<https://baike.baidu.com/item/lex/8558986>

lex

Lex是LEXical compiler的缩写，是Unix环境下非常著名的工具,主要功能是生成一个词法分析器(scanner)的C源码，描述规则采用[正则表达式](https://baike.baidu.com/item/%E6%AD%A3%E5%88%99%E8%A1%A8%E8%BE%BE%E5%BC%8F" \t "_blank)(regular expression)。

**外文名**

LEXical compiler

**缩    写**

lex

**运行环境**

Unix环境

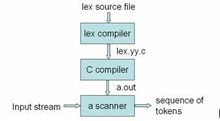
**主要功能**

生成一个词法分析器的C源码

## 生成工具

[编辑](javascript:;)

Lex是LEXical compiler的缩写，是Unix环境下非常著名的工具,主要功能是生成一个词法分析器(scanner)的C源码,描述规则采用[正则表达式](https://baike.baidu.com/item/%E6%AD%A3%E5%88%99%E8%A1%A8%E8%BE%BE%E5%BC%8F" \t "_blank)(regular expression)。描述词法分析器的文件\*.l，经过lex编译后，生成一个lex.yy.c 的文件，然后由C[编译器](https://baike.baidu.com/item/%E7%BC%96%E8%AF%91%E5%99%A8)编译生成一个词法分析器。词法分析器，简单来说，其任务就是将输入

[](https://baike.baidu.com/pic/lex/8558986/0/08b68e527f9e301d0df3e375?fr=lemma&ct=single)

的各种符号，转化成相应的[标识符](https://baike.baidu.com/item/%E6%A0%87%E8%AF%86%E7%AC%A6" \t "_blank)(token)，转化后的标识符 很容易被后续阶段处理。

它被设计用来对输入字符流进行词法处理。它接受一种高级的、面向问题的说明书，并用它匹配字符串中的字符、生成能够识别正则表达式的程序。正则表达式通过用户输入的代码说明书给入。Lex识别这些表达式，并且将输入流分成一些匹配这些表达式的字符串。在这些字符串的分界处，用户提供的程序片段被执行。Lex代码文件将正则表达式和程序片断关联。对每一条输入到由Lex生成程序的表达式，相应的代码片段被执行。

为了完成任务，除了需要提供匹配的表达式以外，用户还需要提供其它代码，甚至是由其他生成器产生的代码。用户提供一般程序设计语言的代码片断完成程序识别表达式。因此，用户自由编写动作时，并不影响其编写高层的表达式语言来匹配字符串表达式。这就避免迫使用户使用字符串语言来进行输入分析时，也必须使用同样的方法来编写字符处理程序，而这样做有时是不合适的。Lex不是完整的语言，但是是一个新语言的生成器，它可以插入到各种不同的被叫做“宿主语言”的程序设计语言中。就像大多数目的语言可以生成在不同计算机硬件上运行的代码，Lex可以生成不同的宿主语言。宿主语言用于Lex生成输出代码，也用于用户插入程序片断。这使得Lex适用于不同的环境和不同的使用者。每一个应用程序可以是硬件、适用于该任务的宿主语言、用户背景和局部接口属性的直接结合。现在，Lex唯一支持的宿主语言是C，尽管Fortran（形式为Ratfor[2]）在过去也被支持。Lex自身存在于UNIX、GCOS和OS/370上；但是Lex生成的代码可以在任何适当的编译器上使用。

Lex将用户输入的表达式和动作actions（在这篇文章中被称作**源代码**）转换为宿主语言；生成的程序叫做**yylex**。**yylex**识别字符流中的表达式（本文称作**输入流**），并且当每一个表达式被检测出来后，输出相应的动作。

过程如图 。

让我们来仔细研究一下这个奇妙的工具吧。先看看Lex文件的结构。 Lex文件结构简单，分为三个部分：

declarations

%%

translation rules

%%

auxiliary procedures

分别是声明，转换规则和其它函数。

声明段包括变量的声明、[符号常量](https://baike.baidu.com/item/%E7%AC%A6%E5%8F%B7%E5%B8%B8%E9%87%8F)的声明和[正则表达式](https://baike.baidu.com/item/%E6%AD%A3%E5%88%99%E8%A1%A8%E8%BE%BE%E5%BC%8F)声明。希望出现在目标C源码中的代码，用%{…%}扩在一起。比如：

%{

#include <stdio.h>

#include "y.tab.h"

typedef char \* YYSTYPE;

char \* yylval;

%}

正则表达式声明如下

/\* regular definitions \*/

delim [ \t\n]ws +letter [A-Za-z]digit [0-9]id ()\*number +(\.+)?(E[+\-]?+}?

