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

What is lex?

- 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

Lex overview

- 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 overview

- 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».

Lex pipeline

1. Source → [Lex] → yylex
2. Input → [yylex] → Output

Lex source code

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

A trivial example

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

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

Regex

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

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

User actions

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

Definitions

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 conditions

```
%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");
```

EOF rules

```
%x quote
%%

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

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

C++ scanners

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

Notes

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

Dura lex, sed lex.