Open source software: Controlling your computing environment

Open source software (OSS) – free to use, reuse, study, modify, and distribute – is quickly being adopted by libraries today. From office productivity suites such as OpenOffice to library-specific applications such as an integrated library system, "next generation" library catalogs and Firefox extensions. Open source software has a lot to offer libraries. This session looks at the many types of OSS available, how libraries are making use of it, and how it can be exploited in order to control your local computing environment.

Myriad of definitions

At its core, open source software is a process for creating and distributing software, but there are many nuances to this overly broad definition. For example, many people associate open source software with the word "free". Such an association itself needs to be qualified because in this case "free" should equated with liberty, not gratis. A person is licensed the freedom – at liberty – to modify open source software in any way they desire. On the other hand, open source software is also "as free as a free kitten", meaning that, while it is given away, it does not come without financial costs. There are costs for hardware, personnel, and training. It is possible to purchase support for open source software, so there is yet another costs. There are costs of migrating from one system to another. There are emotional costs. "I like the way this system works." All of these costs are also associated with "closed" source software, but at least with open source software you get two additional things. First, you get to see the entire package before making a commitment. Second, open source software is more standards-compliant than "closed" source software because there are fewer proprietary features. Consequently, people who spend time installing, customizing, and modifying open source software are learning transferable skills that can be applied more readily in other venues. In this way, the choice of implementing open source software is also an investment in local people instead of remote share holders.

While the concept of liberty is at the core of open source software, the process of writing and distributing open source software are the things that has really made it sustainable. This process is often referred to as "scratching an itch", and it really would not be possible without the Internet. Here is how it works. A person has a computing problem – an itch. They go about writing a program attempting to solve the problem. The programmer realizes other people may have similar problems, and consequently the programmer shares their code with these other people. Some of these other people pick up the software, modify it to suit their own needs, and (sometimes) share their modifications back with the original programmer. If the modifications fit with the overall philosophy of the original developer, then the enhancements are incorporated into the original distribution, and the whole process is begun anew. Rinse. Shampoo. Repeat.[†] If the modifications do not fit within philosophy of the original developer, and the second developer really wants to distribute their enhancements, then a "fork" – a new branch – of the software is created. Again, rinse. Shampoo. Repeat. Without the Internet, the communications process necessary for the sharing of code would be an impediment to success.

Open source software is about community. It is about people coming together to support a common interest. (The phrase "a common cause" may be too strong.) Each community will create its own working dynamics. Some will vote. Some will practice meritocracy – governance or the holding of power by people selected on the basis of their ability. Some will be ruled by a benevolent dictator. The book by Eric Raymond called *The Cathedral and the Bazaar* described open source software development as being fueled by the altruistic hacker interested in praise from their fellow programmers. The book *The Success of Open Source* by Steven Weber sees the open source software community as a set of problem solvers. While I used to lean towards the former, I see more and more the realism of the second.

[†] There is a joke in computing circles, and it goes like this. Did you hear about the programmer who got stuck in the shower? He followed the instructions on the back of the bottle. Rinse. Shampoo. Repeat.

OSS and librarianship

There are many aspects of open source software which are akin to the principles of librarianship. First and foremost in my mind is the expected use of the deliverable – none. In the first case, the deliverable is software. In the second, it is data and information. In both cases, there are very few expectations in regard to what a person does with the information. "You are free to use and modify the program in any way you desire... It is none of my business why you want to know the answer to this particular reference question or borrow that book."

Both the open source community and the academic library community value forms of peer review. In open source, this is commonly paraphrased as "Given enough eyeballs, all bugs are shallow." Meaning, if many people look at the source code, then sooner rather than later all the problems with it will be discovered. This is the root purpose of peer review as well, except the "many eyeballs" are traded for a few experts.

To some degree, open source software is about the common good, but this is increasingly debated. More often it is seen as a way to avoid vendor lock-in and have more control over one's computing environment. Many people believe libraries are a public good, but given the increasing numbers of institutions who provide information services, this is something I debate with myself.

Open source software is about community. So is librarianship. It is not possible to create, grow, and maintain open source software without a high degree of collegiality and collaboration. I believe I read someplace that one of the most common words appearing in library position announcements is some form of the word "collaboration".

Open source software in libraries

As you may or may not know, the Internet just about runs on open source software. For example, most of the Web is run using a program called Apache, an open source HTTP server. The program used to convert most host names (like www.library.nd.edu) into IP addresses (like 129.74.250.207) is called named, a venerable golden oldie when it comes to open source. A whole lot of your email is delivered from computer to computer through a program called sendmail, just about as old as named. All of these applications run on top of different versions of Linux and a suite of software called GNU.

In Library Land open source software is increasingly used. Many of you probably use Firefox as your Web browser. Many of you probably write blogs, and the most popular blog software is WordPress. The root of many "next generation" library catalogs is an indexing application called Lucene. Many of our current integrated library systems implementing Z29.50 use open source software modules from Index Data called Zebra. Then there are the "next generation" library catalog applications themselves: VuFind, Evergreen, Koha, Blacklight, Scriblio, etc. If you have digital collections and built the digital collection software yourself, then dollars to donuts you used a relational database program called MySQL. Open source software is just about everywhere when it comes to computers.

Open source software is not "better" than closed source software. Nor is closed source software "better" than open source source software. For the most part, both types of software get the job done. For the most part, both types of software have similar costs – both financial and emotional. The difference lies in control. Closed source software is akin to buying a car with the hood welded shut. You can not open it up to fix it. You have limited abilities get into its guts and change it functionality. Granted, most people are not mechanics, yet only a small number of mechanics who have seamless ways to communicate are all that are needed to support a large community of automobile drivers. In a profession like our own – one of data and information distributed through computers – it behooves us to have as much control over our computing environment as possible. It behooves us to support and learn how to exploit open source software.

Eric Lease Morgan
University of Notre Dame

March 28, 2009