这段[正则表达式](https://baike.baidu.com/item/%E6%AD%A3%E5%88%99%E8%A1%A8%E8%BE%BE%E5%BC%8F" \t "_blank)描述识别数(number)、[标识符](https://baike.baidu.com/item/%E6%A0%87%E8%AF%86%E7%AC%A6" \t "_blank)(id)的“规则”。过一会我们再细说正则表达式。

规则段是由正则表达式和相应的动作组成的。

p1 p2 …… pn

值得注意的是，lex 依次尝试每一个规则，尽可能地匹配最长的输入流。如果有一些内容根本不匹配任何规则，那么 lex 将只是把它拷贝到标准输出。比如

%%A {printf("you");}AA {printf("love ");}AAAA {printf("I ");}%%

编译后运行一下，

$ ./sample3AAAAAAAI love you

可以看出lex的确按照最长的规则匹配。

[程序段](https://baike.baidu.com/item/%E7%A8%8B%E5%BA%8F%E6%AE%B5)部分放一些扫描器的其它模块，比如一些动作执行时需要的模块。也可以在另一个[程序文件](https://baike.baidu.com/item/%E7%A8%8B%E5%BA%8F%E6%96%87%E4%BB%B6)中编写，最后再链接到一起。 生成C代码后，需用C的[编译器](https://baike.baidu.com/item/%E7%BC%96%E8%AF%91%E5%99%A8" \t "_blank)编译。连接时需要指定链接库。gcc的连接参数为 -ll。

[正则表达式](https://baike.baidu.com/item/%E6%AD%A3%E5%88%99%E8%A1%A8%E8%BE%BE%E5%BC%8F)可以描述有穷状态[自动机](https://baike.baidu.com/item/%E8%87%AA%E5%8A%A8%E6%9C%BA" \t "_blank)(finite automata)接受的语言，也就是定义一个可以接受的串的集.lex中用到的正则表达式的一些规则如下：

[转义字符](https://baike.baidu.com/item/%E8%BD%AC%E4%B9%89%E5%AD%97%E7%AC%A6)（也称操作符）：

" \ [ ] ^ - ? . \* + | ( ) $ / { } % < >

这些符号有特殊含义，不能用来匹配自身。如果需要匹配的话，可以通过引号(")或者转义符号(\)来指示。比如

C"++" C\+\+

都可以匹配C++。

非转义字符：所有除了转义字符之外的字符都是非转义字符。一个非转义字符可以匹配自身。比如

integer

匹配文本中出现的integer。

[通配符](https://baike.baidu.com/item/%E9%80%9A%E9%85%8D%E7%AC%A6)：通配符就是.（dot），可以匹配任何一个字符。

[字符集](https://baike.baidu.com/item/%E5%AD%97%E7%AC%A6%E9%9B%86)：用一对[]指定的字符构成一个字符集。比如[abc]表示一个字符集，可以匹配a、b、c中的任意一个字符。使用 – 可以指定范围。比如[a-z]表示可以匹配所有小写字母的字符集。

重复：

\* 任意次重复+ 至少一次的重复，相当于xx\*? 零次或者一次

选择和分组：|符号表示选择，二者则一；括号表示分组，括号内的组合被看作是一个原子。比如(ab|cd)匹配ab或者cd。

## Lex源代码

[编辑](javascript:;)

