Book Reviews

ELIZABETH ZWICKY, WITH MARK LAMOURINE, TREY DARLEY, AND BRANDON CHING

The Linux Command Line: A Complete Introduction
William E. Shotts, Jr.
No Starch, 2012. 432 pp.
ISBN 978-1-59327-389-7

Some books I like because they fill my personal needs, some because they are good examples of something I have no interest in, and some because I can give them to other people. This book falls into that last category. This is the book that I can give to people who want to know how to do "that UNIX-y stuff you do." It assumes that you are a reasonably bright person with a grasp of how to use a computer, and you want to make the leap from using Linux with a GUI to using Linux from a command line. It introduces you to thinking like a UNIX person, without dragging in lots of history, and covers the most important commands you need to know, with a big helping of bash scripting.

Careful selection of topics keeps this down to a reasonable size. That means making lots of decisions I fully support, such as deciding to only cover Linux, and only modern distributions at that. Keeping the focus relatively narrow makes a book that's much more readable and usable. You're not forever skipping special cases. I am sad that this approach means that awk is only mentioned in passing, but if I'm going to support the drawing of lines, I'm going to have to live with some authorial choices that differ from mine.

I've been waiting for this book for quite a while, and will be enthusiastically pressing it on several people.

Beginning Python: Using Python 2.6 and Python 3.1
James Payne
Wrox, 2010. 558 pp.
ISBN 978-0-470-41463-7

Head First Python
Paul Barry
O'Reilly, 2011. 445 pp.
ISBN 978-1-449-38267-4

Learning Python, 4th Edition
Mark Lutz
O'Reilly, 2009. 1140 pp.
ISBN 978-0-596-15806-4

The Quick Python Book, 2d Edition
Vernon L Ceder, Daryl K. Harms, and Kenneth McDonald
Manning, 2010. 322 pp.
ISBN 978-1-935182-20-7

If somebody had given me column A, with the book titles complete with series names and subtitles, and column B, with an accurate description of what each one covers, and asked me to match them up, I would never have succeeded. I found this group of books both startlingly diverse and oddly titled.

Beginning Python is in the "Programmer to Programmer" series. It also starts with a description of how programming a computer differs from using a computer, and spends pages of its chapter on variables in a discussion of what a variable is. On the other hand, lambda functions appear not long after, and shortly after that you have left Python itself to gallop through topics that drag in extra protocols and topics, ranging from file typing and file system traversal through XML parsing, and on to creating your own fully functioning Web server with a database backend. (Input sanitization, however, is out of scope, so it comes with XSS and SQL injection vulnerabilities.) The Python it teaches is 2.6 with 3.1 enhancements; it uses 2.6 idioms, not 3.1 idioms. I wouldn't recommend it to anybody, and I'd particularly advise against it for anybody who is just learning to program. The example of quotes, which illustrates single, double, and triple quotes with something that's either a single quote, two single quotes, and three single quotes, or a single quote, a double quote, and some punctuation mark I've never seen before, is particularly problematic, especially since it is immediately followed by examples which use triple double quotes.

If you want a rapid introduction to Python for an experienced programmer, I'd suggest *The Quick Python Book* instead. It covers the basics of Python, plus some of the key libraries (regular expressions, Tkinter, pickles, shelves) for Python 3 and Python 2. It does so with enough Monty Python references to suggest that the authors get the Python mindset, but not an unbearable number. (Yes, reviewing Python books will, perforce, involve evaluating them on the number of Monty Python references. It is as inescapable as the Spanish inquisition.) If you do not already understand some programming language—preferably an object-oriented one—you will not find it a rewarding experience.

I have not yet found a book I'd recommend as a Python introduction for your average person new to programming. *Learning Python* is only a reasonable introduction for somebody with a computing background and a burning desire for completeness. Its introductory chapter does not attempt to introduce you to programming as a concept, but it does list all the major varieties of Python implementations and explain them. You get to "What is Jython?" before you get to "Hello, world." It's a very complete introduction to Python, taking 3 as its point of reference but with information on 2, the differences, and how to code portably. It hews quite carefully to the language itself, avoiding more than the briefest of brushes with common libraries. It's a good, readable language reference, and if you like learning languages systematically, it's an unusually good example of a careful guide to the whole language.

Learning Python should put the other books' lengths in context. It takes a bit over a thousand pages to do a nice, thorough job of explaining the language, just the language, with explanations of the idioms, nice clear examples, and plenty of whitespace, but no major detours and the assumption that you already understand all the underlying concepts. *Quick Python* covers that territory, plus common libraries, in a third the space. *Beginning Python* does it in about a fifth the space, and tries to begin with fewer assumptions.

