The Benefits of Free/Open-Source Software

Daniel Brodzik

Written Communications

Name of instructor withheld

The Benefits of Free/Open-Source Software

How would you like to be free to do whatever you want with your computer software? How would you like a stable and secure operating system? Would you like to be able to copy and distribute your software without restriction? If you are a programmer, would you like to be able to study, modify, and adapt your software?

Free software, also known as open-source software, is a great idea if you said "yes" to any of the above questions. Too few people know that there are many viable alternatives to the software "standards" of Microsoft Windows and Microsoft Office. In this paper, I will go over definitions of terms you should be aware of, the free software and open-source definitions, the problems with proprietary software, like Microsoft software, and the benefits of free/open-source software. There is a real problem with companies like Microsoft who try to hijack the future of technology and turn it into a mechanism for control of information.

Glossary

- License agreement: This is a legal document that you must agree to before you can use a particular piece of software. This sets forth the terms and conditions for use, distribution, and modification for a piece of software. Most people ignore this when installing or compiling a program. Every piece of software except public-domain software has one.
- Source code and Binary: These are the two forms programs may be distributed in. *Source code* is human-readable text that defines the program. This is the preferred form for studying and modification. *Binary executables* are usually the form needed to run the program, but there are exceptions to this. Binaries *cannot* be readily understood by humans, and thus they are the form of the program that most software companies distribute the program in. In addition, programs that are distributed in binary form only almost always come with a license

that makes trying to discover the inner workings of the program using the binary illegal. To better understand this, we can relate a program to a car. The source code can be a book and a set of parts needed to build a car. The book would be very detailed and describe exactly how to build the car and how it works. The binary could be the car itself. If car manufacturers were like Microsoft, they would require you to sign an agreement stating that you won't open the hood for any reason whatsoever and then weld or lock the hood shut. This is one of the biggest problems with binary-only software.

- Proprietary software: This is software that cannot be studied, copied onto a number of computers greater than what the license dictates, or modified for any reason whatsoever. In addition, users are not allowed to use the software without paying a licensing fee. Proprietary software is distributed in binary form only.
- Non-disclosure agreement (NDA): This is a legal agreement stating that you will not share a certain piece of information with anyone else.
- Free software: Free software is free in terms of *freedom*, not necessarily price, which makes commercial free software possible. Don't confuse it with *freeware* or *shareware*, which are both like proprietary software except that they don't have copying restrictions. Richard Stallman (2004) says "Free software is a matter of the users' freedom to run, copy, distribute, study, change and improve the software." (para. 3) The software may be distributed with or without charge, whether the distributor paid for the software or not. Advocates of this term emphasize the ethical benefits of this type of licensing.
- Open-source software: This is essentially the same thing as free software in definition, but advocates of this term tend to emphasize the technical benefits of software that is licensed this way. This term is advocated by Eric Raymond and was coined to prevent companies that may

be scared away by the term "free software" due to its ambiguous nature. However, the term "open-source" is *also* ambiguous; it can either refer to the complete freedoms associated with free software, or it can refer exclusively to the right to see the source code, whether or not you can do anything else with it. I tend to advocate both causes because I believe that it is important to emphasize the ethical benefits (free software) as well as the technical benefits (open-source software). Thus, I will be calling it "free/open-source software" for the remainder of this paper.

Problems with Proprietary Software

There are many problems with proprietary software: the corporate virtual monopoly effect, not being sure of the safety of proprietary software, overcharging, unfair license agreements that are getting worse, the lock-in effect, and nobody outside the company being able to see the source code to the software. This section will describe these problems in detail.

First, there is the corporate virtual monopoly effect as demonstrated by Microsoft. Microsoft gets away with so many things because so many people are too willing to accept the myth that Microsoft is a monopoly. The lock-in effect is often the biggest reason people are forced into staying with a particular program. This occurs most often with Microsoft Word. For example, the Word document format is kept a secret by Microsoft. Many people transfer documents in this file format, and Microsoft is using this to their advantage. As soon as other programs, like OpenOffice.org, support the latest version of the Word format, Microsoft changes it again. Microsoft introduces many new, usually buggy, features into the latest version of Microsoft Office, tempting people to upgrade. Those who have an older version of Word must upgrade to read documents from people who are using the newest version. Those using very old versions of Word must also upgrade in order to send documents to people with newer versions of

Word. It's a vicious cycle. Stallman (2001) mentions a similar tactic that's also employed by Microsoft:

> When Microsoft does something new, its purpose is strategic--not to improve computing for its users, but to close off alternatives for them. Microsoft uses an anticompetitive strategy called "embrace and extend". This means they start with the technology others are using, add a minor wrinkle which is secret so that nobody else can imitate it, then use that secret wrinkle so that only Microsoft software can communicate with other Microsoft software. In some cases, this makes it hard for you to use a non-Microsoft program when others you work with use a Microsoft program. In other cases, this makes it hard for you to use a non-Microsoft program for job A if you use a Microsoft program for job B. Either way, "embrace and extend" magnifies the effect of Microsoft's market power. (para. 10-11)

Second, there is the problem with overcharging. Software companies like Microsoft use tactics to justify charging $150-300 for a single-user license copy of Windows or Microsoft Office and offering little or no support without the user paying even more. According to the Debian GNU/Linux distribution's "About Debian" page (2004), "software is not like making cars." If you make one car, it is costly. It will also cost a lot to make another one just like it. With software, it may cost some money to develop the software. With software, once the first copy is made, it doesn't cost a lot to make another copy. ("How can you give it away?" section, para. 1) I think that some reasons that Microsoft software is so costly off-the-shelf are greedy executives, the lawyers that are hired to defend Microsoft in their trials, the costs involved in implementing anti-piracy measures in their software, and all the other predatory business practices they engage in.