一般的Lex源代码格式为

{definitions}

%%

{rules}

%%

{user subroutines}

而definitions和user subroutines经常被忽略。第二个%%是可选择的，但是第一个必须存在以标记rules的开始。因此最简单的Lex程序是

%%

（没有definitions和rules），这个程序输入将不加修改地复制到输出。

由上面的Lex程序轮廓可知，**规则（rules）**反映了用户的控制；它是一个表格，左侧是正则**表达式（regular expressions）**（参见第3节），而右侧是**动作（actions）**，当表达式被识别出以后，动作的程序片断被执行。所以，一个单独的规则可能是

|  |  |
| --- | --- |
| integer | printf("found keyword INT"); |

它用于在输入流中寻找字符串中的**integer**，找到后输出“found keyword INT”。在这个例子中，主程序为C语言并且用C库函数**printf**打印字符串。用第一个出现的空白符或者制表符作为表达式的结束标记。如果action仅仅是一条简单的C表达式，那么它可以直接写在这一行的右侧；如果是复合表达式或者包含了很多行，则必须用大括号括起来。作为一个更有用的例子——用来将一些英式拼写转换为美式拼写——其词法分析器应该以如下规则开始：

|  |  |
| --- | --- |
| colour | printf("color"); |
| mechanise | printf("mechanize"); |
| petrol | printf("gas"); |

这些规则是不够强大的，比如pertroleum应该变为gaseum；一种处理它的方法将在下文中予以介绍。

## 警告和缺陷

[编辑](javascript:;)

有一些病态的表达式会使由表格转化的确定的自动机成指数增长；幸运的是，这样的情况很少见。

REJECT没有重复扫描输入；而是记住先前扫描的结果。这意味着如果一条规则需要回退发现的上下文，并且REJECT被执行了，用户将不能使用**unput**来改变输入流中的后续字符。这是对用户操作后续输入的唯一限制。

<https://en.wikipedia.org/wiki/Flex_(lexical_analyser_generator)>

**Flex** (fast [lexical analyzer](https://en.wikipedia.org/wiki/Lexical_analyzer) generator) is a [free and open-source software](https://en.wikipedia.org/wiki/Free_and_open-source_software) alternative to [lex](https://en.wikipedia.org/wiki/Lex_programming_tool" \o "Lex programming tool).[[2]](https://en.wikipedia.org/wiki/Flex_(lexical_analyser_generator)#cite_note-2) It is a [computer program](https://en.wikipedia.org/wiki/Computer_program) that generates [lexical analyzers](https://en.wikipedia.org/wiki/Lexical_analysis) (also known as "scanners" or "lexers").[[3]](https://en.wikipedia.org/wiki/Flex_(lexical_analyser_generator)#cite_note-3)[[4]](https://en.wikipedia.org/wiki/Flex_(lexical_analyser_generator)#cite_note-4) It is frequently used as the lex implementation together with [Berkeley Yacc](https://en.wikipedia.org/wiki/Berkeley_Yacc) [parser generator](https://en.wikipedia.org/wiki/Parser_generator) on [BSD](https://en.wikipedia.org/wiki/Berkeley_Software_Distribution)-derived operating systems (as both lex and yacc are part of [POSIX](https://en.wikipedia.org/wiki/POSIX)),[[5]](https://en.wikipedia.org/wiki/Flex_(lexical_analyser_generator)#cite_note-5)[[6]](https://en.wikipedia.org/wiki/Flex_(lexical_analyser_generator)#cite_note-6)[[7]](https://en.wikipedia.org/wiki/Flex_(lexical_analyser_generator)#cite_note-7) or together with [GNU bison](https://en.wikipedia.org/wiki/GNU_bison) (a version of [yacc](https://en.wikipedia.org/wiki/Yacc" \o "Yacc)) in [\*BSD ports](https://en.wikipedia.org/wiki/Ports_collection)[[8]](https://en.wikipedia.org/wiki/Flex_(lexical_analyser_generator)#cite_note-8) and in Linux distributions. Unlike Bison, flex is not part of the [GNU Project](https://en.wikipedia.org/wiki/GNU_Project) and is not released under the [GNU General Public License](https://en.wikipedia.org/wiki/GNU_General_Public_License).[[9]](https://en.wikipedia.org/wiki/Flex_(lexical_analyser_generator)#cite_note-9)

