as free or open source is not in the public domain. The contrary is a common misconception that the Free Software Foundation and Open Source Initiative are quick to correct publicly.

Patents. Patents protect the disclosure of a creation and grant the creator sole rights to produce and put it on the market for a period of time (twenty years in the United States). Because they are designed to protect full disclosure, patents are generally deemed incompatible with the ideals of the free software movement. A noteworthy example of software protected by patent is the LZW compression algorithm used to create GIF images. Unisys holds the rights to the LZW patents, which expire in 2003. Until then, the Free Software Foundation cannot create software to generate GIF images for their Web pages because they would not be

legally permitted to include the full source code. Doing so would disclose the compression algorithm and be in violation of federal law.

Trademark. A trademark is a word, name, symbol, or device that is used in trade to indicate the source of the goods. It is also used to distinguish the goods of one entity from those of another. A servicemark is the same as a trademark except that it identifies and distinguishes the source of a service rather than a product. The terms "trademark" and "mark" are commonly used to refer to both trademarks and servicemarks. Trademark rights may be used to prevent others from using a confusingly similar mark. However, they do not prevent others from making the same goods or selling the same goods or services under a clearly different mark.

Licensing Software

While copyright, patents, and trademarks are used for protecting the intellectual properties as owners or creators, it is a license that dictates acceptable use, unless of course, a program is in the public domain. Virtually all software, whether it is available for free or fee, is licensed in some way or other. There is an abundance of license categories and confusing terminology. For example, software is sometimes classified as proprietary or nonproprietary; and it may be licensed as freeware, shareware, commercial, GNU Public License (*GPL*), or open source. Not everyone agrees on what distinguishes one category or license from another. Nonetheless, it is important to recognize and accept the terms of these licenses, whether one is a casual user, developer, or computer hobbyist.

This discussion will consider an application as *proprietary* if some or all of its corresponding source code is withheld from the distribution (11). The classic example of proprietary software is the commercial versions that are sold as shrink-wrap packages at mail order and retail outlets. Proprietary software, however, encompasses much more than the retail variety. In fact, most of the licensing categories that follow can be considered proprietary to some extent.

Shareware. During the early 1980s a different way of distributing software emerged that allowed users to try an application before actually paying for it. This "try before you buy" concept is more formally recognized as *shareware*. Virtually all shareware is copyrighted by its creators, and shareware is often distributed without corresponding source code. Hence, it is considered proprietary. As a rule, users have the opportunity to try a program for a definite period, usually thirty days, before payment is due. On the average, the fee is less than for commercial software because there is little or no marketing or distribution cost.

There is arguably a larger proportion of shareware for the Windows platform than for any other. Two common examples of shareware applications for Windows are the compression utility WinZip (http://www.winzip.com/) and the image editor PaintShop Pro (http://www.jasc.com/). These and other applications are popular downloads from the CNET (http://shareware.cnet.com/), Simtel (http://www.simtel.net/), and TUCOWS (http://www.tucows.com) Web sites.

Freeware. While free software normally implies freedom or liberty to run, change, and redistribute, there is a category of software, freeware, which is frequently and incorrectly confused with it. Rather than implying liberties or freedoms, freeware, for the most part, implies being free of cost. Typically freeware titles are proprietary, copyrighted, sometimes registered as a trademark, and distributed without source code. While traditional practice was to offer a less featured version without charge, the current trend is to support the free version with advertisements. The Eudora (http://www.eudora.com) email client and the Opera (http://www.opera.com/) Web browser are typical examples of freeware titles in circulation that have adopted this approach. Other titles like Pretty Good Privacy (http://www.pgp.com/products/freeware/default.asp), more commonly known as *PGP*, began as free software but then became proprietary, and continue to be available as freeware for noncommercial use. Some vendors are even offering free versions of their commercial office productivity suites. Corel WordPerfect 8 (http://linux.corel.com/products/wp8/download.htm) and Sun's StarOf-

fice (http://www.sun.com/staroffice/) are both available as free downloads. WordPerfect 8 is restricted to Linux, while StarOffice is available for Linux, Windows, and Solaris (Intel and SPARC).