Third, there is a problem with license agreements that are unfair. They try to discourage the human instinct to share information. Some license agreements require activation within a certain amount time. This can introduce privacy problems. For example, when I was activating MS Office XP, I had to enter personal information. Some companies also introduce clauses that state that they can inspect your computer over the Internet without your consent. Also, if you are a home user with multiple computers, you usually have to buy as many copies of the software as computers you want to run the program on. How do companies get away with this? From what I've seen, the answer is that people don't *read* the license agreement and just click "I agree", thus making a legal agreement without fully understanding what they're doing. It wouldn't be a wise idea to sign a contract without reading it first, and, likewise, it wouldn't be a wise idea to agree to a license you haven't read. Stallman (2002a) has this to say:

> Proprietary software means, fundamentally, that you don't control what it does; you can't study the source code, or change it. It's not surprising that clever businessmen find ways to use their control to put you at a disadvantage. Microsoft has done this several times: one version of Windows was designed to report to Microsoft all the software on your hard disk; a recent 'security' upgrade in Windows Media Player required users to agree to new restrictions. But Microsoft is not alone: the KaZaa music-sharing software is designed so that KaZaa's business partner can rent out the use of your computer to their clients. These malicious features are often secret, but even once you know about them it is hard to remove them, since you don't have the source code. (para. 2)

The biggest problem with proprietary software is that people outside the company cannot see the software's source code or try to discover it from the binary. First, you can never be sure a piece of proprietary software is safe, as evidenced by the above paragraphs. Second, bugs are very hard to find and to fix, as demonstrated by Windows and MS Office. Third, new features are often hard to get. Fourth, the lack of the availability of source code limits the program's usefulness, meaning that others cannot modify and adapt the program.

Benefits of Free/Open-Source Software

Free/open-source software has many benefits, including the encouragement of sharing and modification, a very efficient development model, more reliable software, the advocacy of open standards, often low or nonexistent licensing costs, the fact that free/open-source programs are often not tied to one operating system or hardware platform, and better security. These benefits are very helpful in the development of quality software, and they are not possible with proprietary software licensing.

First, the permissive license agreements encourage the human instinct to share information by specifically allowing copying, distribution with or without charge, modification, and studying of the software. Many people already illegally copy and distribute proprietary software. Microsoft and a few other rich companies are already trying to find ways to prevent people from using software they acquired illegally. In other words, the companies that are complaining the most are the ones with the most money. According to Stallman (1994), these companies compute their losses by figuring that people who illegally copy the software would have paid for it. He believes that most people who illegally copy software most likely would not have paid for it anyway, so the companies aren't actually losing much except potential users. (para. 9) However, free/open-source software is licensed to specifically allow copying. In addition, the GNU General Public License and GNU Lesser General Public License, GNU GPL and GNU LGPL for short, two of the most popular licenses in use for free/open-source software,

are designed to prevent a distributor from adding restrictions on use, copying, modification, etc. Because of this, free/open-source software is a public good.

Second, the concept of free/open-source software has introduced a surprisingly efficient and effective development model. According to Eric Raymond (2002), the most effective development model is bazaar-style rather than cathedral-style. He "believed that the most important software (operating systems and really large tools like the Emacs programming editor) needed to be built like cathedrals, carefully crafted by individual wizards or small bands of mages working in splendid isolation, with no beta to be released before its time. Linus Torvalds's style of development--release early and often, delegate everything you can, be open to the point of promiscuity--came as a surprise. No quiet, reverent cathedral-building here--rather, the Linux community seemed to resemble a great babbling bazaar of differing agendas and approaches (aptly symbolized by the Linux archive sites, who'd take submissions from *anyone*) out of which a coherent and stable system could seemingly emerge only by a succession of miracles." (pp. 2-3) The rapid development of such important free/open-source projects as Linux, a fast and stable Unix-like operating system that I use every day, and OpenOffice.org, the very program I used to type this paper, is only getting better. According to Raymond (2002), Linus' Law is "Given enough eyeballs, all bugs are shallow." Linus Torvalds, the creator of Linux, invented the open development model. Torvalds has a lazy personality; this very quality of his contributed to this development model. According to Raymond (2002), "Linus seems to me to be a genius of engineering and implementation, with a sixth sense for avoiding bugs and development dead-ends and a true knack for finding the minimum-effort path from point A to point B. Indeed, the whole design of Linux breathes this quality and mirrors Linus's essentially conservative and simplifying design approach." For example, there are many bugs in Microsoft software. The only people allowed to work on and fix these bugs, or even to see the code, work at Microsoft. These people cannot possibly be as diversified as the high number of users of Windows and MS Office because the code is confined to the relatively small number of people at Microsoft. Linux and OpenOffice.org, on the other hand, are worked on by a lot of people around the world with more diversified computer experience, and there are ways to help even if you're a non-programmer, like filing bug reports, requesting features, donating, and promoting the software. The fact that anyone can see and work on the code means that anyone can proofread the code. The way I see it, it's exactly like getting someone else to proofread something you write; if the only people who can see the code are the original authors, flaws are a lot harder to find. Developers of free/open-source software are generally good at listening to their users, and their users can often file more useful bug reports due to the open nature of the code.