## Contents

 [hide]

* [1History](https://en.wikipedia.org/wiki/Flex_(lexical_analyser_generator)#History)
* [2Example lexical analyzer](https://en.wikipedia.org/wiki/Flex_(lexical_analyser_generator)#Example_lexical_analyzer)
* [3Internals](https://en.wikipedia.org/wiki/Flex_(lexical_analyser_generator)#Internals)
* [4Issues](https://en.wikipedia.org/wiki/Flex_(lexical_analyser_generator)#Issues)
  + [4.1Time complexity](https://en.wikipedia.org/wiki/Flex_(lexical_analyser_generator)#Time_complexity)
  + [4.2Reentrancy](https://en.wikipedia.org/wiki/Flex_(lexical_analyser_generator)#Reentrancy)
  + [4.3Usage under non-Unix environments](https://en.wikipedia.org/wiki/Flex_(lexical_analyser_generator)#Usage_under_non-Unix_environments)
  + [4.4Using flex from other languages](https://en.wikipedia.org/wiki/Flex_(lexical_analyser_generator)#Using_flex_from_other_languages)
* [5Flex++](https://en.wikipedia.org/wiki/Flex_(lexical_analyser_generator)#Flex++)
* [6See also](https://en.wikipedia.org/wiki/Flex_(lexical_analyser_generator)#See_also)
* [7References](https://en.wikipedia.org/wiki/Flex_(lexical_analyser_generator)#References)
* [8Further reading](https://en.wikipedia.org/wiki/Flex_(lexical_analyser_generator)#Further_reading)
* [9External links](https://en.wikipedia.org/wiki/Flex_(lexical_analyser_generator)#External_links)

## History[[edit](https://en.wikipedia.org/w/index.php?title=Flex_(lexical_analyser_generator)&action=edit&section=1" \o "Edit section: History)]

|  |  |
| --- | --- |
| https://upload.wikimedia.org/wikipedia/commons/thumb/b/b4/Ambox_important.svg/20px-Ambox_important.svg.png | This section **is incomplete**.*(November 2012)* |

Flex was written in [C](https://en.wikipedia.org/wiki/C_(programming_language)) by [Vern Paxson](https://en.wikipedia.org/wiki/Vern_Paxson) around 1987.[[1]](https://en.wikipedia.org/wiki/Flex_(lexical_analyser_generator)#cite_note-releasehistorypage9-1) He was translating a [Ratfor](https://en.wikipedia.org/wiki/Ratfor" \o "Ratfor) generator, which had been led by [Jef Poskanzer](https://en.wikipedia.org/wiki/Jef_Poskanzer" \o "Jef Poskanzer).[[10]](https://en.wikipedia.org/wiki/Flex_(lexical_analyser_generator)#cite_note-10)

## Example lexical analyzer[[edit](https://en.wikipedia.org/w/index.php?title=Flex_(lexical_analyser_generator)&action=edit&section=2" \o "Edit section: Example lexical analyzer)]

This is an example of a Flex scanner for the instructional programming language [PL/0](https://en.wikipedia.org/wiki/PL/0).

The tokens recognized are: '+', '-', '\*', '/', '=', '(', ')', ',', ';', '.', ':=', '<', '<=', '<>', '>', '>='; numbers: 0-9 {0-9}; identifiers: a-zA-Z {a-zA-Z0-9} and keywords: begin, call, const, do, end, if, odd, procedure, then, var, while.

%{

#include *"y.tab.h"*

%}

digit [0-9]

letter [a-zA-Z]

%%

"+" { **return** PLUS; }

"-" { **return** MINUS; }

"\*" { **return** TIMES; }

"/" { **return** SLASH; }

"(" { **return** LPAREN; }

")" { **return** RPAREN; }

";" { **return** SEMICOLON; }

"," { **return** COMMA; }

"." { **return** PERIOD; }

":=" { **return** BECOMES; }

"=" { **return** EQL; }

"<>" { **return** NEQ; }

"<" { **return** LSS; }

">" { **return** GTR; }

"<=" { **return** LEQ; }

">=" { **return** GEQ; }

"begin" { **return** BEGINSYM; }

"call" { **return** CALLSYM; }

"const" { **return** CONSTSYM; }

"do" { **return** DOSYM; }

"end" { **return** ENDSYM; }

"if" { **return** IFSYM; }

"odd" { **return** ODDSYM; }

"procedure" { **return** PROCSYM; }

"then" { **return** THENSYM; }

"var" { **return** VARSYM; }

"while" { **return** WHILESYM; }

{letter}({letter}|{digit})\* {

yylval.id = strdup(yytext);

