Open Source Software in Libraries: Exercises for the Windows Users in the Crowd

Eric Lease Morgan

September 17, 2004

Table of Contents

Introduction 1
Installation 2
World Wide Web (Apache) 2
Programming/scripting (Perl) 3
Z39.50 (Yaz and Zebra) 3
MARC records (MARC::Record) 4
Indexing and searching (Swish-e) 5
XML (xmllint and xsltproc) 5
Databases (MySQL) 6
Gift Cultures, Librarianship, and Open Source Software Development 7
    Acknowledgements 9
    Notes 9

Introduction

This handout brings together much of my experience with open source software. It describes sets of successful open source software projects and tries to enumerate the qualities of successful project. The handout has been in created the hopes people will read it, give the exercises a whirl, learn from the experience, and share their newly acquired expertise with the world at large. I believe open source software is more about building communities and less about computer programs. It is more about making the world a better place and less about personal profit. Allow me to explain. Through this process I hope we can make the world we live in just a little bit better place. Idealist? Maybe. A worthy goal? Definitely.

As I was putting this handout together I thought about a conference Apple Computer sponsored in 1995 called Ties That Bind: Converging Communities. In the conference we shared and discussed ideas about community and the ways technology can help make communities happen. In between a session Steve Cisler, the facilitator of the conference, displayed the original piece of art that became the motif for the

conference. He noted that he got the painting in Australia some time the previous year. He liked it for its simplicity and connectivity. The painting is acrylic, approximately 1' 6" X 2" 6", and is composed of many simple dots of color.

The image at the top of the page is that piece of art, and it is significant today. It too is a whole lot like open source software and the "the Unix way." The value of open source software is measured in terms of its simplicity and connectivity. The simpler and more connective the software, the more it is valued. The Unix way is a philosophy of computing. It posits that a computer program will take some input, do some processing, and provide some output. There is very little human interface to these sorts of programs because they get their input from a thing called standard input (STDIN) and send the output to a thing called standard output (STDOUT). If errors occur, errors are sent to standard error (STERR). Since the applications are expected to get their input from STDIN and send it to STOUT it is possible to string many together to create a working application. Connectivity. Such a design philosophy allows tiny programs to focus on one thing, and one thing only. Simplicity. This modular approach allows for the creation of new applications by adding or deleting older modules from the string.

The motif brought to my attention by Cisler is a lot like stringing together open source software applications. Each individual dot does not do a whole lot on its own, but strung together they form a pattern. The pattern's whole is greater than the sum of its parts. This is true of communities as well. Individuals bring something to the community, and the community is made better for the contribution. The open source community exists because of individuals. These individuals have particular strengths (and weaknesses). As people add what they can to the community, the community is strengthened. The rewards for these contributions are rarely monetary. Instead, the contributions are paid for with respect. People who give freely of themselves and their time are rewarded by the community as experts whose opinions are to be taken seriously. True, participation in open source software activities does not always put food on the table, but neither do other community-based activities our society values to one degree or another such as participation in community theater, helping out at the local soup kitchen, being involved in church activities, picking up litter, giving directions to a stranger, supporting charities, participating in fund-raisers, etc. Open source software is about communities, communities that have been easier to create with the advent of globally networked computers. As described later, it is about "scratching an itch" to solve a problem, but it is also about giving "freely" to the community in the hopes that the community will be better off for it in the end.

Installation

1. Copy all of directories in the For the Root of the C Drive directory to... the root of the C drive.

2. Update your path environment variable to include the following: c:\Perl\bin;c:\Apache\bin;c:\Mysql\bin;c:\Yaz\bin;c:\extras;c:\;c:\Libxml;c:\Swish-e;. This can be done through the System control panel or through the use of the PATH command in a terminal window.

World Wide Web (Apache)

Apache is the most popular Web (HTTP) server on the Internet and a standard open source piece of software. It's name doesn't really have anything to do with American Indians. Instead, it's name comes from the way it is built. It is "a patchy" server, meaning that it is made up of many modular parts to create a coherent whole. This design philosophy has made the application very extensible. For example, there are the core modules that make up the server's ability to listen for connections, retrieve files, and return them to the requesting client (the "user agent" in HTTP parlance). There are other modules dealing with logging transactions and CGI (common gateway interface) scripting. Other modules allow you to rewrite incoming requests, manage email, implement the little-used HTTP PUT method, write other modules in Perl, or transform XML files using XSLT. Apache is currently at version 2.0, but for some reason many people are still using the 1.3 series. I don't really know why. I have not upgraded my Apache servers to

version 2.0 because I do not want to loose the functionality of AxKit, an XML transformation engine. Apache is a part of LAMP (Linux Apache MySQL Perl/PHP), a term coined by RedHat to denote the core open source applications dealing with stuff Web.