GNU Public License. The guiding principle behind the Free Software Foundation is the GNU (pronounced "guh new") Public License (http://www.gnu.org/copyleft/gpl.html) a rather complex legal document based on four simple freedoms. First, it gives the user the freedom to run the software. Second, it gives the user the freedom to change the software to meet his or her individual needs. Third, it gives the user the freedom to redistribute the software to other potential users. Fourth, it gives the user the freedom to improve the software and redistribute the improvements to others. Unlike the commercial, shareware, and freeware models described earlier, the GPL requires that the source code be included with all distributions. This requirement is dubbed "copyleft," a gibe at the word "copyright" and the proprietary practices of others who restrict modifications by distributing only machine-readable code with distributions.

Open Source Licenses. While the GPL restricts the mixing of proprietary (without source code) with free software, the FSF is not opposed to building an enterprise around free software as long as the business does not use proprietary methods to hide its intellectual property. It argues that organizations and business can and do add value to free software by offering services like custom packing, redistribution, training, and support without stifling innovation and creativity. In fact, the FSF endorses several organizations like the Linux distributor Debian (http://www.debian.org) and the X-Windows development organization, X.org (http://www.x.org/).

While the FSF objects to the commercialization of software, others argue that its ideals are, ironically, too restrictive. They reason that mixing proprietary with free software (with source code) allows an application to reach a much wider user population (12). The Open Source Initiative (*OSI*) supports these efforts to provide semifree software and goes so far as to certify several derivatives of its Open Source Definition (http://www.opensource.org/docs/definition.html) as acceptable open source licensing models. Software vendors who participate in the open source model recognize the value of software improvement and innovation through sharing and collaboration, yet recover their investment and protect their intellectual property (13). Not surprising, the FSF views most of these licensing models (e.g., the Apache and Mozilla Public Licenses) as incompatible with its GPL (14).

Free and Open Source Distributions

The ensuing discussion will highlight several free or open source titles available for download. The reader will soon discover that a complete desktop or server environment can be built around free or open source software. Most of the titles are licensed under the GPL or certified by the Open Source Initiative. The list is not, by any means, meant to be comprehensive. Rather it is meant to illustrate the fact that a collection of free or open source software can be obtained without the payment and restrictions commonly associated with their proprietary counterparts.

Linux. The foundation of any computer system, in terms of software, is the operating system. Distributed under the GNU General Public License, Linux is a Unix-like operating system that includes true multitasking, virtual memory, shared libraries, demand loading, shared copy-on-write executables, proper memory management, and TCP/IP networking. Although Linux was initially developed for the Intel 386/486 PC, it has been successfully ported to virtually all architectures, including DEC Alphas, Sun SPARC, M68000 machines (like the Atari and Amiga), MIPS, and PowerPC. Numerous companies have developed their own distributions of the Linux kernel and not only sell them but also make them available for download without charge. A listing of these distributions is available from Linux Online (http://www.linux.org/dist/ftp.html). Before the Linux kernel surfaced in 1994, clones of Unix were developed and continue to be available, albeit on a much smaller scale. These include FreeBSD (http://www.freebsd.org/), OpenBSD (http://www.openbsd.org/), and NetBSD (http://www.netbsd.org/).

Apache. The software serving more than half of the Web pages on the Internet is not a commercial product, but rather an open source application named Apache HTTP Server (http://httpd.apache.org/). Apache HTTP Server started out in 1995 as a collection of extensions and bug fixes ("patches") to the National Center for Supercomputing Applications (*NCSA*) HTTP daemon, which was in the public domain and not being maintained. The Apache License is certified by the OSI but deemed incompatible with the GPL. The software is a free download for virtually any operating system. It is also the foundation for IBM's commercial HTTP daemon, WebSphere.