**return** IDENT; }

{digit}+ { yylval.num = atoi(yytext);

**return** NUMBER; }

[ \t\n\r] */\* skip whitespace \*/*

. { printf("Unknown character [%c]**\n**",yytext[0]);

**return** UNKNOWN; }

%%

int yywrap(void){**return** 1;}

## Internals[[edit](https://en.wikipedia.org/w/index.php?title=Flex_(lexical_analyser_generator)&action=edit&section=3" \o "Edit section: Internals)]

*Main article:*[*Lexical analysis*](https://en.wikipedia.org/wiki/Lexical_analysis)

These programs perform character parsing and tokenizing via the use of a [deterministic finite automaton](https://en.wikipedia.org/wiki/Deterministic_finite_automaton) (DFA). A DFA is a theoretical machine accepting [regular languages](https://en.wikipedia.org/wiki/Regular_language). These machines are a subset of the collection of [Turing machines](https://en.wikipedia.org/wiki/Turing_machine). DFAs are equivalent to [read-only right moving Turing machines](https://en.wikipedia.org/wiki/Read-only_right_moving_Turing_machines). The syntax is based on the use of [regular expressions](https://en.wikipedia.org/wiki/Regular_expressions). See also [nondeterministic finite automaton](https://en.wikipedia.org/wiki/Nondeterministic_finite_automaton).

## Issues[[edit](https://en.wikipedia.org/w/index.php?title=Flex_(lexical_analyser_generator)&action=edit&section=4" \o "Edit section: Issues)]

### Time complexity[[edit](https://en.wikipedia.org/w/index.php?title=Flex_(lexical_analyser_generator)&action=edit&section=5" \o "Edit section: Time complexity)]

A Flex lexical analyzer usually has time complexity {\displaystyle O(n)} in the length of the input. That is, it performs a constant number of operations for each input symbol. This constant is quite low: [GCC](https://en.wikipedia.org/wiki/GNU_Compiler_Collection) generates 12 instructions for the DFA match loop.[*[citation needed](https://en.wikipedia.org/wiki/Wikipedia:Citation_needed" \o "Wikipedia:Citation needed)*] Note that the constant is independent of the length of the token, the length of the regular expression and the size of the DFA.

However, using the REJECT macro in a scanner with the potential to match extremely long tokens can cause Flex to generate a scanner with non-linear performance. This feature is optional. In this case, the programmer has explicitly told Flex to "go back and try again" after it has already matched some input. This will cause the DFA to backtrack to find other accept states. The REJECT feature is not enabled by default, and because of its performance implications its use is discouraged in the Flex manual.[[11]](https://en.wikipedia.org/wiki/Flex_(lexical_analyser_generator)#cite_note-performance-11)

### Reentrancy[[edit](https://en.wikipedia.org/w/index.php?title=Flex_(lexical_analyser_generator)&action=edit&section=6" \o "Edit section: Reentrancy)]

By default the scanner generated by Flex is not [reentrant](https://en.wikipedia.org/wiki/Reentrant_(subroutine)). This can cause serious problems for programs that use the generated scanner from different threads. To overcome this issue there are options that Flex provides in order to achieve reentrancy. A detailed description of these options can be found in the Flex manual.[[12]](https://en.wikipedia.org/wiki/Flex_(lexical_analyser_generator)#cite_note-12)

### Usage under non-Unix environments[[edit](https://en.wikipedia.org/w/index.php?title=Flex_(lexical_analyser_generator)&action=edit&section=7" \o "Edit section: Usage under non-Unix environments)]

