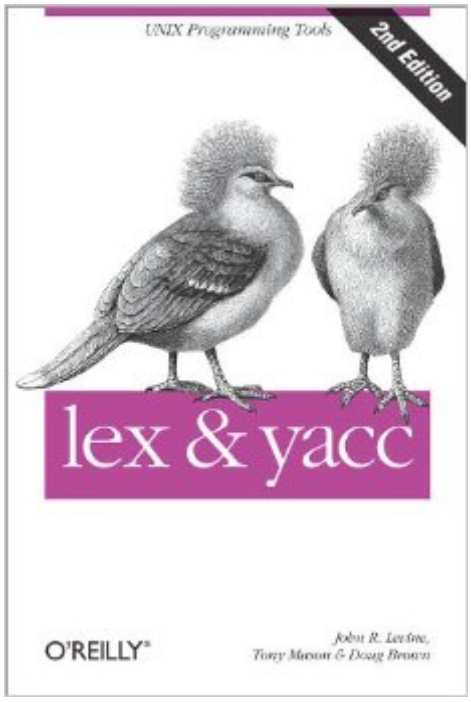
Lex & Yacc



Synopsis

This book shows you how to use two Unix utilities, lex andyacc, in program development. These tools help programmers build compilers and interpreters, but they also have a wider range of applications.The second edition contains completely revised tutorial sections for novice users and reference sections for advanced users. This edition is twice the size of the first and has an expanded index.The following material has been added:Each utility is explained in a chapter that covers basic usage and simple, stand-alone applicationsHow to implement a full SQL grammar, with full sample codeMajor MS-DOS and Unix versions of lex and yacc are explored in depth, including AT&T lex and yacc, Berkeley yacc, Berkeley/GNU Flex, GNU Bison, MKS lex andyacc, and Abraxas PCYACC

Book Information

File Size: 614 KB

Print Length: 388 pages

Simultaneous Device Usage: Unlimited

Publisher: O'Reilly Media; 2 edition (October 1, 1992)

Publication Date: October 19, 2012

Sold by:Â Digital Services LLC

Language: English

ASIN: B009THDEBC

Text-to-Speech: Enabled

X-Ray: Not Enabled

Word Wise: Not Enabled

Lending: Not Enabled

Enhanced Typesetting: Not Enabled

Best Sellers Rank: #178,564 Paid in Kindle Store (See Top 100 Paid in Kindle Store) #25 inÂ Books > Computers & Technology > Software > Utilities #39 inÂ Kindle Store > Kindle eBooks > Computers & Technology > Operating Systems > Unix #68 inÂ Books > Computers & Technology > Operating Systems > Unix

Customer Reviews

As with several other O'Reilly books, I found Lex & Yacc to be maddeningly uneven. The approach is to give a too-brief synopsis of the tool, then illustrate its use using a very specific example that, one suspects, is merely the handiest project the authors had available.I had a fair bit of

programming experience when I bought the book, but none with Lex or Yacc. Some fundamental questions came up during the course of my muddling through, and these were left unanswered. I actually got more insight into these tools from a ~20-page web site on the topic.The reference chapters are organized alphabetically ("ambiguities & conflicts", "bugs", ..., "%ident declaration"), and in a way that does not help someone who is looking for a specific answer (in trying to find out about the possibility of more than one parser in a program, who would think to look under 'v' for "variant and multiple grammars"?). These 'reference chapters' seemed more like a place to dump the information not discussed elsewhere.Maybe it's a lost cause, finding a comprehensive, well-written introduction to such an arcane topic, but I'm still looking.

To keep it simple, the book "Introduction to Compiler Construction in UNIX" introduces and explains LEX/YACC far better than this book. It uses a more realistic example and shows the error handling in more detail. This book is ok for a quick intro, but for a 'real" user, refer to the book I mentioned above.

This book was disappointing. I had hoped for a tutorial and reference on Lex/Yacc Flex/Bison for building language recognisers, but in the tradition of "yet another boring and useless reverse polish notation" calculator - it gives us a desktop infix expression calculator. (How could I have possibly guessed?)The book presents some interesting material for those who want to parse SQL, but if you're eager to learn about translating programming languages, this book is nearly useless. It gives superficial and confusing treatment of language constructs like "if-then-else" statements, and gives only an exercise for recognising a function call and "playing it back." (I feel these are cop-outs.) It also fails to explain clearly how to construct unambiguous grammars, or use facilities for operator precedence or whatver to control the ambiguity.To be fair, I am guessing this was meant as a reference for Lex/Yacc for those who already know how to construct language recognisers and have some knowledge of using these tools. But then why bother to give especially bad tutorial material in the early chapters if this were the case?This books is also in need of a new edition.

There is too much repetition of basic ideas in the first few chapters of the book, and not enough coverage of more advanced topics (like how to use marker nonterminals, how to use $$ constructs, etc).The discussion of shift/reduce and reduce/reduce conflicts in chaper 8 is pretty good though and would make a fine introduction for a beginner wanting to learn /basic/ concepts.Good error handling is definitely a black art, but I still would have liked to have been given more information

about it in chapter 9. The examples were also a bit too soft.Chapters 4 and 5 have some interesting (and highly unusual)examples for scanning and parsing applications, but they do not show off many of the advanced capabilites of lex and yacc. What this book really needs are couple of examples that demonstrate how to overcome classic scanning and parsing horrors (like how to do type checking in, say, C); a chapter like this instead of one of chapters 4 or 5 would be great.Even today lex and yacc are very important tools in the computer scientist's toolkit. They were designed 25 (or so) years ago, but /real/ documentation is still nonexistant. This means that unfortunately, this book is one of the best.I think that the FSF's Bison manual is much better value for money. It also does not cover advanced topics in enough depth, but what is does explain, it explains quite clearly.

In general this book is very good, and I would highly recommended it for anybody using tools like lex and yacc. It covers all the areas (both good and bad) of these tools. However, I find that the book is frustrating and incomplete in two ways :-(1) when it comes to solving some of the problems reported by such tools as lex and yacc, there is not enough of examples, especially less-trivial examples, and some of the more potentially obscure problems that can occur. (2) it is very frustrating that the page numbers listed in the index are out by 1 or 2 pages. In this era of technology this should not happen.

Firstly there are not many books on Lex & Yacc.Unix programming by Kernighan & Pike providessome necessary information but it is not sufficientto write a complex parser or scanner. In these circumstancesthis book acts like a God given gift for compiler developers.The examples which introduce the tools are very good.But the theory behind these examples is not well explained.Probably NFA/DFA concepts add more strenght to thebook. Adding more and more concepts and exampleswill make this as a unique book in the Market.Many guys who desire jobs as compiler developersfollow this book. So some contemporary examples thatI expect from future editions of this book areA) Parsing a Gate level designB) A small simulator which simulates an analog/digital designC) An assembler etcD) Relational algebra tutor etc.Hope the authors can consider this request.

lex & yacc