

# Flex

Kulikov Alexey

ABBYY

Moscow, 2017

# What is Flex?

- Flex – The Fast Lexical Analyzer;
- A tool for generating programs that perform pattern-matching on text;
- Flex is a free implementation of the original Unix **lex** program;
- Free, but non-GNU.
- [github.com/westes/flex](https://github.com/westes/flex)

- Lex – A Lexical Analyzer Generator;
- A tool for writing programs that are directed by regex instances from the input;
- Is designed to simplify interfacing with Yacc;
- The «host language» can be **C**, Ratfor, ...
- [dinosaur.compilertools.net](http://dinosaur.compilertools.net)

- Lex source – a table ( regex  $\rightarrow$  action );
- Lex generates a DFA based on the table;
- DFA parses the input line by line and splits lines into tokens;
- If a token matches some **regex**, the corresponding **action** occurs.

- Lex accepts ambiguous specifications and chooses **the longest match** possible;
- All the actions occur when a line is fully parsed;
- The order of actions is preserved;
- If no match found, the token is printed «as is».

- 1 Source  $\rightarrow$  Lex  $\rightarrow$  yylex
- 2 Input  $\rightarrow$  yylex  $\rightarrow$  Output

```
{definitions}  
%%  
{rules}  
%% // (optional)  
{user subroutines} // (optional)
```

# A trivial example

- Strips whitespace at the EOL;
- Squashes all other whitespace characters.

```
%0%  
[ \t]+$      ;  
[ \t]+$      printf("  ");
```



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

- $xyz'' + +'' \Leftrightarrow xyz \backslash + \backslash +$
- $[a - z0 - 9 < > \_]$
- $[\backslash 40 - \backslash 176]$  – printable ASCII
- $(ab|cd+)?(ef)^*$
- $ab/cd$  – trailing context
- $^abc\$ \Leftrightarrow ^abc/\backslash n$
- $< x > a$  – initial state of DFA is x

- Default = input  $\rightarrow$  output;
- yytext = matched token, yyleng = len(yytext);
- ECHO = printf("%s", yytext);
- input(), unput(), output() macros;
- yymore(), yylest(yyleng - 1);
- yywrap() after EOF, default retval == 1;
- REJECT; means «go to the next alternative»;
- #define YY\_USER\_ACTION.

Copied directly to `lex.yy.c`:

- Any line that is not a regex-action pair and begins with a whitespace;
- Anything between `{%` and `%}`;
- Anything after `%%`.

YACC uses retvals from `yylex()`.

So, each user action block should end with

```
return ( token );
```

Link it with `-lfl` key.

```
%START AA BB CC
```

```
%%
```

```
^a {ECHO; BEGIN AA;}
```

```
^b {ECHO; BEGIN BB;}
```

```
^c {ECHO; BEGIN CC;}
```

```
\n {ECHO; BEGIN 0;}
```

```
<AA>magic printf(" first ");
```

```
<BB>magic printf(" second ");
```

```
<CC>magic printf(" third ");
```

```
%x quote
%%
```

...other rules for dealing with quotes...

```
<quote><<EOF>> {
    error( "unterminated quote" );
    yyterminate();
}
<<EOF>> {
    if ( *++filelist )
        yyin = fopen( *filelist , "r" );
    else
        yyterminate();
}
```

- %option c++, or 'flex -+', or 'flex++'
- 'lex.yy.c' → 'lex.yy.cc', includes 'FlexLexer.h'
- FlexLexer
  - const char\* YYText() ↔ yytext
  - int YYLeng() ↔ yyleng
  - int lineno() const ↔ %option yylineno
- yyFlexLexer : FlexLexer
  - yyFlexLexer( istream\* yyin, ostream\* yyout )
  - virtual int yylex()

- Remember that `.` does not include newline;
- Don't use `(.|\n)+`, it causes buffer overflow;
- REJECT;



Dura lex, sed lex.