LaTeX. Scientific and technical documents that use formulas and character sets requiring high-quality typesetting can be difficult or impossible to create with a word processor. LaTeX (pronounced "lay tech") is a "document preparation system" that uses a markup language to emphasis content over form. It is widely used by authors and publishers in the mathematics, engineering, chemistry, and physical sciences. Documents are formatted as device-independent (*DVI*) pages and can be easily translated to PostScript or portable document format (*PDF*). LaTeX (http://www.latex-project.org/) is distributed with source code and at no charge under the LaTeX Project Public License (*LPPL*). Like Apache, it is an open license compatible with the OSI, but incompatible with the GPL because of restrictions on distributing modified versions of the program.

Perl. Most of the server-side scripting or Common Gateway Interface (*CGI*) functionality of the Web began and continues to be developed with Perl (Practical Extraction and Reporting Language). Perl, like many other scripting applications, is an interpreted language. Perl is licensed as free software and is compatible with OSI and GPL. It is available as a free download from the Comprehensive Perl Archive Network (http://www.cpan.org/).

Mozilla. In 1998, Netscape announced plans to introduce an open source version of its browser. Shortly after the announcement, it began releasing its source code for the Navigator browser. The open source version was named Mozilla, the original code name for Navigator and, later, Communicator. Mozilla is described as an open-source Web browser, designed for standards compliance, performance, and portability. Its proprietary browser, Communicator, is based on the Mozilla source code. The Mozilla browser (http://www.mozilla.org/) is available as a free download in both binary and source code versions. It is distributed as open source under the Mozilla Public License (*MPL*), which is deemed compatible with OSI, but incompatible with GPL because it does not permit part of the program which might be licensed under MPL to be linked to another licensed under GPL.

Ghostscript. Many documents on the Internet are formatted as Adobe PostScript or PDF files. Postscript is a language embedded in some, but not all, printers. AFPL (formerly Aladdin) Ghostscript (http://www.cs.wisc.edu/~ghost/) is an interpreter that accepts the PostScript language as input and displays the results on a monitor or creates a PDF file as output. It will also print a Postscript file to a non-Postscript printer. The newest version is distributed under the Aladdin Ghostscript Free Public License for noncommercial use only. Early releases or versions are distributed under the GPL as GNU Ghostscript.

The Free–Open-Source Debate

The free and open source community has its share of enthusiastic supporters. These virtual communities of highly talented individuals proudly and collectively contribute to the development of high-quality free software. The free and open source movements offer models of success in distributed software engineering. Ljungberg (7) highlights the cases of Linux and Apache by pointing out that both have demonstrated new ways of sharing knowledge, managing projects, creating organizations, and doing business. Jagielski (15), in comparing a few open source applications against their commercial counterparts, cites price (free), community support, stability, robustness, extensibility, and adherence to standards as supportive arguments. Fowler (2) describes open source as a process that benefits from endless testing and innovation rather than being stifled by strict deadlines and profit margins.

The free and open source movement also has its detractors. Edwards (16) complains that users of free and open source software often experience poor interfaces, difficult installation, and no vendor-backed support. Wilson (17) argues that too much emphasis is placed on coding, compiling, and debugging rather than established development fundamentals like planning, designing, and implementation. McGraw (18) disputes claims from the open source community that its software is more secure than what is available from the commercial sector. Murphy (11) argues that the open source movement has shifted from a set of principles based on developing software independent of a commercial community to building products that blend with for-profit offerings. In doing so, the movement's guiding philosophy has transformed from one established on engineering principles to another based on business and marketing policies. Fausett (14) contends that, unlike public domain software, there are strings attached to the GPL copyleft policy stipulating that if developers take they must give back to the larger community.

Evaluating Free and Open Source Software

The availability of free and open source software provides users with not only a source of quality software at little or no cost, but also a community for collaboration. The inclusion of source code provides the means for innovation, endless testing, and continuous improvement. While the inclusion of source code distinguishes free and open source software from its commercial or proprietary alternatives, similar criteria can be applied to evaluate an application's quality or suitability. These include, but are not limited to, needs assessment, availability, requirements, ease of installation, comparison with similar products, documentation, and support.

