;login:

THE MAGAZINE OF USENIX & SAGE
February 2002 • Volume 27 • Number 1

inside:

USENIX & SAGE
The Advanced Computing Systems Association &
The System Administrators Guild

the bookworm

BOOKS REVIEWED IN THIS COLUMN

DOSSIER
Rich Morin, ed.
www.ptf.com/dossier/

LINUX STANDARD BASE SPECIFICATION 1.0.0
www.opencontent.org/openpub/

BROADBAND INTERNET CONNECTIONS
Roderick W. Smith
Boston, MA: Addison-Wesley, 2002. Pp. 616.
ISBN 0-201-73827-9.

CISCO IP ROUTING
Alex Zinin
Boston, MA: Addison-Wesley, 2002. Pp. 635.
ISBN 0-201-60473-6.

THE COMPUTER REVOLUTION IN CANADA
John N. Vardalas
Cambridge, MA: MIT Press, 2001. Pp. 409.
ISBN 0-262-22064-4.

MECHANIZING PROOF
Donald MacKenzie
Cambridge, MA: MIT Press, 2001.
ISBN 0-262-13393-8.

THE BOOK OF ZOPE
Beehive Electronic Media
San Francisco, CA: No Starch Press, 2001.
Pp. 408. ISBN 1-886411-57-3.

LEARNING PERL, 3RD ED.
Randal L. Schwartz and Tom Phoenix
Sebastopol, CA: O'Reilly & Associates, 2001.
Pp. 316. ISBN 0-596-00132-0.

by Peter H. Salus

Peter H. Salus is a member of the ACM, the Early English Text Society, and the Trollope Society, and is a life member of the American Oriental Society. He is Editorial Director at Matrix.net. He owns neither a dog nor a cat.

peter@matrix.net

Too many books. Too many books.

Useful books. Interesting books.

But . . .

If you're reading this and don't have a copy of *The Root of All Evil* (O'Reilly's third User Friendly volume), stop reading and buy it. It will shed new light on your colleagues.

Docs

If you're like me, the first thing you do when you don't know/understand something is ask the person in the next office (or cubicle) or you phone someone you think might know. The next thing you do is look at the man pages. I keep my UNIX manuals right by my desk because I prefer paper to the screen. But there's a ton of stuff out there.

Rich Morin has begun a wonderful series, called DOSSIER (Documenting Open Source Software for Industry, Education, and Research). Rich has collected documents, placed them into a straightforward taxonomy, and printed them in handy 400-page volumes. I spent several hours reading in the *Email: Mail and Sendmail* volume. (There's also an *Exim* volume.) It's just great! There are *File Systems*, *Kernel*, and *Text* volumes, three *PostgreSQL* volumes, and two *Python* volumes. They run just over $40 each, including postage. A bargain.

Collect 'em all from: *www.ptf.com/ptf/dossier/*

Great idea, Rich.

A Note on Standards

Back in 1987, USENIX hosted a POSIX workshop at the Berkeley marina. I've been reading standards docs since then. So I wasn't startled when the Free Standards Group gave me a hardcopy of the *Linux Standard Base Specification 1.0.0* at the ALS in November. This is a brave attempt at defining a system interface for compiled applications. As we have seen the UNIX world fragment and subsequently attempt standardization, perhaps efforts like this can prevent something similar happening to Linux.

Internet Stuff

Roderick W. Smith has taken up a much tougher job: explaining broadband connections. I was taken by his early chapters and thought that I might try to actually connect a box at home following his instructions. I failed. The box I wanted to connect was a SPARC Ultra 5. Smith provides "detail for Windows, MacOS, and Linux." I was out of luck. So, if you're running FreeBSD, OpenBSD, NetBSD, BSDi, or Solaris, this book will not be your cup of tea. It's a nice book. B, but it wasn't helpful to me.

In fact, apropos of this, I'm really tired of reading stuff that assumes that I've been grafted to an Intel chip. While I recognize that much of the world creeps along Wintel Way, I recall only too well the various instruction set/RISC arguments (and a really good paper by Dan Klein). Hey, remember the Z80?