1. Open the Apache directory and launch Apache by double-clicking on its icon.
2. Connect to http://127.0.0.1/.
3. Write an HTML file named home.html and save it in the htdocs directory.
4. Connect to http://127.0.0.1/home.html.

Programming/scripting (Perl)

Perl is a programming language. Originally written to handle various systems administration tasks, Perl's strength lies in its ability to manipulate strings (text). Perl matured through the era of Gopher but really started becoming popular with the advent to CGI scripting. Perl has been ported to just about any computer operating system, has one of the largest numbers of support forums, and has been written about in more books than you can count. Perl can be compiled into Apache making it possible to run Perl scripts as fast as C programs. It easily connects to database applications through a module called DBI. It can be run from the command line. It can listen and respond to networking connections. It can call many aspects of your computer's operating system. In short, Perl is mature and very robust. Other very good programming languages exist and can do much of what Perl can do. Examples include other "P" languages such as PHP and Python. These languages are becoming increasingly popular, especially PHP, but at the risk of starting a religious war, I advocate Perl because of its very large support base and its cross-platform functionality.

1. Read Perl's help texts. (c:\Perl\html\perltoc.html).
2. Open a command prompt and change directories to the Extras directory (**cd c:\extras**).
3. Run Hello, World! (**perl hello-world.pl**).

Z39.50 (Yaz and Zebra)

YAZ is a C library and resulting binary application implementing a Z39.50/SRW client. Zebra is an indexer and Z39.50 server. The yaz-client is a straight-forward terminal application. Zebraidx is the indexer, and requires bunches o' configuration files. It is not as straight-forward as other indexers, but its data can be served by zebrasrv. Since the client is built on a library, it can (and has) been compiled into other tools such as PHP and Apache. The YAZ API also has a Perl interface. YAZ/Zebra are definitely worth your time exploring if you want to make your collections available through Z39.50. Yes, you will spend time learning the in's and out's of Z39.50 in the process, but that experience can be taken forward and applied on other venues where Z39.50 is needed.

1. Read Yaz's help text (c:\Yaz\doc\yaz.html).
2. Create a set of MARC records. Open a command prompt and enter these commands:

    - **yaz-client**

- **help**
- **open z3950.loc.gov:7090/voyager**
- **set_marcdump c:\extras\catalog.marc**
- **find origami**
- **show 1 + 100**
- **close**
- **quit**

MARC records (MARC::Record)

This Perl module is the Perl module to use when reading and writing MARC records. It is very well supported on the Perl4Lib mailing list, and a testament to the module's abilities is its incorporation into things like Koha and Net::Z3950. If you are not familiar with object oriented programing techniques in Perl, then MARC::Record might take a bit of getting used to. On the other hand, learning to use MARC::Record will not only improve your programming abilities but it will educate you on the intricacies of the MARC record data structure, a structure that was designed in an era of scarce disk space, non-relational databases, and little or no network connectivity.

1. View MARC records (**more c:\extras\catalog.marc**).
2. View them again:
    a. **perldoc marcdump**
    b. **marcdump c:\extras\catalog.marc | more**
3. View them yet again:
    a. **cd c:\extras**
    b. **perl marc-read.pl c:\extras\catalog.marc | more**
4. Write new MARC records (**perl marc-write.pl**).
5. Convert MARC to XHTML:
    a. **cd c:\extras**
    b. **perl marc2xhtml2.pl c:\extras\marc\single\ c:\extras\xhtml**
    c. use your browser to peruse the contents of c:\extras\xhtml\

Indexing and searching (Swish-e)

