<<flex   bison>>

<<flex   bison>>

13   ISBN   9787564119324

10   ISBN   7564119322

2010-1

John Levine

271

PDF

http://www.tushu007.com

<<flex   bison>>

Flex and bison are tools designed for writers of compilers and interpreters, althoughthey are also useful for many applications that will interest noncompiler writers. Anyapplication that looks for patterns in its input or has an input or command language isa good candidate for flex and bison. Furthermore, they allow for rapid applicationprototyping, easy modification, and simple maintenance of programs. To stimulateyour imagination, here are a few things people have used flex and bison, or their predecessors lex and yacc, to develop:The desktop calculator bcThe tools eqn and pic, typesetting preprocessors for mathematical equations and complex picturesMany other "domain-specific languages" targeted for a particular application PCC, the Portable C Compiler used with many Unix systemsFlex itselfA SQL database language translatorScope of This BookChapter 1, Introducing Flex and Bison, gives an overview of how and why flex and bisonare used to create compilers and interpreters and demonstrates some simple applica-tions including a calculator built in flex and bison. It also introduces basic terms weuse throughout the book.Chapter 2, Using Flex, describes how to use flex. It develops flex applications that countwords in files, handle multiple and nested input files, and compute statistics on C programs.Chapter 3, Using Bison, gives a full example using flex and bison to develop a fullyfunctional desktop calculator with variables, procedures, loops, and conditional expressions. It shows the use of abstract syntax trees fASTs   , powerful and easy-to-usedata structures for representing parsed input.Chapter 4, Parsing SQL, develops a parser for the MySQL dialect of the SQL relationaldatabase language.

Page 3

flex   bison(   )   Linux   Unix
                        flex   bison
flex   bison   O' Reilly   lex & yacc
                20   flex   bison   Unix
flex   bison   Linux   Unix

        flex   bison
                    ·   (address syrltax crunching)·
                                ·   ·
                                ·   SQI   ·
        (   )   (Iexer)   (parser)   JG LR   C++

<<flex   bison>>

John Levine, Taughannock Networks   20   lex & yacc   qmail
        O'Reilly

<<flex 与 bison>>

<<flex bison>>

Conflict? IF/THEN/ELSE (Shift/Reduce) Loop Within a Loop (Shift/Reduce) Expression Precedence (Shift/Reduce) Limited Lookahead (Shift/Reduce or Reduce/Reduce) Overlap of Alternatives (Reduce/Reduce) Summary Exercises 8. Error Reporting and Recovery Error Reporting Locations Adding Locations to the Parser Adding Locations to the Lexer More Sophisticated Locations with Filenames Error Recovery Bison Error Recovery Freeing Discarded Symbols Error Recovery in Interactive Parsers Where to Put Error Tokens Compiler Error Recovery Exercises 9. Advanced Flex and Bison Pure Scanners and Parsers Pure Scanners in Flex Pure Parsers in Bison Using Pure Scanners and Parsers Together A Reentrant Calculator GLR Parsing GLR Version of the SQL Parser C++ Parsers A C++ Calculator C++ Parser Naming A C++ Parser Interfacing a Scanner with a C++ Parser Should You Write Your Parser in C++ ? Exercises Appendix: SQL Parser Grammar and Cross-Reference Glossary Index

<<flex   bison>>

A bison specification has the same three-part structure as a flex specification.   Flexcopied its structure from the earlier lex, which copied its structure from yacc, the predecessor of bison.   The first section, the definition section, handles control informationfor the parser and generally sets up the execution environment in which the parser willoperate. The second section contains the rules for the parser, and the third section isC code copied verbatim into the generated C program.Bison creates the C program by plugging pieces into a standard skeleton file. The rulesare compiled into arrays that represent the state machine that matches the input tokens.The actions have the SN and @N values translated into C and then are put into a switchstatement within yyparse0that runs the appropriate action each time there's a reduction. Some bits of the skeleton have multiple versions from which bison chooses depending on what options are in use; for example, if the parser uses the locations feature,it includes code to handle location data.In this chapter we take the simple calculator example from Chapter 1 and extend itsignificantly. First, we rewrite it to take advantage of some handy bison shortcuts andchange it to produce a reusable data structure rather than computing the values on thefly. Later, we'll add more complex syntax for loops and functions and show how toimplement them in a simple interpreter.One of the most powerful data structures used in compilers is an abstract syntax tree   AST   . In Chapter 1 we saw a parse tree, a tree that has a node for every rule used toparse the input string. In most real grammars, there are rules that exist to managegrouping but that add no meaning to the program. In the calculator example, the rulesexp: term and term: factor exist only to tell the parser the relative precedence of theoperators. An AST is basically a parse tree that omits the nodes for the uninterestingrules.

<<flex   bison>>

"   John

"   ——Joel E. Denny bison

lex   yacc

<<flex   bison>>

flex   bison(   )

<<flex   bison>>

PDF

:http://www.tushu007.com