Alex Zinin's fat volume is not for the fainthearted. It's a first-rate, thorough work on the "packet forwarding and intra-domain routing protocols." Zinin actually describes just what's going on inside the router. It's not an easy read, but it is worth the time you'll put into it.

A Version of History

John N. Vardalas has produced a fascinating narrative of the development of computer technology in Canada from

1945 through 1980. Unfortunately, it is a partial one. Vardalas has chosen to center his analysis on a few very large companies and on the military. The result is that he misses some of the more significant events, largely because they were either academic (or semi-academic) or because they concern software, not hardware.

Though the Canadian contributions to mail sorting and the airlines reservations system are mentioned, NewsWhole, the first real page-layout system (at the Toronto *Globe & Mail*), designed by David Tilbrook, has been missed. In fact, the incredible contributions of Ron Baecker and his students (Mike Tilson, Tom Duff, Rob Pike, etc.) are absent. HCR is missing. And some other notable Canadian software folk find no place here: Morwen Gentleman, Brian Kernighan, Heinz Lycklama, to pick a few. And while there are mentions of Toronto and British Columbia, I sought in vain for Waterloo or McMaster.

Ferranti of Canada, Sperry Gyroscope of Canada, and Control Data of Canada were important. But hardware gets overemphasized here. And the ARPANET/Internet doesn't even get an entry in the index, despite the fact that Atomic Energy of Canada in Chalk River, ON, was the first ARPANET site outside the US.

I learned a lot from Vardalas; I expect the expanded, second edition (in five years?) will be even better.

Prove It!

Donald MacKenzie has written a dense, difficult book on a fascinating topic: the history of proof, applied to what's done by traditional mathematicians and to what's required of formal, mechanized (computational) proof. This is a history of the interaction of mathematics and computation as viewed by a sociologist. MacKenzie's examination of the social influences on the development of automated proof is superb. He concludes that in pursuing dependable computer systems we don't eclipse the need for trust in human judgment. Tough stuff, but worthwhile.

Web Servers

Zope is the leading open source Web application server. Beehive Electronic Media is a German company that does Zope training and development. *The Book of Zope* is a straightforward introduction that appears to cover everything. Zope appears to be written, for the most part, in Python.

And a Final Perl

It's eight years since I got my first llama. Since then the book has become larger and better. Everything that Randal and Tom have learned in teaching Perl over the years seems to have gone into this new edition. If you don't have it, you need it. If you've been relying on the first or second editions, this one is far better.

WEB CACHING

DUANE WESSELS

Sebastopol, CA: O'Reilly & Associates, 2001. Pp. 318. ISBN: 1-56592-536-X

Reviewed by Alex Rousskov

The technical books I've read are usually of low quality. The factual material is buggy, analysis is shallow, and entertainment value is expectedly low. Most technical authors are unaccustomed to writing books and are too busy with their day-to-day routine to put much time into a book-length project. Most good editors are overloaded. Most publishers have to think of the bottom line first. Knuth-quality masterpieces are rare.

Duane Wessels' *Web Caching* would need thorough editing to meet my stringent quality requirements. However, as the first and only book devoted to an important subject, it is worth your consideration.

If your responsibilities or interests are related to the Internet and Web traffic, you must know about Web caching. Duane is a Web caching guru. He is the original and ongoing author of Squid, an open source free caching software that rivals commercial offerings in market share and features. Not familiar with Squid? Think Apache's httpd in the origin servers category. Duane also runs an international caching hierarchy that has been used by numerous researchers and practitioners worldwide to study and improve the Internet. Duane's name is on the Internet Cache Protocol (ICP), RFCs, and several landmark papers on Web caching. Duane has helped to make a series of international Web caching workshops a success. As an early industry player, Duane knows Web caching inside and out. He certainly has the expertise to write a book on Web caching. If only he could disable his email like Donald Knuth . . .

Web Caching starts with a two-chapter introduction to the topic that is meant to bring folks unfamiliar with HTTP and caching up-to-speed. Newbies may have a difficult time swallowing the dry cocktail of technical details, HTTP headers, and acronyms. In trying to build a foundation of knowledge for the rest of the book, the author often dives into unnecessary detail. If this is your first encounter with HTTP, brace yourself.