Swish-e is an uncomplicated indexer/search engine. Once built you feed the swish-e binary a configuration file and/or a set of command line switches to index content. This content can be individual files on a file system, files retrieved by crawling a website, or a stream of content from another application such as a database. The indexing half of swish-e is able to index specifically marked up text in XML and HTML as fields for searching later. The indexes created by swish-e are portable from file system to file system. The same binary that creates the indexes can be used to search the indexes. Swish-e supports relevance ranking, Boolean operations, right-hand truncation, field searching, and nested queries. Later versions of swish-e come with a C and Perl API allowing developers to create CGI interfaces to these indexes. Swish-e is an unsung hero. It's inherently open nature allows for the creation of some very smart search engines supporting things like spelling correction, thesaurus intervention, and "best bets" implementations. Of all the different types of information services librarians provide, access to indexes is one of the biggest ones. With swish-e librarians could create their own indexes and rely on commercial bibliographic indexers less and less.

1. Read the swish-e help (**swish-e -h**).
2. Read more help (**c:\Swish-e\share\doc\swish-e\html\index.html**).
3. Index XHTML files. Open a command prompt and enter these commands:
    a. **cd c:\extras**
    b. **swish-e -c xhtml.cfg**
4. Search XHTML files:
    a. **swish-e -f xhtml\xhtml.idx -w art**
    b. **swish-e -f xhtml\xhtml.idx -w science**
    c. **swish-e -f xhtml\xhtml.idx -w art and science**
5. Search XHTML files on the Web (http://127.0.0.1/xhtml.cgi).

XML (xmllint and xsltproc)

Xsltproc and its companion program, xmllint, are very useful applications for processing XML files with XSL. Both applications are built from a C library that is becoming increasingly popular for parsing and processing XML documents. By feeding xsltproc an XSL stylesheet and an XML data file you can transform the XML data file into any one of a number of text files whether they be SQL, (X)HTML, tab-delimited files, or even plain text files intended for printing. Xmllint is a syntax checker. Given an XML file, xmllint will check the validity of your XML files against a DTD. By first installing the C library and mod_perl, you will be able to incorporate AxKit into your Apache HTTP server allowing you to transform XML data on the fly and serve it accordingly. Swish-e desires the C library. It is easy to use the DocBook stylesheets with xsltproc to create XHTML versions of your DocBook files. With xsltproc

and a plain o' text editor, you can learn a whole lot about XML.

1. Convert MARC to MARCXML. Open a command prompt and:
    a. **cd c:\extras**
    b. **perl marc2marcxml.pl c:\extras\marc\single\ c:\extras\marcxml**
    c. again, use your browser to peruse the contents of c:\extras\marcxml\
2. Convert MARC to MARCXML, again:
    a. **cd c:\extras * perl marc2xml.pl c:\extras\marc\single\catalog.marc > catalog.xml**
    b. yet again, use your browser to examine c:\extras\catalog.xml
3. Read help. Open a command prompt and:
    a. **xmllint**
    b. **xsltproc**
4. Convert MARCXML to MODS (**xsltproc MARC21slim2MODS3.xsl catalog.xml > catalog.mods**)
5. Use your browser to view c:\extras\catalog.mods.
6. Validate XML (**xmllint --valid twain.xml**).
7. Validate XML, again (**xmllint --valid --noout twain.xml**).
8. Convert TEI to HTML (**xsltproc tei2html.xsl twain.xml > twain.html**).
9. Open c:\extras\twain.html in your Web browser.
10. Convert MODS to SQL (**xsltproc mods2sql.xsl catalog.mods > catalog.sql**).
11. Open c:\extras\catalog.sql in your favorite text editor.

Databases (MySQL)

MySQL is a relational database application, pure and simple. Billed as "The World's Most Popular Open Source Database" MySQL certainly has a wide support in the Internet community. Many people think MySQL can't be very good because it is free, especially Oracle database administrators. True, it does not have all the features of Oracle, nor does it require a specially trained person to keep it up and running. A part of the LAMP suite, MySQL compiles easily on a multitude of platforms. It comes as a pre-compiled binary for Windows. It has been used to manage millions of records and gigabytes of data. Fast and robust, it supports the majority of people's relational database needs. On its down side, it does not cur-

rently support triggers, transactions, nor roll-backs. Nor does it have a GUI interface. At the same time, a program called phpMyAdmin, a set of PHP scripts, can be used to manage, manipulate, and query MySQL database through a Web browser window. If there were one technical skill I could teach the library profession, it would be the creating and maintenance of relational databases, and I would teach them how to use MySQL.

