Principles of Programming Languages

COMP251: Lex (Flex) and Yacc (Bison)

Prof. Dekai Wu

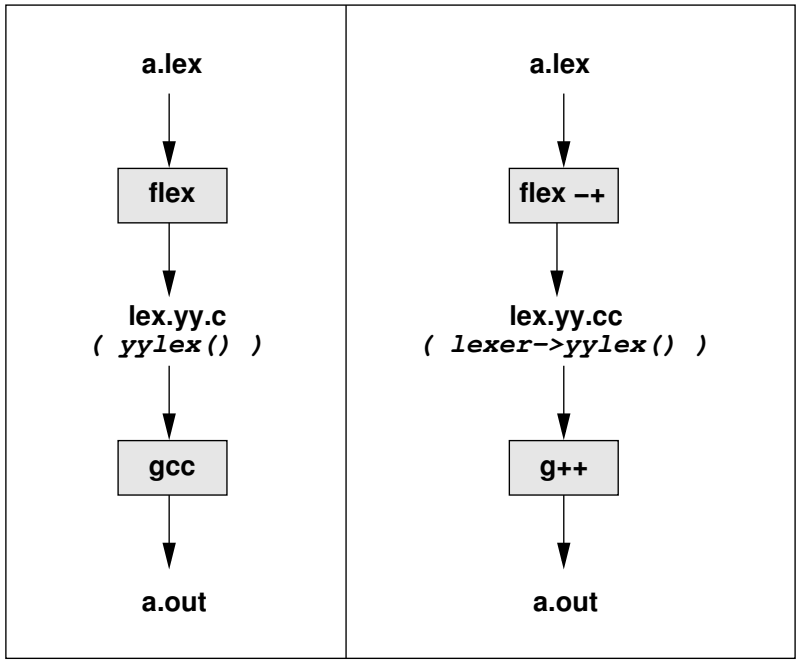
Department of Computer Science and Engineering
The Hong Kong University of Science and Technology
Hong Kong, China

Fall 2006

Part I

flex

flex: Fast Lexical Analyzer



- flex is GNU's extended version of the standard UNIX utility lex, that generates scanners or tokenizers or lexical analyzers.
- flex reads a description of a scanner written in a lex file and outputs a C or C++ program containing a routine called `yylex()` in C or (FlexLexer*)lexer→yylex() in C++.
- flex compiles `lex.yy.c` to `a.out` which will be the lexical analyzer.

flex Example 1

```
%option noyywrap /* see pp. 30 */

%{
int numlines = 0;
int numchars = 0;
%}

%%
\n   ++numlines; ++numchars;
.    ++numchars;
%%

int main(int argc, char** argv)
{
    yylex();
    printf("# of lines = %d, # of chars = %d\n", numlines, numchars);
    return 0;
}
```

flex Input File Format

```
%{
    text to be copied exactly to the output
%}
flex Definitions

%%
    Rules = patterns in RE + actions in C or C++
%%

user code (in C or C++)
```

- Patterns, written in REs, must start on the first column, and action must start on the same line as its pattern.
- In the Definitions or Rules sections, any indented text or text enclosed in "%{" and "%}" is copied verbatim to the output.

How the Input is Matched?

- The generated lexical analyzer should have a loop calling the function `yylex()` for the input file to be scanned.
- Each call to `yylex()` will scan the input from left to right looking for strings that match any of the RE patterns.
- If it finds more than 1 match, it takes the longest match.
- If it finds 2 matches of the same length, it takes the first rule.
- When there is a match,

```
extern char* yytext = /* content of matched string */
extern int yyleng = /* length of the matched string */
```

- If no rule is given, the default rule is to echo the input to the output.

flex Example 2: Default Rule

```
%option noyywrap

%%
%%

int main(int argc, char** argv)
{
   yylex();
   return 0;
}
```

How the Input is Matched? ..

- Actually the variable yytext can be specified as a pointer or an array in the flex-definition section.

```
%pointer    /* extern char* yytext */
%array      /* extern char yytext[YYLMAX] */
```

- Using pointer for yytext renders faster operation and avoids buffer overflow for large tokens. While it may be modified but you should NOT lengthen it or modify beyond its length (as given by yyleng). Using array for yytext allows you to modify the matched string freely.
- You cannot use %array with C++ programs.

flex Example 3: Use of yytext

```
%option noyywrap

%{
#include <stdio.h>
%}

%%
[a-yA-Y]     printf("%c", *yytext + 1);
[zZ]         printf("%c", *yytext - 25);
.            printf("%c", *yytext);
%%

int main(int argc, char** argv)
{
   yylex();
   return 0;
}
```

2 flex Directives: ECHO, REJECT

1. ECHO: copy `yytext` to the output
2. REJECT: ignore the current match and proceed to the next match.
   - if there are 2 rules that match the same length of input, it may be used to select the 2nd rule.
   - may be used to select the rule that matches less text.

flex Example 4: REJECT

```
%option noyywrap

%{
#include <stdio.h>
%}

%%
a      |
ab     |
abc    |
abcd   ECHO; REJECT;
.|\n   printf("xx%c", *yytext);
%%

int main(int argc, char** argv)
{
   yylex(); return 0;
}
```

Global Variables/Classes

C Implementation	C++ Implementation
FILE* yyin	abstract base class: FlexLexer
FILE* yyout	derived class: yyFlexLexer
char* yytext	member function: const char* YYText()
int yyleng	member function: int YYLeng()

Miscellaneous

Exceptions about character class REs:

- For character class: special symbols like *, + lose their special meanings and you don't have to escape them. However, you still have to escape the following symbols: \, -,], ∧, etc.
- There are some pre-defined special character class expressions enclosed inside "[:" and ":]", e.g.,

```
[:alnum:]  [:alpha:]  [:digit:]
[:lower:]  [:upper:]
```

Some important command-line options:

Option	Meaning
-d	debug mode
-p	performance report
-s	suppress default rule; can find holes in rules
-+	generate C++ scanners

flex Example 5: Generating C++ Scanners

```
%option noyywrap

%{
int mylineno = 0;
%}

string   \"[^\n"]+\"
ws       [ \t]+
alpha    [A-Za-z]
dig      [0-9]
name     ({alpha}|{dig}|\$)({alpha}|{dig}|[_.\-/$])*
num1     [-+]?{dig}+\.?([eE][-+]?{dig}+)?
num2     [-+]?{dig}*\.{dig}+([eE][-+]?{dig}+)?
number   {num1}|{num2}

%%
{ws}     /* skip blanks and tabs */
{number} cout << "number " << YYText() << '\n';
{name}   cout << "name " << YYText() << '\n';
{string} cout << "string " << YYText() << '\n';
\n       ++mylineno;
```

flex Example 5: Generating C++ Scanners ..

```
"/*"      {   int c;
              while ((c = yyinput()) != 0)
              {
                  if (c == '\n') {
                      ++mylineno;
                  } else if (c == '*') {
                      if ((c = yyinput()) == '/') {
                          break;
                      } else {
                          unput(c);
                      }
                  }
              }
          }
.         /* cout << "unrecognized " << YYText() << endl; */
%%

int main(int argc, char** argv)
{
    FlexLexer* lexer = new yyFlexLexer;
    while (lexer->yylex() != 0) {
    }
    return 0;
}
```