Academia's Obligation to Software Freedom

The case for Free Software within educational institutions

By Danny Piccirillo

The Free Software Movement is revolutionizing the way software is made based on the ideals of freedom and openness. The sharing of software is as old as computers themselves, but now it is done in a concerted effort to completely transform the software world, and in fact has spread so much that the same principles are now being applied to content like writing, music, and other artwork, hardware designs, business models, journalism, school systems, and even politics and governance (Definition 2008). The Free Software Movement is built upon the free exchange of ideas, open sharing of knowledge, actualizing goals together, and working in a way that benefits the community as a whole. The parallels between this movement and the role of educational institutions in our society are clear. If schools are true to this philosophy, they have an obligation to be a part of the movement. Speaking with tech staff at Newton North High School and diving into the local Free Software community, I have found a number of ways to encourage and help schools to adopt Free Software.

The details of the inner workings of computers may be intimidating to most people, but to understand Free Software, only an easily-grasped, basic understanding of some technical jargon is necessary. These definitions are more than enough for one to be able to fully understand the philosophy of Free Software:

Hardware: The physical, tangible machine (hence, "*hard*"), components, and devices like mice, keyboards, monitors, printers, and the computer itself, are all examples of hardware. They are the things that either the software runs on, or allow you to interact with the software.

Software: Everything that runs *on* the machine. Microsoft Windows, Mozilla Firefox, and all other programs and operating systems are software.

Operating System (commonly abbreviated to *OS*): The interface between the hardware and user. Ubuntu, Mac OS, and Windows are popular examples.

Source Code: All software is made up of code written in programming languages. The code that makes up software is called the source code. Free Software leaves this source code open, as opposed to proprietary software, which keeps the source code secret. Imagine software as a slice of pie. If you can see, modify, and change the recipe, it's Free. If not, it's proprietary.

Copyleft: The ingenious method of using copyright law to prevent copyright restrictions. It grants permission to use, modify, and share, as long as derivatives preserve the same freedoms.

GNU (Gnu's Not Unix): Refers to the GNU Project, an effort to make a Free operating system based off of Unix, a style of operating systems, or to the GNU Operating System itself.

Linux Kernel: If operating systems, like Windows, were apples (the fruits), the kernel would be the core. Linux is the name of the core of the GNU/Linux Operating System.

GNU/Linux: Commonly called Linux, refers to the family of Linux-based distributions. Distributions are like different flavors of the GNU/Linux OS, the most popular being Ubuntu.

The Free Software Definition, first published in 1986 by Richard M. Stallman, founder of the Free Software Foundation, lists the four fundamental freedoms any piece of software must have to be considered Free:

- The freedom to run the program, for any purpose (freedom 0)
- The freedom to study how the program works, and adapt it to your needs (freedom 1)
- The freedom to redistribute copies so you can help your neighbor (freedom 2)

- The freedom to improve the program, and release your improvements to the public, so that the whole community benefits (freedom 3)

(Stallman 2002)

When unfamiliar people first hear the term Free Software, they usually assume that Free refers to price as in "free beer", but instead the "free" in Free Software refers to liberty as in "free speech" and "free market". Software that is available for no cost is called freeware, but isn't necessarily Free. Naming controversy has arisen around a parallel movement that chooses to use the term "open source" which doesn't regard freedom and only cites practical values. Sometimes the term Free and Open Source Software (FOSS) is used, to combine the terms. To resolve the ambiguity of the word "free", some replace it with "libre", which distinguishes it from gratis software (zero cost), and that has led to some using the term Free/Libre and Open Source Software (FLOSS). However one refers to it, it still means the same thing.

Those freedoms may seem very basic on the surface, but the consequences are huge. In order for these freedoms to exist, access to the source code is a necessary condition (Stallman 2002). Proprietary software keeps the source code secret, and restricts people from viewing, modifying, and sharing it. Vendors who produce such software make money directly from the software itself, and so they depend on keeping their control over it. The company who owns the software allows others to use the software under license agreements, usually for a price. In other words, users must pay for a license that allows them to use software that is owned and controlled by someone else, and only use it in whichever ways are allowed by the end-user license agreement (EULA). Not having the freedom to examine the source code makes it impossible to determine what the software does or how it works. Most people don't know how to modify their software, so why should it matter to them? Even if one has no need to access the source code themselves, it is still essential that it is open because there are others that can. It could be spying on you, or doing anything without your knowledge or consent. Instead of the user being in full control of their software, it is really the software owner who controls it. On top of that, any problems with the software can only be handled by the software creators so if they neglect to address them, the users are helpless. Proprietary software also tends to use unfair tactics to lock users in and keep them from switching. Often, they will use proprietary, non-standard formats that can only be used with their software, so users are stuck with it and forced to pay for upgrades over time. These issues are non-existent with Free Software.