And then there's *Head First Python*, which I like better than *Beginning Python* even though it is even more of a breathless gallop. In fewer pages with more pictures, it not only walks you through creating your own fully functioning Web server with a database backend (and no input sanitization), it also has you create an Android app and move your Web server onto Google Apps. On the other hand, it does a believable job of explaining the things it does explain, and it makes no pretense to have taught you how these things work. It teaches a number of general programming concepts (not just why objects and exceptions are good ideas, but also some concepts in software design), and it explicitly walks the reader through a number of debugging situations, which is important for novices. Like *Learning Python*, its audience as a book to learn from is a relatively narrow one, but the right person will find it a fun and educational ride. But please, please, do not decide that it is a good idea to build your own fully functioning safety-free Web server, especially with a database backend.

Seven Languages in Seven Weeks

Bruce A. Tate

The Pragmatic Programmers LLC, 2010. 300 pp.
http://pragprog.com/book/btlang/seven-languages-in-seven-weeks
ISBN 978-1-93435-659-3

When I was a freshman in college, I learned seven programming languages. Computer concepts were taught in Pascal. Engineering was in FORTRAN and VAX and 68000 Assembly. Business used COBOL. Artificial Intelligence research was done in LISP and Prolog. I have always been glad that I had that grounding in the variety of ways it is possible to express a problem.

Seven Languages in Seven Weeks offers a similar survey of modern programming languages and language concepts. The creators of these languages each feel that there's something that needs to be expressed and that no other language they know does quite what they want. Tate sets out to show what makes each one special. He's chosen Ruby, Io, Prolog, Scala, Erlang, Clojure, and Haskell. Except for Ruby, most of these will be obscure or unknown to ordinary mainstream coders.

Tate's introduction is very clear about what this book is not: it's not a tutorial or an installation guide. It's not complete or comprehensive. He didn't pick the most popular or most academically acclaimed languages. He apologizes up front to those whose favorite working language isn't included, and explains that he was not interested in producing a "Best of" book. He chose a set of languages which covers the range of current practice. His goal is to explore the significant features of each language, how those help express different ideas clearly and concisely.

The book is divided into a section for each language. The introduction to each section provides the resources and information needed to install the language and to begin interacting or coding. Each section is further broken down into single-day sessions. Tate knows you have real work to do, so each section only contains three days.

The daily sessions start with the common language constructs: variables, types, logic, flow control, and so on. Tate glosses the basics and highlights how each language is special. By the third day, you're deep into the core concepts that make each language unique. Each day ends with a summary of the key concepts and a set of exercises to help you explore for yourself and to set them in your mind. The sections conclude with a wrap-up of the significant features and a little discussion of why they're important.

The book closes with a summary of the families of modern programming concepts and how each of these languages fits into those families. Tate highlights each language's strength, but he doesn't shy away from exposing the warts or showing how one problem or another might not be suited to a given language.

Tate's style is conversational and tutorial. He writes as if he's sitting down with you to show you something cool. He opens each day with some kind of informal anecdote or metaphor that leads to the day's topic. His preparation has included interviews with the language writers or researchers, and in some cases he includes portions of his interview if it highlights the character or taste of the language he's teaching. In at least one case he gets the author of a language to say what he'd most like to change if he could go back and start again.

When you've finished with this book, you should have a clear understanding of some of the more esoteric concepts of current programming languages, and some sense of the flavor of each of the individual languages. This book may be frustrating to someone who's not already familiar with at least a couple of programming languages. I'd steer away from it if your interest is solely in writing application code in any one of them.

I like exploring and understanding the capabilities of different programming languages, even ones I don't expect to use. There's no example in any of these sections that could not be implemented using one of the other languages. What I enjoy is seeing the elegance that each one brings to solving a problem. I suspect I'll pass it on to friends who also like that kind of thing.

—Mark Lamourine

The Art of Readable Code: Simple and Practical Techniques for Writing Better Code

Dustin Boswell and Trevor Foucher

O'Reilly Media Inc., 2012. 190 pp.

ISBN 978-0-596-80229-5

This is my first experience with an O'Reilly book from the "Theory in Practice" series. This series tries to "impart the knowledge and wisdom of leading-edge experts" (http://shop.oreilly.com/category/series/theory.do). *The Art of Readable Code* does feel like a series of lessons or conversations with a colleague or mentor. The authors claim that many of the examples come from their own real applications.

The Art of Readable Code opens by making a case that code should be written with the human reader in mind. Anyone who's ever read someone else's code (or even their own after a time) should be able to get behind that.