Needs Assessment. Assessing the need for an application is a fundamental step in evaluating software. Frequently this involves identifying a problem and determining whether an application of technology will lead to an appropriate resolution. The need might be as simple as selecting an operating system, such as Linux, FreeBSD, OpenBSD, or NetBSD, for a computer acquired without one included. On the other hand, there may be a more complex need for managing Web pages that require frequent and ongoing content changes. In this case, one would select several applications, including a Web server (Apache), a scripting language (Perl or PHP), and a database application (MySQL).

Availability. Once the requirements are defined, a next step involves determining what applications are available. While mainstream magazines and books target the commercial and shareware industry with reviews and advertisements, they tend to overlook the free and open source alternatives. Fortunately, there are a variety of Web pages that offer listings of free software. The GNU Free Software Directory (http://www.gnu.org/directory/index.html) provides a complete listing of software distributed under the GPL and organized by broad classification: system, development, libraries, utilities, games, and so on. The GNU project also provides a Links to Other Free Software Sites (http://www.gnu.org/links/links.html) page that includes references to sources for GNU software ported to other operating systems.

Requirements. A critical step in evaluating software is determining what its requirements are. Software requirements should be explicitly specified in terms of memory (RAM and video), storage space (hard disk and removable media), processor (80386, 80486, Pentium, etc.), operating system (Linux, FreeBSD, Windows, Mac OS X), networking (Ethernet, modem, etc.), and multimedia (sound and video).

Ease of Installation. Although more difficult to determine than its requirements, an application's ease of installation is nonetheless important. There are several ways to assess the difficulty of installation. One is to consider the opinions of others expressed on Usenet newsgroups, Web pages, or published magazine reviews. Alternatively, the reputation of the developer or supplier may help. Perhaps the developer or supplier has produced other applications that installed with little or no trouble.

Product Comparison. In most cases where an application of technology is warranted for solving a problem, there are a variety of competing solutions. In an organization that dictates what it acquires and supports, the decision is somewhat less complicated. In that case, an organization's information technology

environment is often structured around a particular processor and operating system platform or suite of products offered by a particular vendor. Consequently, a comparison with similar products may be less complicated. In a less structured or heterogeneous environment where decision making is decentralized, it is especially important to make a comparison with similar products. As a result, the comparison can be rather difficult. For example, there are more than fifty businesses selling prepackaged versions of Linux, and numerous other sites that provide free downloads of that operating system. In this particular case, the Linux Distributions page (http://www.linux.org/dist/index.html) provides guidance in selecting a particular derivative, according to the language and according to its availability as a free download, as a packaged CD-ROM, or preinstalled by the vendor.

Documentation. An application can be virtually worthless without accompanying documentation. Conventional documentation from packaged software sold by retailers or value-added sellers is normally provided in a printed user manual. With downloaded software, whether it is shareware, freeware, free, or open source, the documentation often accompanies the distribution as plain text (ASCII), PostScript, or PDF files. In each case, the quality and quantity of documentation can be quite comprehensive or rather sparse. At the very minimum, the documentation should be of sufficient quantity and quality in providing installation, operational, and troubleshooting procedures.

Support. Ongoing support after installation is sometimes overlooked when evaluating software. An application is sometimes only as good as the level of support provided. Often support comes from users who post questions and answers to a Usenet newsgroup, e-mail discussion list, or Web page. This is particularly true for free and open source applications, whose developers or users provide needed support. There are literally hundreds of Usenet newsgroups and thousands of Web pages devoted to GNU software in general and Linux in particular. There are more than 50 Usenet newsgroups covering GNU software like Emacs and GCC, and 150 covering Linux. These are respectively located under the gnu.∗ and linux.∗ hierarchies of Usenet newsgroups. The Web pages cited earlier in the discussion regarding specific free and open source distributions generally include information for common support issues. Many also include links to other sources for more specific troubleshooting problems.