Free/open-source software can be developed commercially, but Microsoft doesn't seem to believe this is possible and even posted their criticism of the GNU GPL on their website. They state that the GPL is against the American Dream, even though Bradley Kuhn started his own consulting company based entirely on software under the GPL. (Kuhn, 2001, para. 8-9) They don't seem to realize that while the GPL was designed to keep other people from taking a program that is under the GPL and using the specific development model that Microsoft uses, there are many other ways of making money from software. For example, while almost all free software is available without charge on the Internet, there are many companies that sell the programs on CD. The distributor may even publish manuals and sell those. In the case of a full operating system like Debian GNU/Linux, it may even be more convenient to buy the program on CD than to download it because many people, including me, still have a slow Internet connection. For example, depending on where you buy a Linux distribution, you can get a nicely-packaged set of CD's with manuals and a support contract, or you can spend as little as $5-15 but get only the CD's and no support from the organization that made the distribution.

One major benefit of open-source software is that the open development model often ensures that the software can be useful on more than one operating system and hardware platform. Major free/open-source applications like OpenOffice.org and AbiWord are available for Linux, Macintosh, Windows, and BSD software platforms.

A big benefit of free/open-source software is security. We have all heard the news reports of security break-ins. There are many reasons that free/open-source software is secure. One of the most obvious to a programmer, even a very casual one such as myself, but actually commonly held as a potential problem by the average user is the very fact that the source code is available to anyone who wants it. One may think that the code being open to everyone would lead to *more* security problems, but the reverse is actually true. Consider this: Microsoft doesn't release the code for Windows. The only people who can get the information needed to do a security break are people who do it illegally, and Microsoft has sole control over fixing security problems. Nobody else has any control over it. In the case of Linux, for example, people have access to the source code. This usually means that security updates are released almost upon discovery. Stallman (2002a) writes, "If a free program has a malicious feature, other developers in the community will take it out, and you can use the corrected version." (para. 12)

The final, and most important, benefit of free/open-source software is that it is *real*, and it exists *today*. While all this probably seems too good to be true, it really isn't. This used to be the way *all* software was licensed until companies like Microsoft emerged. Stallman (2002b) writes, "In the 70s, computer users lost the freedoms to redistribute and change software because they didn't value their freedom. Computer users regained these freedoms in the 80s and 90s because a group of idealists, the GNU Project, believed that freedom is what makes a program better, and were willing to work for what we believed in." (para. 10) Now that this is becoming mainstream, the time is now right to spread the word.

Conclusion

The concept of sharing information is not new. However, even though software is technically information, major companies like Microsoft are treating it like a tangible object by requiring every user to buy a copy for each computer he or she owns. There are many reasons we should avoid the use of proprietary software, like the virtual monopoly effect, the overcharging, unfair license agreements, the lock-in effect, and the obscurity of the code. There are many reasons to use free/open-source software, including encouragement to share information, the effective development model, cross-platform compatibility, security, and the fact that it exists today. For further reading, visit http://www.fsf.org, http://www.opensource.org, http://www.linux.org, and http://www.debian.org.

References

About Debian. (2004). Retrieved November 30, 2004, from http://www.debian.org/intro/about.

Kuhn, Bradley. (2001). *The GNU GPL and the American Dream*. Retrieved November 30, 2004, from http://www.fsf.org/philosophy/gpl-american-dream.html.

Raymond, Eric. (2002). *The Cathedral and the Bazaar*. Retrieved November 30, 2004, from http://www.catb.org/~esr/writings/cathedral-bazaar.

Stallman, Richard. (1994). *Why Software Should Not Have Owners*. Retrieved November 15, 2004, from http://www.fsf.org/philosophy/why-free.html.

Stallman, Richard. (2001). *The GNU GPL and the American Way*. Retrieved November 30, 2004, from http://www.fsf.org/philosophy/gpl-american-way.html.

Stallman, Richard. (2002a). *Can you trust your computer?* Retrieved November 30, 2004, from http://www.fsf.org/philosophy/can-you-trust.html.

Stallman, Richard. (2002b). *Linux, GNU, and freedom*. Retrieved November 30, 2004, from http://www.fsf.org/philosophy/linux-gnu-freedom.html.

Stallman, Richard. (2004). *The Free Software Definition*. Retrieved November 30, 2004, from http://www.fsf.org/philosophy/free-sw.html.