**LEX**

In computer science, **LEX** is a program that generates lexical analyzers ("scanners" or "lexers"). LEX is commonly used with the YACC parser generator. LEX, originally written by Mike Lesk and Eric Schmidt, is the standard lexical analyzer generator on many Unix systems, and a tool exhibiting its behavior is specified as part of the POSIX standard.

LEX reads an input stream specifying the lexical analyzer and outputs source code implementing the lexer in the C Programming language.

Though traditionally proprietary software, versions of LEX based on the original AT&T code are available as open source, as part of systems such as OpenSolaris and Plan 9 from Bell Labs. Another popular open source version of LEX is Flex, the "fast lexical analyzer".

**Structure of a LEX File**

The structure of a LEX file is intentionally similar to that of a YACC file; files are divided up into three sections, separated by lines that contain only two percent signs, as follows:

*Definition Section*

*%%*

*Rules Section*

*%%*

*C Code Section*

* The **definition** section is the place to define macros and to import header files written in C. It is also possible to write any C code here, which will be copied verbatim into the generated source file.
* The **rules** section is the most important section; it associates patterns with C statements. Patterns are simply regular expressions. When the lexer sees some text in the input matching a given pattern, it executes the associated C code. This is the basis of how lex operates.
* The **C code** section contains C statements and functions that are copied verbatim to the generated source file. These statements presumably contain code called by the rules in the rules section. In large programs it is more convenient to place this code in a separate file and link it in at compile time.

**Usage**

*lex [options] lex-file[.l] [output-file[.pas]]*

Options

1. -v “Verbose:" LEX generates a readable description of the generated lexical analyzer, written to lex-\_le with new extension .lst.
2. -o “Optimize:" LEX optimizes DFA tables to produce a minimal DFA.

**Using LEX with other Programming tools**

Using LEX with parser generators

LEX and parser generators, such as YACC or Bison, are commonly used together. Parser generators use a formal grammar to parse an input stream, something which LEX cannot do using simple regular expressions (LEX is limited to simple finite state automata). However, parser generators cannot read from a simple input stream – they require a series of tokens. LEX is often used to provide the parser generator with these tokens.

LEX and Make

Make is a utility that can be used to maintain programs involving LEX. Make assumes that a file that has an extension of .l is a LEX source file. The make internal macro LFLAGS can be used to specify LEX options to be invoked automatically by make.