Normally the generated scanner contains references to *unistd.h* header file which is [Unix](https://en.wikipedia.org/wiki/Unix-like) specific. To avoid generating code that includes *unistd.h*, *%option nounistd* should be used. Another issue is the call to *[isatty](https://en.wikipedia.org/wiki/Not_a_typewriter" \o "Not a typewriter)* (a Unix library function), which can be found in the generated code. The *%option never-interactive* forces flex to generate code that doesn't use *isatty*.[[13]](https://en.wikipedia.org/wiki/Flex_(lexical_analyser_generator)#cite_note-13)

### Using flex from other languages[[edit](https://en.wikipedia.org/w/index.php?title=Flex_(lexical_analyser_generator)&action=edit&section=8" \o "Edit section: Using flex from other languages)]

Flex can only generate code for [C](https://en.wikipedia.org/wiki/C_(programming_language)) and [C++](https://en.wikipedia.org/wiki/C%2B%2B). To use the scanner code generated by flex from other languages a [language binding](https://en.wikipedia.org/wiki/Language_binding) tool such as [SWIG](https://en.wikipedia.org/wiki/SWIG) can be used.

## Flex++[[edit](https://en.wikipedia.org/w/index.php?title=Flex_(lexical_analyser_generator)&action=edit&section=9" \o "Edit section: Flex++)]

|  |  |
| --- | --- |
| https://upload.wikimedia.org/wikipedia/en/thumb/f/f2/Edit-clear.svg/40px-Edit-clear.svg.png | This section may **require**[**cleanup**](https://en.wikipedia.org/wiki/Wikipedia:Cleanup) to meet Wikipedia's [quality standards](https://en.wikipedia.org/wiki/Wikipedia:Manual_of_Style). No [cleanup reason](https://en.wikipedia.org/wiki/Template:Cleanup/doc) has been specified. Please help [improve this section](https://en.wikipedia.org/w/index.php?title=Flex_(lexical_analyser_generator)&action=edit) if you can. *(November 2010) (*[*Learn how and when to remove this template message*](https://en.wikipedia.org/wiki/Help:Maintenance_template_removal)*)* |

**flex++** is a similar lexical scanner for [C++](https://en.wikipedia.org/wiki/C%2B%2B) which is included as part of the flex package. The generated code does not depend on any [runtime](https://en.wikipedia.org/wiki/Runtime_library) or external [library](https://en.wikipedia.org/wiki/Library_(computing)) except for a memory allocator ([malloc](https://en.wikipedia.org/wiki/Malloc" \o "Malloc) or a user-supplied alternative) unless the input also depends on it. This can be useful in [embedded](https://en.wikipedia.org/wiki/Embedded_system) and similar situations where traditional [operating system](https://en.wikipedia.org/wiki/Operating_system) or [C runtime](https://en.wikipedia.org/wiki/C_standard_library) facilities may not be available.

The flex++ generated C++ scanner includes the header file FlexLexer.h, which defines the interfaces of the two C++ generated classes.

## See also[[edit](https://en.wikipedia.org/w/index.php?title=Flex_(lexical_analyser_generator)&action=edit&section=10" \o "Edit section: See also)]

* ***https://upload.wikimedia.org/wikipedia/commons/thumb/3/31/Free_and_open-source_software_logo_%282009%29.svg/28px-Free_and_open-source_software_logo_%282009%29.svg.png***[***Free software portal***](https://en.wikipedia.org/wiki/Portal:Free_software)
* [Comparison of parser generators](https://en.wikipedia.org/wiki/Comparison_of_parser_generators)
* [Lex](https://en.wikipedia.org/wiki/Lex_programming_tool)
* [yacc](https://en.wikipedia.org/wiki/Yacc)
* [GNU Bison](https://en.wikipedia.org/wiki/GNU_Bison)
* [Berkeley Yacc](https://en.wikipedia.org/wiki/Berkeley_Yacc)

## References[[edit](https://en.wikipedia.org/w/index.php?title=Flex_(lexical_analyser_generator)&action=edit&section=11" \o "Edit section: References)]

