

Compiler Construction

~ Flex ~

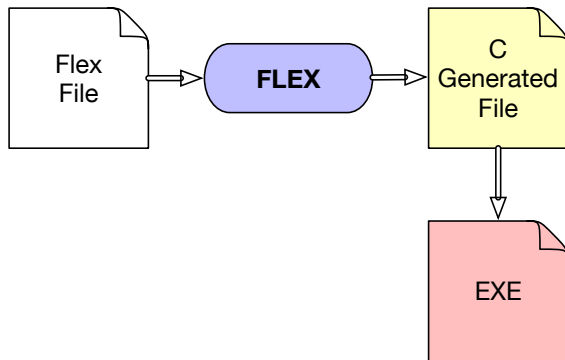
Flex

- **Flex: Fast Lexical Analyzer** generator
- **Initial release:** 1987
- Written in C by Vern Paxson
- Generates lexical analyzers
- GNU version of Lex (written by Mike Lesk and Eric Schmidt in 1975 – BellLabs)

Disclaimer

TC now uses [RE/Flex](#) as its lexer generator. This enables more features than Flex and generates higher quality code, but is nonetheless very similar (especially since it is mostly compatible with Flex files).

Overview



Typical Flex file

```
%{  
  [pre-code C (nec. def.)]  
%}  
  
[definitions and options]  
  
%%  
  
[rules]  
  
%%  
  
[post-code C (subprograms)]
```

Flex file structure

- C declarations, prologue and custom code are copied to the lexer *verbatim* and can be used for auxiliary functions, global variables...
- Definitions can be used specify regex shorthands.
- Rules have the form `pattern { action }` where `pattern` is a regex and `action` is C/C++ code.

First example

```
%{  
%}  
/* Only one input file */  
%option noyywrap  
num [0-9]+  
%%  
{num} { printf("NUMBER [%s]\n",  
              yytext); }  
" "    {}  
.      { printf("UNKNOWN [%s]\n",  
              yytext); }  
%%  
int main(void) {  
    yylex();  
    return 0;  
}
```

Try it:

```
$ ls  
tmp.lex  
$ flex tmp.lex  
$ gcc lex.yy.c  
$ echo "1 ==1" | ./a.out  
NUMBER [1]  
UNKNOWN [=]  
UNKNOWN [=]  
NUMBER [1]
```

Flex – details

- **yytext** the recognized text
- **yytext** the size of the recognized text
- **yytext** starts the scanning
- **yywrap** called when the end of the text to analyze is encountered. Can be refined if needed.
- For each of matched regexps one can return an identifier (a token)

Flex – details

- **yytext** the recognized text
- **yytext** the size of the recognized text
- **yytext** starts the scanning
- **yywrap** called when the end of the text to analyze is encountered. Can be refined if needed.
- For each of matched regexps one can return an identifier (a token)

*Bison (the parser) will analyze
this stream of tokens...*

Flex example – wc linux command

```
%{
#include <stdio.h>
static int chars_ = 0, lines_ = 0, words_ = 0;
%}

%%
\n          { ++chars_; ++lines_; }
[^ \t\n]+   { chars_ += yyleng; ++words_; }
.           { ++chars_; }

%%
int yywrap () {
    printf ("%7d %7d %7d\n", lines_, words_, chars_);
    return 1;
}
int main(){ yylex(); return 1; }
```

Remarks

Rules order

Always start by the more specific rule!

Reentrancy

Problems may occur when using simultaneously multiple instances of the lexer.

Summary




yylex



yytext



yyleng



yywrap