Chapter 3 talks about legal and ethical issues of Web caching and the politics that surrounds those matters. This is a rare case when politics belongs in a technical book. This may also be the only chapter that has a lot of original material not available in digestable form elsewhere on the Web. If you think caching is strictly a problem of placing content closer to the user, this chapter will open your eyes. If you are a cache maintainer advising company lawyers on technical matters or a lawyer advising technical folks, I strongly recommend reading this chapter before your users or content providers come after you or your company. Even caching experts are likely to find the discussion interesting and useful. The weak side of this chapter is mostly US-specific content and absence of clear-cut conclusions. Your company will still need a lawyer.

The next seven chapters talk mostly about configuring Web clients, Web caches, cache hierarchies, and origin servers to work with caches and/or to cache content efficiently. Generally useful, good technical stuff, though some topics would benefit from a more thorough treatment.. Many complex issues such as traffic interception or cache hierarchies are discussed with pros and cons of specific solutions compared. If you are deploying a cache on your network, you will find answers to many of your questions in these chapters. Some configuration details are likely to become outdated soon, but the book does a good job discussing general concepts and common pitfalls that are likely to remain relevant for at least a few years. Duane's open source software bias is especially evident in these chapters.

Chapter 11 is devoted to monitoring cache operation and would have been extremely useful to cache operators had it not been only nine pages long. I don't know why the author decided to hide his expertise and experience behind a few URLs pointing to available monitoring tools. The book would significantly benefit from more specific examples; perhaps a large-scale Web caching bibliography. The index is surprisingly incomplete considering currently available indexing tools.

Web Caching is definitely missing a chapter on content delivery networks (CDNs). The author claims that CDNs lack proven results. Perhaps total unavailability of all major news sites that were not using CDNs following the terrorist attacks in the US will convince him otherwise.

Web Caching's strongest points are breadth of coverage and fearless attempts to discuss controversial and even nontechnical issues. In the absence of competition, this is the best reference book on caching today. To stay on top, however, O'Reilly needs to do significantly better editing for the second edition as well as prompt Duane to provide in-depth treatment of key subjects.

Disclaimer: my objectivity in this review was in no significant way affected by the fact that Duane and I have been working together for the past three years and own the same company.

BUILDING SECURE SOFTWARE: HOW TO AVOID SECURITY PROBLEMS THE RIGHT WAY

Gary McGraw and John Viega

Boston, MA: Addison-Wesley Professional Computing Series, 2001.Pp. 528.
ISBN: 0-201-72152-X

Reviewed by Ray Schneider
ray@hackfoo.net

In the world of interconnected computers, whether on LANs or WANs, it has never been more important than now to have secure systems. The authors believe that it all starts with software source code; without source code written with security in mind, none of the rest matters.

Viega and McGraw break *Building Secure Software* down into two logical sections. The first section, consisting of several chapters, successfully introduces the reader to security by covering technologies, goals, as well as risk management.

Readers who make decisions about security in the development of software will find the first chapters enlightening.

There are many references to online sources of security information. The authors mention mailing lists such as BUGTRAQ and the RISKS Digest. They also cover things like "Penetrate and Patch" and the "Art of Engineering." The authors introduce the reader to ideas such as "The Common Criteria," which is an ISO standard that has its roots in the DOD and the Orange Book.

The authors continue initiating the reader into security by examining a few basic ideas: keep it simple, be reluctant to trust, and fail securely. Viega and McGraw also discuss such popular beliefs as the now infamous Eric Raymond quote, "given enough eyeballs, all bugs are shallow." The reader is also introduced to issues surrounding full disclosure and open source. This section of the book looks at these ideas with a critical eye, noting the potential problems that arise from following the popular mojo without considering the effects.

The later chapters of *Building Secure Software* get down to the nitty-gritty of it: source code. Readers will explore buffer and heap overflows, race conditions, cryptography, random number generation, and entropy. These topics are covered in detail. Source code is developed, examined, and improved. Viega and McGraw even supply working exploit code and demonstrate how it works. The majority of the source in the book is in C, but there are examples in Java and Perl as well.

As Bruce Schneier says in his foreword, "*Building Secure Software* is a critical tool in the understanding of secure software." I highly recommend this book to anyone responsible for the development of software in the sometimes hostile environment of interconnected systems.