The Lex & Yacc Page

The asteroid to kill this dinosaur is still in orbit.
- Lex Manual Page

ON THIS PAGE

Overview | Lex | Yacc | Flex | Bison | Tools | Books

OVERVIEW

A compiler or interptreter for a programminning language is often decomposed into two parts:

1. Read the source program and discover its structure.
2. Process this structure, e.g. to generate the target program.

Lex and *Yacc* can generate program fragments that solve the first task.

The task of discovering the source structure again is decomposed into subtasks:

1. Split the source file into tokens (*Lex*).
2. Find the hierarchical structure of the program (*Yacc*).

- A First Example: A Simple Interpreter

LEX

Lex - A Lexical Analyzer Generator

M. E. Lesk and E. Schmidt

Lex helps write programs whose control flow is directed by instances of regular expressions in the input stream. It is well suited for editor-script type transformations and for segmenting input in preparation for a parsing routine.

Lex source is a table of regular expressions and corresponding program fragments. The table is translated to a program which reads an input stream, copying it to an output stream and partitioning the input into strings which match the given expressions. As each such string is recognized the corresponding program fragment is executed. The recognition of the expressions is performed by a deterministic finite automaton generated by Lex. The program fragments written by the user are executed in the order in which the corresponding regular expressions occur in the input stream.

- Online Manual
- PostScript
- Lex Manual Page

YACC

Yacc: Yet Another Compiler-Compiler

Stephen C. Johnson

Computer program input generally has some structure; in fact, every computer program that does input can be thought of as defining an ``input language'' which it accepts. An input language may be as complex as a programming language, or as simple as a sequence of numbers. Unfortunately, usual input facilities are limited, difficult to use, and often are lax about checking their inputs for validity.

Yacc provides a general tool for describing the input to a computer program. The Yacc user specifies the structures of his input, together with code to be invoked as each such structure is recognized. Yacc turns such a specification into a subroutine that han- dles the input process; frequently, it is convenient and appropriate to have most of the flow of control in the user's application handled by this subroutine.

- Online Manual
- PostScript
- Yacc Manual Page

FLEX

Flex, A fast scanner generator

Vern Paxson

flex is a tool for generating scanners: programs which recognized lexical patterns in text. flex reads the given input files, or its standard input if no file names are given, for a description of a scanner to generate. The description is in the form of pairs of regular expressions and C code, called rules. flex generates as output a C source file, `lex.yy.c', which defines a routine `yylex()'. This file is compiled and linked with the `-lfl' library to produce an executable. When the executable is run, it analyzes its input for occurrences of the regular expressions. Whenever it finds one, it executes the corresponding C code.

- Online Manual
- PostScript
- Flex Manual Page
- Download Flex from ftp://prep.ai.mit.edu/pub/gnu/

BISON

Bison, The YACC-compatible Parser Generator

Charles Donnelly and Richard Stallman

Bison is a general-purpose parser generator that converts a grammar description for an LALR(1) context-free grammar into a C program to parse that grammar. Once you are proficient with Bison, you may use it to develop a wide range of language parsers, from those used in simple desk calculators to complex programming languages.

Bison is upward compatible with Yacc: all properly-written Yacc grammars ought to work with Bison with no change. Anyone familiar with Yacc should be able to use Bison with little trouble.

- Online Manual
- PostScript
- Bison Manual Page
- Download Bison from ftp://prep.ai.mit.edu/pub/gnu/

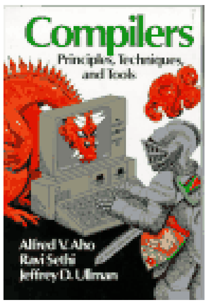
TOOLS

Other tools for compiler writers:

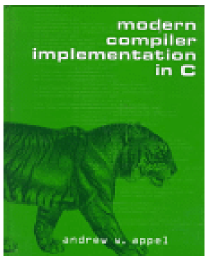
- Compiler Construction Kits
- Lexer and Parser Generators
- Attribute Grammar Systems
- Transformation Tools
- Backend Generators
- Program Analysis and Optimisation
- Environment Generators
- Infrastructure, Components, Tools
- Compiler Construction with Java

BOOKS

Lex & Yacc
John R. Levine, Tony Mason, Doug Brown
Paperback - 366 pages 2nd/updated edition (October 1992)
O'Reilly & Associates
ISBN: 1565920007



Compilers: Principles, Techniques, and Tools
Alfred V. Aho, Ravi Sethi, Jeffrey D. Ullman
Addison-Wesley Pub Co
ISBN: 0201100886



Modern Compiler Implementation in C
Andrew W. Appel, Maia Ginsburg
Hardcover - 560 pages Rev expand edition (January 1998)
Cambridge University Press
ISBN: 052158390X