Conclusion

In this article the concepts of the free and open source movement have been introduced. While general in nature, it has emphasized the principles behind the free and open source movement of promoting and advocating the freedoms to use, modify, and redistribute software without restraint. It has offered a brief history of the movement, introduced some terms and concepts associated with intellectual property, highlighted a few widely known free or open source applications, analyzed the pros and cons, and provided evaluation techniques.

There are numerous Web pages that offer more intensive and current coverage of the topic. Several operating under the auspices of the Open Source Development Network (http://www.osdn.org/) that deserve special attention include Advogato (http://www.advogato.org/), Slashdot (http://www.slashdot.org/), Fresh Meat (http://www.freshmeat.net), and SourceForge (http://sourceforge.net/). Advogato serves as a community resource for free software developers. The site includes a comprehensive list of free software projects with corresponding links to their Web and developer pages. Slashdot is a respected source for free and open source events and news. Freshmeat is considered the authoritative site for announcing software releases. SourceForge, which offers a variety of free resources for free and open source developers, is a good source for keeping track of such releases.

BIBLIOGRAPHY

1. R. Stallman The importance of free software, *Automatisierungstechn. Praxis*, **43** (1): 20–27, 2001.
2. D. Fowler Open season [will open-source software finally take the commercial world by storm?], *Networker*, **4** (2): 18–25, 2000.
3. E. S. Raymond A brief history of hackerdom, in C. DiBona, S. Ockman, and M. Stone (eds.), *Open Sources: Voices from the Open Source Revolution*, Sebastopol, CA: O'Reilly, 1999, pp. 19–29.
4. R. Stallman The GNU operating system and the free software movement, in C. DiBona, S. Ockman, and M. Stone (eds.), *Open Sources: Voices from the Open Source Revolution*, Sebastopol, CA: O'Reilly, 1999, pp. 53–70.
5. J. Woehr What's GNU?, *Embedded Syst. Programm.*, *7* (1): 70–72, 74, 1994.
6. L. Torvalds The Linux edge, in C. DiBona, S. Ockman, and M. Stone (eds.), *Open Sources: Voices from the Open Source Revolution*, Sebastopol, CA: O'Reilly, 1999, pp. 101–111.
7. J. Ljungberg Open source movements as a model for organising, *Eur. J. Inf. Syst.*, **9** (4): 208–216, 2000.
8. F. Da Cruz *Kermit, a File Transfer Protocol*, Bedford, MA : Digital Press, 1987.
9. L. Graham Legal implications of operating systems, *IEEE Softw.*, **16** (1): 20–22, 1999.
10. D. Fiedler Free software!, *Byte*, **15** (6): 97, 100, 1990.
11. N. Murphy Open source point/counterpoint: Are open source and innovation compatible?, *Embedded Syst. Programm.*, **13** (10): 78–86, 2000.
12. J. Ousterhout Free software needs profit, *Comm. ACM*, **42** (4): 44–45, 1999.
13. D. K. Rosenberg "Business issues in free software licensing," *Proc. FREENIX Track*, 1999 USENIX Annual Technical Conference, Monterey, CA, 1999.
14. B. Fausett Getting it together, *WEB Techniques*, **6** (1): 18–20, 2001.
15. J. Jagielski Open source: Breaking through the hype, *WEB Techniques*, **6** (1): 40–43, 2001.
16. J. Edwards The changing face of freeware, *Computer*, **31** (10): 11–13, 1998.
17. G. Wilson Is the open-source community setting a bad example? *IEEE Softw.*, **16** (1): 23–25, 1999.
18. G. McGraw "Will openish source really improve security?" *Proc. 2000 IEEE Symp. on Security and Privacy. S&P 2000*, Berkeley, CA, 2000.

ERIC P. DELOZIER
Penn State Harrisburg