Sprinkled throughout the book are a set of "key ideas." Each one relates to the clarity of the style or structure of the code. They range from choosing good names to knowing your libraries. They also include a couple of examples of traditional structural refactoring. Some of this may sound quaint, but the authors illustrate their points in practical ways.

The first three sections cover cosmetic and aesthetics, then logic and branching structures, and, finally, application structure.

There is a fourth section with two unrelated chapters. The first makes a case for writing tests that can be read and that, when they fail, indicate clearly what failed. They also include a remarkably non-trivial application and work through three phases of development.

In most books I don't look at the table of contents much after I begin reading, but in this one the chapter headings make the best summary of those key concepts. It would have been nice to see a cheat sheet or a one- or two-page compact summary of the key ideas.

I've been coding for long enough that there's not a lot here that's new to me, but I did pick up a few tips, and the book presents the ideas in a concise and coherent way. For someone just starting out or who is interested in approaching coding for readability in a systematic way, there's something here for you that I haven't seen anywhere else.

My bookshelf is made up mostly of pure references. I have a few classics which don't get much use, but which I don't feel I can part with. I think this one may fit between those two groups. I won't be looking up function calls, but I can imagine scanning it again when I find myself facing something ugly.

—Mark Lamourine

TCP/IP Illustrated, Volume 1, 2d Edition: The Protocols

Kevin R. Fall and W. Richard Stevens

Addison-Wesley, 2011. 1017 pp.

ISBN 978-0-321-33631-6

There's no shortage of technical books. Most quickly fade in value due to the constant churn of innovation. A select few stand out, forming something like a canon of computer science. It is a testament to W. Richard Stevens's depth of knowledge and communication style that after nearly two decades people still refer to his books. Kevin R. Fall had big shoes to fill when he undertook the ambitious task of producing this updated edition. (Fall is certainly no slouch himself, having served on both the Internet Architecture Board and IETF.) The result is impressive, a true labor of love. It remains true to the spirit of the original while bringing it up to date.

Fall leads the reader gently up the OSI stack, from media layer framing all the way up to DNSSEC and TLS. He assumes a certain level of innate intelligence in his reader but tries hard not to assume much knowledge about TCP/IP. The text incorporates fascinating historical notes, from the ARPANET days to the present, which illuminate both the human politics and technical drivers for change.

One major difference between this and the former edition is how much material Fall elected to remove. The first edition was, in some respects, wider in scope, addressing such topics as NFS, SNMP, SMTP, and dynamic routing protocols. Fall has focused exclusively on core Internet protocols. One might well object that dynamic routing protocols *are* core but, as Fall explains in his preface, there's a world of difference between RIP and BGP/OSPF, and to properly treat the latter would have made this already sizable tome an unreadable doorstop.

While a good bit of material has been elided from this new edition, much has been added. The core protocols have substantially evolved over the past two decades. Fall has done a great service to his readers in assessing those changes. He's essentially read a great pile of RFCs, distilled the essence, and highlighted further reading on topics most relevant to you.

As in the first edition, this incorporates countless packet traces (both tcpdump and wireshark) to illustrate what's going down on the wire. On the inside front cover are three example network diagrams: a home network, a coffee house, and an enterprise. Fall uses these throughout the book, and it proves an effective trope. Back in Stevens's day, there were some pretty stark differences between different TCP/IP stacks. Things have improved, but Fall keeps to Stevens's penchant for mixing traces from different OSes (OS X, FreeBSD, Windows, and Linux), reflecting the heterogeneity of the real world.

Fall has tried to make each chapter self-contained (each chapter is followed immediately by its footnotes, for example). IPv4 and IPv6 are totally integrated within each chapter (except in a few cases where the topic is only applicable to one or the other, as with ARP vs. Neighbor Discovery). Security, too, is integral to the entire text.

It's worth commenting on the final chapter. Although Fall has made security an integral part of the entire book, his final chapter focuses exclusively on security issues. He starts off with an excellent refresher in crypto, then goes on to deal with EAP, IPsec, PKIs, DNSSEC, TLS, and DKIM. I would buy the book on the basis of this chapter alone.

Some who buy this book will just stick it on a shelf and only refer to it occasionally. But while this is most assuredly a reference book (and an excellent one at that), you definitely *can* read this book in its entirely, and I would argue that you are cheating yourself if you don't. Some material is a bit dense by its very nature, to be sure, but the writing is incisive and engaging. As I write this, we're up to RFC 6528. Nobody has time to read all of that. Who among us isn't constantly skirmishing with networks, be you coder, DBA, researcher, policy wonk, or sysadmin? The network is pervasive. Perhaps, like me, your knowledge of TCP/IP is an amalgamated hodgepodge gained through years of experience. If you take the time to read this book you will fill in your gaps and deepen your understanding of not only the "whats" of the Net but also the crucial whys and hows.