There is just as much pragmatism as idealism in the Free Software philosophy. Free Software encourages people to examine, share, and modify the source code, and in fact it thrives upon this. The Free Software model is much more effective at producing higher quality software, faster. Instead of development resting on the shoulders of whoever owns the software, anyone is welcome to contribute. Free Software is examined and developed by hackers and programmers around the world. People working on improving Free Software are most often volunteering, but many organizations with a financial interest in the software will invest money or hire developers to work on it. For example, Canonical, which makes money offering paid support for Ubuntu, hires developers to improve it so that it will gain more users and Canonical will have a larger potential customer base (Moody 2008). All of this results in rapidly evolving software that develops exponentially (Deshpande 2008). This means that the software improves faster, bugs and security vulnerabilities are fixed sooner, and innovation is fostered. The Free Software model produces software that is quantitatively much better than the proprietary alternatives in virtually every conceivable way. It has a lower total cost of ownership (TCO), is safer, faster, more flexible, scalable, secure, stable, and reliable (Wheeler 2007).

A common myth about Free Software is that it is maintained by a community of people and not by a corporate entity, the quality of the product is greatly reduced (O'Reilley 1999). It is actually the openness and transparency of the the Free Software development process that makes the resulting software better, and, as mentioned before, many Free Software projects are funded and developed by companies and paid programmers. When new code is submitted to be included in a free software project, it will first be scrutinized by several people before it can be approved, so there is no risk of malicious modifications. The pace at which a Free Software project evolves is proportional to the popularity and usefulness of the software. This means that any successful Free Software project is almost guaranteed to be very reliable.

Another misconception is that Free Software is only secure because less people use it so it is less of a target. The truth is that the Free Software model inherently produces more stable and secure software. With the code open and available for so many people to examine, bugs and security vulnerabilities are found and responded to immediately. As the notion of Linux's Law by Eric S. Raymond states, “given enough eyeballs, all bugs are shallow” (Raymond 1997).

Proprietary software relies on "security by obscurity" or hiding the code to prevent the exposure of flaws, which has been shown over and over again to fail. It is frequently the case that a patch to fix one security problem in closed-source software has created another problem or even failed to fix the actual problem, and other times a vendor may leave a known flaw unresolved for months or even years at a time (Wikipedia 2009). The open source software model doesn't have these issues because it does not serve the interests of the software owners. It exists for its users and will always improve in their interest.

Other arguments against Free Software usually say things like, there is no way to make money with open source software, or open source software is anti-business, but Free Software is commercial software as well (O'Reilley 1999). These lies are propagated by huge marketing campaigns of fear, uncertainty, and doubt (FUD) to smear Linux funded by software giants like Microsoft (Asay 2009). These companies are incapable and unwilling to adapt to the changing software world, and cannot compete with Free Software. The proprietary software model is dying, and Free Software is unstoppable.

Linux is far more widespread than most people are aware. Businesses, governments, schools, scientific institutions, and homes have all adopted Linux on a variety of platforms. Linux is used on servers and desktop computers, is the most popular choice for supercomputers, and, being open source, is included on many embedded systems from mobile devices like phones, to gaming devices, to media appliances like TiVo.

Aside from these technological and ethical reasons to adopt Free Software, schools should immediately recognize the parallels of the philosophy of this movement with academic freedom and the open dissemination of knowledge and information common in academia. Free Software is about sharing information. Free Software is about learning from each other. Free Software is about community. This is consistent with the function of schools, so they should be using Free Software. "The advances in all of the arts and sciences, indeed the sum total of human knowledge, is the result of the open sharing of ideas, theories, studies and research. Yet throughout many school systems, the software in use on computers is closed and locked, making educators partners in the censorship of the foundational information of this new age." (Vessels 2001)

Schools also owe it to their community to adopt Free Software. The financial advantage of Free Software is especially important for schools since taxpayer money should not be given to serve the interests of proprietary software providers when open source software is available. On top of that, by using proprietary software at school, students and their parents are forced into having the same proprietary software at home, further propagating proprietary software's control and suppressing Software Freedom. Finally, since all trends indicate that Free Software will continue to grow at a faster pace than proprietary software as it always has, there is an ever-increasing demand for skills using Free Software. If schools are supposed to prepare students for the future, they should be the first to abandon proprietary software. Schools should be independent of corporate control over their software.