* 1. ^ [Jump up to:***a***](https://en.wikipedia.org/wiki/Flex_(lexical_analyser_generator)#cite_ref-releasehistorypage9_1-0) [***b***](https://en.wikipedia.org/wiki/Flex_(lexical_analyser_generator)#cite_ref-releasehistorypage9_1-1) [*Levine, John*](https://en.wikipedia.org/wiki/John_R._Levine) (August 2009). [*flex & bison*](https://books.google.com/books?id=3Sr1V5J9_qMC&printsec=frontcover&dq=flex+and+bison&hl=en&sa=X&ei=4lipUJ76Kuu80QGGi4HwCg&ved=0CD0Q6AEwAA#v=snippet&q=%22In%20about%201987%2C%20Vern%20Paxson%22&f=false). O'Reilly Media. p. 9. [*ISBN*](https://en.wikipedia.org/wiki/International_Standard_Book_Number) [*978-0-596-15597-1*](https://en.wikipedia.org/wiki/Special:BookSources/978-0-596-15597-1). In about 1987, Vern Paxson of the Lawrence Berkeley Lab took a version of lex written in ratfor (an extended Fortran popular at the time) and translated it into C, calling it flex, for *'*Fast Lexical Analyzer Generator.*'*
  2. [**Jump up^**](https://en.wikipedia.org/wiki/Flex_(lexical_analyser_generator)#cite_ref-2) [*Levine, John R.*](https://en.wikipedia.org/wiki/John_R._Levine); Mason, Tony; Brown, Doug (1992). *[lex & yacc](https://books.google.com/books?id=YrzpxNYegEkC&printsec=frontcover" \l "v=onepage&q=%22a%20freely%20available%20version%20of%20lex%20is%20flex%22&f=false)* (2nd ed.). [*O'Reilly*](https://en.wikipedia.org/wiki/O%27Reilly_Media). p. 279. [*ISBN*](https://en.wikipedia.org/wiki/International_Standard_Book_Number) [*1-56592-000-7*](https://en.wikipedia.org/wiki/Special:BookSources/1-56592-000-7). A freely available version of lex is flex.
  3. [**Jump up^**](https://en.wikipedia.org/wiki/Flex_(lexical_analyser_generator)#cite_ref-3) [*Levine, John R.*](https://en.wikipedia.org/wiki/John_R._Levine); Mason, Tony; Brown, Doug (1992). *[lex & yacc](https://books.google.com/books?id=YrzpxNYegEkC&printsec=frontcover" \l "PPA1,M1)* (2nd ed.). [*O'Reilly*](https://en.wikipedia.org/wiki/O%27Reilly_Media). pp. 1–2. [*ISBN*](https://en.wikipedia.org/wiki/International_Standard_Book_Number) [*1-56592-000-7*](https://en.wikipedia.org/wiki/Special:BookSources/1-56592-000-7).
  4. [**Jump up^**](https://en.wikipedia.org/wiki/Flex_(lexical_analyser_generator)#cite_ref-4) [*Levine, John*](https://en.wikipedia.org/wiki/John_Levine) (August 2009). [*flex & bison*](http://oreilly.com/catalog/9780596155988). O'Reilly Media. p. 304. [*ISBN*](https://en.wikipedia.org/wiki/International_Standard_Book_Number) [*978-0-596-15597-1*](https://en.wikipedia.org/wiki/Special:BookSources/978-0-596-15597-1).
  5. [**Jump up^**](https://en.wikipedia.org/wiki/Flex_(lexical_analyser_generator)#cite_ref-5) *[OpenBSD](https://en.wikipedia.org/wiki/OpenBSD" \o "OpenBSD)* (2015-12-11). [*"src/usr.bin/lex/"*](http://bxr.su/o/usr.bin/lex/). BSD Cross Reference*. Retrieved 2015-12-26*. This is flex, the fast lexical analyzer generator.
  6. [**Jump up^**](https://en.wikipedia.org/wiki/Flex_(lexical_analyser_generator)#cite_ref-6) [*"flex(1)"*](http://mdoc.su/-/flex.1). \*BSD [*man pages*](https://en.wikipedia.org/wiki/Man_page).
  7. [**Jump up^**](https://en.wikipedia.org/wiki/Flex_(lexical_analyser_generator)#cite_ref-7) [*"yacc(1)"*](http://mdoc.su/-/yacc.1). \*BSD [*man pages*](https://en.wikipedia.org/wiki/Man_page).
  8. [**Jump up^**](https://en.wikipedia.org/wiki/Flex_(lexical_analyser_generator)#cite_ref-8) [*"bison-3.0.4 – GNU parser generator"*](http://ports.su/devel/bison). *[OpenBSD ports](https://en.wikipedia.org/wiki/OpenBSD_ports" \o "OpenBSD ports)*. 2015-11-15*. Retrieved 2015-12-26*.
  9. [**Jump up^**](https://en.wikipedia.org/wiki/Flex_(lexical_analyser_generator)#cite_ref-9) [Is flex GNU or not?](http://flex.sourceforge.net/manual/Is-flex-GNU-or-not_003f.html#Is-flex-GNU-or-not_003f) [Archived](https://web.archive.org/web/20160303224333/http:/flex.sourceforge.net/manual/Is-flex-GNU-or-not_003f.html) 2016-03-03 at the [Wayback Machine](https://en.wikipedia.org/wiki/Wayback_Machine" \o "Wayback Machine)., flex FAQ
  10. [**Jump up^**](https://en.wikipedia.org/wiki/Flex_(lexical_analyser_generator)#cite_ref-10) [When was flex born?](http://flex.sourceforge.net/manual/When-was-flex-born_003f.html#When-was-flex-born_003f), flex FAQ
  11. [**Jump up^**](https://en.wikipedia.org/wiki/Flex_(lexical_analyser_generator)#cite_ref-performance_11-0) [*"Performance - Lexical Analysis With Flex, for Flex 2.5.37"*](http://flex.sourceforge.net/manual/Performance.html). Flex.sourceforge.net*. Retrieved 2013-02-25*.
  12. [**Jump up^**](https://en.wikipedia.org/wiki/Flex_(lexical_analyser_generator)#cite_ref-12) [*"Reentrant - Lexical Analysis With Flex, for Flex 2.5.37"*](http://flex.sourceforge.net/manual/Reentrant.html). Flex.sourceforge.net*. Retrieved 2013-02-25*.
  13. [**Jump up^**](https://en.wikipedia.org/wiki/Flex_(lexical_analyser_generator)#cite_ref-13) [*"Code-Level And API Options - Lexical Analysis With Flex, for Flex 2.5.37"*](http://flex.sourceforge.net/manual/Code_002dLevel-And-API-Options.html). Flex.sourceforge.net*. Retrieved 2013-02-25*.