1. Read MySQL's help (c:\Mysql\Docs\manual_toc.html).
2. Start Mysql (open a command prompt and **mysqld --standalone**).
3. Create database (open another command prompt and **mysqladmin -uroot create catalog**).
4. Fill database:
    a. **cd c:\extras**
    b. **mysql -uroot catalog < catalog.sql**
5. Use the database. Enter these commands from the command prompt:
    a. **mysql -u root**
    b. **show databases;**
    c. **use catalog;**
    d. **show tables;**
    e. **explain items;**
    f. **select title from items where title like '%origami%';**

Gift Cultures, Librarianship, and Open Source Software Development

This short essay examines more closely the concept of a "gift culture" and how it may or may not be related to librarianship. After this examination and with a few qualifications, I still believe my judgments about open source software and librarianship are true. Open source software development and librarianship have a number of similarities -- both are examples of gift cultures.

I have recently been reading a book about open source software development by Eric Raymond. [1] The book describes the environment of free software and tries to explain why some programers are willing to give away the products of their labors. It describes the "hacker milieu" as a "gift culture":

Gift cultures are adaptations not to scarcity but to abundance. They arise in populations that do not have significant material scarcity problems with survival goods. We can observe gift cultures in action among aboriginal cultures living in ecozones with mild climates and abundant food. We can also observe them in certain strata of our own society, especially in show business and among the very wealthy. [2]

Raymond alludes to the definition of "gift cultures", but not enough to satisfy my curiosity. Being the

good librarian, I was off to the reference department for more specific answers. More often than not, I found information about "gift exchange" and "gift economies" as opposed to "gift cultures." (Yes, I did look on the Internet but found little.)

Probably one of the earliest and more comprehensive studies of gift exchange was written by Marcell Mauss. [3] In his analysis he says gifts, with their three obligations of giving, receiving, and repaying, are in aspects of almost all societies. The process of gift giving strengthens cooperation, competitiveness, and antagonism. It reveals itself in religious, legal, moral, economic, aesthetic, morphological, and mythological aspects of life. [4]

As Gregory states, for the industrial capitalist economies, gifts are nothing but presents or things given, and "that is all that needs to be said on the matter." Ironically for economists, gifts have value and consequently have implications for commodity exchange. [5] He goes on to review studies about gift giving from an anthropological view, studies focusing on tribal communities of various American indians, cultures from New Guinea and Melanesia, and even ancient Roman, Hindu, and Germanic societies:

The key to understanding gift giving is apprehension of the fact that things in tribal economics are produced by non-alienated labor. This creates a special bond between a producer and his/her product, a bond that is broken in a capitalistic society based on alienated wage-labor.[6]

Ingold, in "Introduction To Social Life" echoes many of the things summarized by Gregory when he states that industrialization is concerned:

exclusively with the dynamics of commodity production. ... Clearly in non-industrial societies, where these conditions do not obtain, the significance of work will be very different. For one thing, people retain control over their own capacity to work and over other productive means, and their activities are carried on in the context of their relationships with kin and community. Indeed their work may have the strengthening or regeneration of these relationships as its principle objective. [7]

In short, the exchange of gifts forges relationships between partners and emphasizes qualitative as opposed to quantitative terms. The producer of the product (or service) takes a personal interest in production, and when the product is given away as a gift it is difficult to quantify the value of the item. Therefore the items exchanged are of a less tangible nature such as obligations, promises, respect, and interpersonal relationships.

As I read Raymond and others I continually saw similarities between librarianship and gift cultures, and therefore similarities between librarianship and open source software development. While the summaries outlined above do not necessarily mention the "abundance" alluded to by Raymond, the existence of abundance is more than mere speculation. Potlatch, "a ceremonial feast of the American Indians of the northwest coast marked by the host's lavish distribution of gifts or sometimes destruction of property to demonstrate wealth and generosity with the expectation of eventual reciprocation", is an excellent example. [8]

Libraries have an abundance of data and information. (I won't go into whether or not they have an abundance of knowledge or wisdom of the ages. That is another essay.) Libraries do not exchange this data and information for money; you don't have to have your credit card ready as you leave the door. Libraries don't accept checks. Instead the exchange is much less tangible. First of all, based on my experience, most librarians just take pride in their ability to collect, organize, and disseminate data and information in an effective manner. They are curious. They enjoy learning things for learning's things sake. It is a sort of Platonic end in itself. Librarians, generally speaking, just like what they do and they certainly aren't in it for the money. You won't get rich by becoming a librarian.