In my work with Newton North High School, I've found that the same roadblocks that stop people from adopting Linux personally exist for schools as well. The main obstacle is simply that students, teachers, and most importantly system administrators are reluctant to change, but that quickly goes away once they have had time to warm up to it. Another significant problem is government contracts requiring schools purchase new hardware from certain companies that might only sell machines with Windows installed. These roadblocks are inconvenient but can be dealt with without too much trouble.

I've found a number of ways to promote Ubuntu, a free operating system, to the Newton Public Schools. As long as you have one person from within the school system who is willing to work with you, you can make progress. Firstly, it is best to start from the top and work down from there. I contacted the Head of Technology for the Newton Public Schools, Ms. Chamberlain, and she told me to start by working to implement a test lab at North. I've worked with Chris Murphy and Phil Golando to get two demonstration computers set up in the school library and helped Mr. Golando install Ubuntu on his own machine to get familiar with. Once the success of this lab is recognized, other labs may be converted to Ubuntu as well.

Diving into the Free Software community around Ubuntu, I have seen how local advocacy teams run and develop, and I've been able to start a global team focusing on marketing and activism for Free Software gaming. from my experience, I plan to take advantage of the Free Software community and start an activist team to advocate Free Software to schools and write a Free Software statement which schools can sign to pledge their commitment towards adoption.

Free Software protects users' freedoms, all the while producing better software that's more stable and safe and improves faster. Instead of software being made by them for the usage rights to be sold to us, it is software owned by everyone, for everyone. Schools would save money by adopting it, as well as prepare their students for the future by adopting Free Software. Despite some obstacles that educational institutions and school systems face migrating to Free Software, they have an obligation to promote the fundamental philosophy of freedom, openness, and sharing that the Free Software Movement and academia share.

Bibliography

Stallman, Richard M. 2002. *Free Software, Free Society: Selected Essays of Richard M. Stallman*. Boston, Massachusetts. GNU Press. http://www.gnu.org/philosophy/fsfs/rms-essays.pdf (accessed May 6, 2009).

Moody, Glyn. 2008. *Interview: Mark Shuttleworth, founder of Ubuntu*. http://www.guardian.co.uk/technology/2008/may/22/internet.software (accessed May 13, 2009).

Wheeler, David A. *Why Open Source Software / Free Software (OSS/FS, FLOSS, or FOSS)? Look at the Numbers!* http://www.dwheeler.com/oss_fs_why.html (accessed May 8, 2009)

Deshpande, Amit and Dirk Riehle eds. 2008. *The Total Growth of Open Source.* http://dirkriehle.com/2008/03/14/the-total-growth-of-open-source/ (accessed May 10, 2009).

O'Reilley, Tim. 1999. *Ten Myths about Open Source Software*. http://www.oreillynet.com/pub/a/oreilly/opensource/news/myths_1199.html (accessed May 10, 2009).

Asay, Matt. 2009. *Facts behind Microsoft's anti-Linux campaign*. http://news.cnet.com/8301-13505_3-10145332-16.html (accessed May 2, 2009).

Vessels, Terry. 2001. Why should open source software be used in schools?http://edge-op.org/grouch/schools.html (accessed May 8, 2009).

Wikipedia contributors. 2009. Comparison of open source and closed source. Wikipedia, The Free Encyclopedia, http://en.wikipedia.org/w/index.php?title=Comparison_of_open_source_and_closed_source&oldid=261647655 (accessed January 2, 2009).

Definition of Free Cultural Works contributors. 2008. Definition. http://freedomdefined.org/index.php?title=Definition&oldid=5437 (accessed May 16, 2009).

Raymond, Eric S. 1997. *Release Early, Release Often*. http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/ar01s04.html (accessed May 15, 2009)