## Further reading[[edit](https://en.wikipedia.org/w/index.php?title=Flex_(lexical_analyser_generator)&action=edit&section=12" \o "Edit section: Further reading)]

* [*Levine, John*](https://en.wikipedia.org/wiki/John_R._Levine) (August 2009). [*flex & bison*](http://oreilly.com/catalog/9780596155988). O'Reilly Media. [*ISBN*](https://en.wikipedia.org/wiki/International_Standard_Book_Number) [*978-0-596-15597-1*](https://en.wikipedia.org/wiki/Special:BookSources/978-0-596-15597-1).
* M. E. Lesk and E. Schmidt, *LEX - Lexical Analyzer Generator*
* Alfred Aho, Ravi Sethi and Jeffrey Ullman, *Compilers: Principles, Techniques and Tools*, Addison-Wesley (1986). Describes the pattern-matching techniques used by flex (deterministic finite automata)

## External links[[edit](https://en.wikipedia.org/w/index.php?title=Flex_(lexical_analyser_generator)&action=edit&section=13)]

* [Official website](https://github.com/westes/flex)
* [ANSI-C Lex Specification](http://www.quut.com/c/ANSI-C-grammar-l-1998.html)
* [JFlex: Fast Scanner Generator for Java](http://www.jflex.de/)
* [Brief description of Lex, Flex, YACC, and Bison](http://dinosaur.compilertools.net/)