Information is not free. It requires time and energy to create, collect, and share, but when an information exchange does take place, it is usually intangible, not monetary, in nature. Information is intangible. It is difficult to assign it a monetary value, especially in a digital environment where it can be duplicated effortlessly:

An exchange process is a process whereby two or more individuals (or groups) exchange goods or ser-

vices for items of value. In Library Land, one of these individuals is almost always a librarian. The other individuals include tax payers, students, faculty, or in the case of special libraries, fellow employees. The items of value are information and information services exchanged for a perception of worth -- a rating valuing the services rendered. This perception of worth, a highly intangible and difficult thing to measure, is something the user of library services "pays", not to libraries and librarians, but to administrators and decision-makers. Ultimately, these payments manifest themselves as tax dollars or other administrative support. As the perception of worth decreases so do tax dollars and support. [9]

Therefore when information exchanges take place in libraries librarians hope their clientele will support the goals of the library to administrators when issues of funding arise. Librarians believe that "free" information ("think free speech, not free beer") will improve society. It will allow people to grow spiritually and intellectually. It will improve humankind's situation in the world. Libraries are only perceived as beneficial when they give away this data and information. That is their purpose, and they, generally speaking, do this without regards to fees or tangible exchanges.

In many ways I believe open source software development, as articulated by Raymond, is very similar to the principles of librarianship. First and foremost with the idea of sharing information. Both camps put a premium on open access. Both camps are gift cultures and gain reputation by the amount of "stuff" they give away. What people do with the information, whether it be source code or journal articles, is up to them. Both camps hope the shared information will be used to improve our place in the world. Just as Jefferson's informed public is a necessity for democracy, open source software is necessary for the improvement of computer applications.

Second, human interactions are a necessary part of the mixture in both librarianship and open source development. Open source development requires people skills by source code maintainers. It requires an understanding of the problem the computer application is trying to solve, and the maintainer must assimilate patches with the application. Similarly, librarians understand that information seeking behavior is a human process. While databases and many "digital libraries" house information, these collections are really "data stores" and are only manifested as information after the assignment of value are given to the data and inter-relations between datum are created.

Third, it has been stated that open source development will remove the necessity for programers. Yet Raymond posits that no such thing will happen. If anything, there will an increased need for programmers. Similarly, many librarians feared the advent of the Web because they believed their jobs would be in jeopardy. Ironically, librarianship is flowering under new rubrics such as information architects and knowledge managers.

It has also been brought to my attention by Kevin Clarke (kevin_clarke@unc.edu) that both institutions use peer-review:

Your cultural take (gift culture) on "open source" is interesting. I've been mostly thinking in material terms but you are right, I think, in your assessment. One thing you didn't mention is that, like academic librarians, open source folks participate in a peer-review type process.

All of this is happening because of an information economy. It sure is an exciting time to be a librarian, especially a librarian who can build relational databases and program on a Unix computer.

Acknowledgements

Thank you to Art Rhyno (arhyno@server.uwindsor.ca) who encouraged me to post the original version of this text.

Notes

1. Raymond, E.S., The cathedral and the bazaar : musings on Linux and open source by an accidental revolutionary. 1st ed. 1999, [Sebastopol, CA]: O'Reilly.

2. Ibid. pg. 99.

3. Mauss, M., The gift; forms and functions of exchange in archaic societies. The Norton library, N378. 1967, New York: Norton.

4. Lukes, S., Mauss, Marcel, in International encyclopedia of the social sciences, D.L. Sills, Editor. 1968, Macmillan: [New York] volume 10, pg. 80.

5. Gregory, C.A, "Gifts" in Eatwell, J., et al., The New Palgrave : a dictionary of economics. 1987, New York: Stockton Press. volume 3, pg. 524.

6. Ibid.

7. Ingold, T., Introduction To Social Life, in Companion encyclopedia of anthropology, T. Ingold, Editor. 1994, Routledge: London ; New York. p. 747.

8. Merriam-Webter Online Dictionary, http://search.eb.com/cgi-bin/dictionary?va=potlatch

9. Morgan, E.L., Marketing Future Libraries, http://infomotions.com/musings/marketing/