Chapter 1 & 2; Lex and Yacc, Using Lex

From Lex and Yacc Book

Lex and yacc help you write programs that transform structured input. from a simple text search program that looks for patterns in its input file to a C compiler that transforms a source program into optimized object code.

For a C program, the units are variable names, constants, strings, operators, punctuation, and so forth. This division into units (which are usually called tokens) is known as lexical analysis, or lexing for short. The token descriptions that lex uses are known as regular expressions.

A C compiler needs to find the expressions, statements, declarations, blocks, and procedures in the program. This task is known as *parsing* and the list of rules that define the relationships that Zex C yacc the program understands is a *grammar*

This first section, the definition section, introduces any initial C program code we want copied into the final program. Lex copies the material between "%{" and "%}" directly to the generated C file. Outside of "%{" and "%}", comments must be indented with whitespace for lex to recognize them correctly. The %% marks the end of this section.

The next section is the rules section. Each rule is made up of two parts: a pattern and an action. These patterns are Unix-style regular expressions. The square brackets, "[]", indicate that any one of the characters within the brackets matches the pattem. "[\t]+".

The second part of the rule, the action, is simply a semicolon, a do-nothing C statement.

The next set of rules uses the " | " (vertical bar) action. This is a special action that means to use the same action as the next pattern. Our patterns match any of the verbs in the list. Once we recognize a verb, we execute the action, a C printf statement. The array yytext contains the text that matched the pattern. The last two rules are:

The two that make our lexer work are:

1. Lex patterns only match a given input character or string once.
2. Lex executes the action for the longest possible match for the current input. Thus, lex would see "island as matching our all-inclusive rule because that was a longer match than "is."

The final section is the user subroutines section. The lexer produced by lex is a C routine called yylex(), yylex() won't return until it has processed the entire input.