—Trey Darley

Head First HTML5 Programming: Building Web Apps with JavaScript

Eric Freeman and Elisabeth Robson

O'Reilly Media, 2011. 610 pp.

ISBN 978-1449390549

O'Reilly's Head First series is a definite departure from traditional technical publishing methods. Instead of pages and pages of text and code, the Head First series uses images, comedy, and a variety of methods to assist your brain in remembering what it is that you are learning. *Head First HTML 5 Programming: Building Web Apps with JavaScript* is one of the latest in this series and, like its predecessors, it does not fail to provide the reader with ample information in an understandable format.

Weighing in at 610 pages, you might think that *Head First HTML 5* is a bit of overkill for a relatively simple updated Web standard, and you'd be right. However, the extent of what is possible in HTML 5 is highly dependent on associated technologies such as JavaScript. Thus, the vast majority of this beefy text is actually focused on how JavaScript, in combination with HTML 5, can be used to usher in a new generation of Web applications and features.

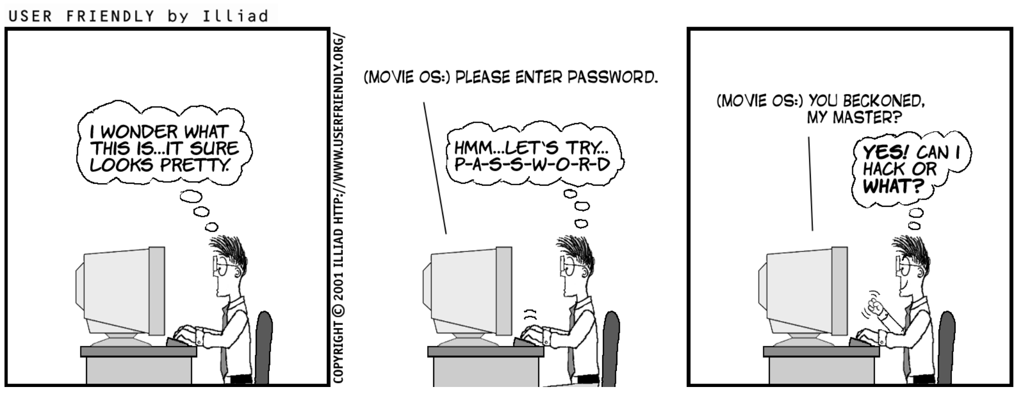
In fact, of the ten chapters in the book, all but one are focused primarily on JavaScript. The book opens with a history and general overview of HTML 5 and how the HTML standard has come to be what it is today. The next three chapters offer a crash course in JavaScript. The overview provided does assume some previous programming experience and focuses primarily on DOM parsing, events and handlers, and functions and objects. Chapters 5 through 9 provide a sort of "greatest hits" for Web applications, covering such topics as geolocation, canvas, AJAX and JSON, and video/media. The book wraps up with coverage of local Web storage and Web workers, for those beefy applications.

As with a lot of instructional texts, each new concept is supported by the construction of a dedicated mock project. All code samples in the book are concise, well explained, and relevant. One of the neat things about the Head First series is that visually, important gotchas and side notes are made easy to identify and remember and do much to help you understand what you are learning.

There was no single chapter that I thought was better than the rest. This is one of those rare books that I found to be well written and on target throughout. If I had to raise a complaint at all, it would be that all the examples are in standard JavaScript, as opposed to a JavaScript library such as jQuery. JavaScript libraries are so ubiquitous in the Web development community that it seems very little new development is done outside of them.

Head First HTML 5 is a book well suited to be on almost any Web developer's bookshelf. There is definitely something in here for everybody, from the junior developer to the expert. In the new era of HTML 5, JavaScript is no longer an option, it is a necessity, and *Head First HTML 5 Programming: Building Web Apps with JavaScript* offers a solid foundation.

—Brandon Ching


USER FRIENDLY by Illiad
COPYRIGHT © 2001 ILLIAD HTTP://WWW.USERFRIENDLY.ORG/
I WONDER WHAT THIS IS...IT SURE LOOKS PRETTY.
(MOVIE OS:) PLEASE ENTER PASSWORD.
HMM...LET'S TRY... P-A-S-S-W-O-R-D
(MOVIE OS:) YOU BECKONED, MY MASTER?
YES! CAN I HACK OR WHAT?