The journal of Apple technology.

Volume Number: 19 (2003)
Issue Number: 12
Column Tag: Programming

Section 7

ESR, etc.

by Rich Morin

A multiplex review of a singular individual...

Eric S. Raymond (aka ESR) is best known for his activities in support of Open Source software. Several years ago, he convinced a group of key developers to adopt the term as a less-ambiguous and more "marketable" replacement for the existing term, "free software".

Note: The term "Open Source" was actually coined by Christine Peterson (President of the Foresight Institute; http://www.foresight.org). Also, the term "Free Software" still has its adherents; see http://www.fsf.org/philosophy.

Since that time, Eric has written extensively on topics related to Open Source software. His essay "The Cathedral and the Bazaar" is probably the best known of these essays, but the others are also worth reading. See "The Cathedral and the Bazaar: Musings on Unix and Open Source by an Accidental Revolutionary" (O'Reilly; http://www.oreilly.com).

Two of Eric's essays, "A Brief History of Hackerdom" and "The Revenge of the Hackers" appear in the excellent collection "Open Sources: Voices from the Open Source Revolution" (DiBona, et al; O'Reilly). If you want to understand hacker culture and the Open Source movement, both of these O'Reilly books would be excellent starting points.

Eric has also been busy on the organizational and legal sides of Open Source. He is President of the Board of Directors of the Open Source Institute (http://www.opensource.org). He has also published (and publicized) a number of internal Microsoft documents, weighed in on relevant lawsuits (e.g., the SCO unpleasantness), etc.

Other Publications

Eric writes extensively, covering a wide range of topics. His home page (http://catb.org/~esr) has links to essays on anthropology, economics, politics, science fiction, and many other areas. Even if you don't agree with Eric's opinions, you should find his writings to be interesting and thought-provoking.

As the long-standing steward of the Jargon File (http://www.jargon.org; also available as "The New Hacker's Dictionary" from MIT Press, http://mitpress.mit.com), Eric acts as an editor, lexicographer, historian, and (occasionally) anthropologist. The definitions tend to be far more interesting and enjoyable than one might expect from a conventional "dictionary".

The Art of Unix Programming

Over the past five years, Eric has been working on a distillation of the lessons that Unix has to offer the

programming community. Taking his own advice ("Many eyes make all bugs shallow.") seriously, he brought in a number of Unix wizards as advisors. As a result, "The Art of Unix Programming" (Addison-Wesley; http://www.aw.com) is enjoyable, educational, and for the most part, authoritative.

Although the book doesn't address Mac OS X, in particular, it has quite a bit to say about the BSD side of the operating system. For instance, it talks at length about the Unix practice of encoding information in text files. OSX, with something like 100K text files, certainly seems to have adopted this idea.

It also discusses popular ideas such as object-oriented programming, threads, giving reasons why these might not be the best possible approach in all circumstances. All told, this book is the best introduction I can suggest for a Mac OS programmer who wants to understand the BSD (read, Unix) side of Mac OS X.

More can be less

True to Unix tradition, Eric extols the benefits of having many different tools. Again, OSX is right there, providing a plethora of programming languages and other useful utilities. I wonder, however, whether OSX (and the Open Source community, in general) may not be suffering from an embarrassment of riches.

In perusing leads for programming positions, many of us have noted the extreme diversity of skills that may be demanded for a single position. Applicants are expected to have several years of experience in each of several languages and tools. Often, it is hard to conceive reasons for using these languages and tools in a single project.

Ignoring the discouraging effect of such postings on prospective applicants, let's consider how the company managed to get into the position of needing this sort of polymath. Speculating, I propose that the company once had a number of programmers, each developing part of a complex system. Each programmer chose the "right tool for the job", in true Unix fashion.

The company now has far fewer programming positions, but it still needs to maintain a polyglot mass of software. Their job postings are totally unrealistic - no single programmer can have extensive skills in more than a few areas - but they are merely a symptom of a larger problem.

OSX has similar historical baggage. If you look in /etc, you will find numerous control files. Most of these are "flat files", employing newlines to separate records and some form of delimiter (e.g., colons or "white space") to separate fields. Unfortunately, the exact format varies, so programs (and administrators) must "understand" different syntax for each file.

The log files are in even worse shape. They may be written by multiple programs, so the format can vary from entry to entry. Because the entries were written for human consumption, they often provide no unambiguous way for a program to distinguish fields. This is a serious problem; if log files are too big for humans to read, and too messy for programs to read, exactly what purpose are they serving?

Moving on to OSX's "plist" files, we find two distinct formats in evidence. One, inherited from NeXT, uses a C-like syntax. The other, strongly encouraged by Apple's current documentation, is based on XML. Using the appropriate frameworks, an Apple programmer can read either file. An administrator, however, may need to be familiar with both.

Although I'm a big fan of YAML (YAML Ain't Markup Language; http://www.yaml.org), I sometimes wonder if I might be making some programmer or administrator's life more difficult, by introducing yet another data format. Sigh.

Programming languages are another area of concern. OSX is built out of C, C++, Objective-C, and a smattering of scripting languages. C is used for the "Core OS" (kernel, BSD libraries and programs), but C++ is used for the IOKit and most device drivers. Objective-C is used for the higher-level frameworks and applications. Finally, the scripting languages are used for all sorts of administrative glue.

Fortunately, few programmers write programs in all these areas. If you're writing GUI-based apps, you're unlikely to be writing device drivers at the same time. If you're a serious scripter, you may never look at any C (etc) code. Unless, as I am, you're using Perl, Python, or Ruby to build Cocoa-based apps.

For all my misgivings, however, I wouldn't want to be restricted to a single data format or programming language. XML is powerful and very "buzzword-compliant", but it's also verbose and a poor match for associative arrays. Objective-C is a tidy integration of OO into the base C language, but its memory management seems a bit awkward to my Perl-accustomed eyes.

Ockham's Razor ("Pluralitas Non Est Ponenda Sine Necessitate.") transliterates to "Plurality should not be posited without necessity", reminding scientists to remain parsimonious in imagining new mechanisms. It is also translated, however, as "Entities are not to be multiplied beyond necessity" and even "Keep It Simple, Stupid".

In short, our challenge as developers is to decide how to achieve simplicity (once we have decided what "simplicity" is, in a given context :-), while retaining the power and flexibility that our mixed programming heritage provides.

Eric's latest offering, like many of his earlier writings, helps to show us the way...

Rich Morin has been using computers since 1970, Unix since 1983, and Mac-based Unix since 1986 (when he helped Apple create A/UX 1.0). When he isn't writing this column, Rich runs Prime Time Freeware (www.ptf.com), a publisher of books and CD-ROMs for the Free and Open Source software community. Feel free to write to Rich at rdm@ptf.com.