Lex (software)

Lex is a computer program that generates lexical analyzers ("scanners" or "lexers").[1][2]

Lex is commonly used with the yacc parser generator. Lex, originally written by Mike Lesk and Eric Schmidt[3] and described in 1975,[4][5] is the standard lexical analyzer generator on many Unix systems, and an equivalent tool is specified as part of the POSIX standard.[6]

Lex reads an input stream specifying the lexical analyzer and writes source code which implements the lexical analyzer in the C programming language.

In addition to C, some old versions of Lex could also generate a lexer in Ratfor.[7]

Lex

Original author(s)	Mike Lesk, Eric Schmidt
Initial release	1975
Repository	bxr.su/OpenBSD/usr.bin/lex/ (http://bxr.su/OpenBSD/usr.bin/lex/)
Written in	C
Operating system	Unix, Unix-like, Plan 9
Platform	Cross-platform
Type	Command

Contents

- Open source
- Structure of a Lex file
- Example of a Lex file
- Using Lex with other programming tools
  - Using Lex with parser generators
  - Lex and make
- See also
- References
- External links

Open source

Although originally distributed as proprietary software, some versions of Lex are now open-source. Open-source versions of Lex, based on the original proprietary code, are now distributed with open-source operating systems such as OpenSolaris and Plan 9 from Bell Labs. One popular open-source version of Lex, called flex, or the "fast lexical analyzer", is not derived from proprietary coding.

Structure of a Lex file

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

- The **definitions** section defines macros and imports 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 associates regular expression patterns with C statements. When the lexer sees text in the input matching a given pattern, it will execute the associated C code.
- 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 linked in at compile time.

Example of a Lex file

The following is an example Lex file for the flex version of Lex. It recognizes strings of numbers (positive integers) in the input, and simply prints them out.

```
/*** Definition section ***/

%{
/* C code to be copied verbatim */
#include <stdio.h>
%}

%%
    /*** Rules section ***/

    /* [0-9]+ matches a string of one or more digits */
[0-9]+  {
            /* yytext is a string containing the matched text. */
            printf("Saw an integer: %s\n", yytext);
        }

.|\n    {   /* Ignore all other characters. */   }

%%
/*** C Code section ***/

int main(void)
{
    /* Call the lexer, then quit. */
    yylex();
    return 0;
}
```

If this input is given to `flex`, it will be converted into a C file, `lex.yy.c`. This can be compiled into an executable which matches and outputs strings of integers. For example, given the input:

```
abc123z.!&*2gj6
```

the program will print:

```
Saw an integer: 123
Saw an integer: 2
Saw an integer: 6
```

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, as Lex is limited to simple finite state automata.

It is typically preferable to have a parser, one generated by Yacc for instance, accept a stream of tokens (a "token-stream") as input, rather than having to process a stream of characters (a "character-stream") directly. Lex is often used to produce such a token-stream.

Scannerless parsing refers to parsing the input character-stream directly, without a distinct lexer.

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.[8]

See also

- Flex lexical analyser
- Yacc
- Ragel
- PLY (Python Lex-Yacc)
- Comparison of parser generators

References

1. Levine, John R.; Mason, Tony; Brown, Doug (1992). *lex & yacc* (https://archive.org/details/lexyacc00levi) (2 ed.). O'Reilly. pp. 1 (https://archive.org/details/lexyacc00levi/page/1)–2. ISBN 1-56592-000-7.
2. Levine, John (August 2009). *flex & bison* (http://oreilly.com/catalog/9780596155988). O'Reilly Media. p. 304. ISBN 978-0-596-15597-1.
3. Lesk, M.E.; Schmidt, E. "Lex – A Lexical Analyzer Generator" (http://dinosaur.compilertools.net/lex/index.html). Retrieved August 16, 2010.
4. Lesk, M.E.; Schmidt, E. (July 21, 1975). "Lex – A Lexical Analyzer Generator" (http://epaperpress.com/lexandyacc/download/lex.pdf) (PDF). *UNIX TIME-SHARING SYSTEM:UNIX PROGRAMMER'S MANUAL, Seventh Edition, Volume 2B*. bell-labs.com. Retrieved Dec 20, 2011.
5. Lesk, M.E. (October 1975). "Lex – A Lexical Analyzer Generator". *Comp. Sci. Tech. Rep. No. 39*. Murray Hill, New Jersey: Bell Laboratories.
6. The Open Group Base Specifications Issue 7, 2018 edition § Shell & Utilities § Utilities § lex (https://pubs.opengroup.org/onlinepubs/9699919799/utilities/lex.html)
7. John R. Levine; John Mason; Doug Brown (1992). *Lex & Yacc* (https://archive.org/details/lexyacc00levi). O'Reilly.
8. "make" (http://www.opengroup.org/onlinepubs/009695399/utilities/make.html). *The Open Group Base Specifications*. The IEEE and The Open Group (6). 2004. IEEE Std 1003.1, 2004 Edition.

External links

- Using Flex and Bison at Macworld.com (http://www.mactech.com/articles/mactech/Vol.16/16.07/UsingFlexandBison/)
- `lex(1) (https://docs.oracle.com/cd/E26505_01/html/816-5165/lex-1.html)` – Solaris 10 User Commands Reference Manual
- `lex(1) (https://9p.io/magic/man2html/1/lex)` – Plan 9 Programmer's Manual, Volume 1