Lex Tutorial

Lex的工作

- Lex會把input當作 a sequence of characters
  - 一個以上連續的character會形成一個token
- Lex的目的是檢查token是否合法
  - 例如不合法的變數名稱(identifier)
- Lex必須事先定義規則
  - Regular expression
    - 可以被辨識的token

Lex的input

- 以Java為例
- public static void main() {

```
  int c;
  int a = 5;
  int 5a; //不合法的identifier
   c = add(a, 10);
   if (c > 10)
       print("c = " + -c);
  else
       print(c);
   print("Hello World");
}
```

Lex格式

- 分成三部分，每個部分以%%區隔開來

Definition

%%

Lex Rules

%%

User code

Definition

```
%{
#include<stdio.h>
unsigned charCount = 1, idCount = 0, lineCount = 1;
%}
operator [\+\-\*\/]
space [ \t]
eol \n

/* You should write your own regular expression. */
reserved_word
symbol
id
%%
```

Rules

- 定義token及對應的action

```
13  %%
14  {operator} {
15      printf("Line: %d, 1st char: %d, \"%s\" is an \"operator\".\n", lineCount, charCount, yytext);
16      charCount += yyleng;
17  }
18  {space} {
19      charCount++;
20  }
21  {eol}   {
22      lineCount++;
23      charCount = 1;
24  }
25
26  {reserved_word} {
27      /* You shold write your own code */
28  }
```

Rules

- Scanner所匹配規則的優先順序
  - Scanner會scan出長度最長的token去進行匹配
  - 如果匹配長度一樣，則看被定義的先後順序(由上到下)

Our code

```
37  int main(){
38      yylex();
39      return 0;
40   }
41
```

Test file

```
1  public class Test1 {
2      public static int add(int a, int b) {
3          return a + b;
4      }
5  }
6
```

Output

```
shchiang@ubuntu:~/Desktop/Lex_Yacc/Lex/MyLex$ ./demo < test1.java
Line: 1, 1st char: 1, "public" is a "reserved word".
Line: 1, 1st char: 8, "class" is a "reserved word".
Line: 1, 1st char: 14, "Test1" is an "ID".
Line: 1, 1st char: 20, "{" is a "symbol".
Line: 2, 1st char: 5, "public" is a "reserved word".
Line: 2, 1st char: 12, "static" is a "reserved word".
Line: 2, 1st char: 19, "int" is a "reserved word".
Line: 2, 1st char: 23, "add" is an "ID".
Line: 2, 1st char: 26, "(" is a "symbol".
Line: 2, 1st char: 27, "int" is a "reserved word".
Line: 2, 1st char: 31, "a" is an "ID".
Line: 2, 1st char: 32, "," is a "symbol".
Line: 2, 1st char: 34, "int" is a "reserved word".
Line: 2, 1st char: 38, "b" is an "ID".
Line: 2, 1st char: 39, ")" is a "symbol".
Line: 2, 1st char: 41, "{" is a "symbol".
Line: 3, 1st char: 9, "return" is a "reserved word".
Line: 3, 1st char: 16, "a" is an "ID".
Line: 3, 1st char: 18, "+" is an "operator".
Line: 3, 1st char: 20, "b" is an "ID".
Line: 3, 1st char: 21, ";" is a "symbol".
Line: 4, 1st char: 5, "}" is a "symbol".
Line: 5, 1st char: 1, "}" is a "symbol".
shchiang@ubuntu:~/Desktop/Lex_Yacc/Lex/MyLex$
```

Lex file中的特殊字元

- 這些字元在regular expression中有特殊意義，如果要當成一般字元，請在前面加上\這一個跳脫字元(Escape character)
  - ? * + | ( ) ^ $. [ ] { } " \
- Digit [0-9]
- Letter [a-zA-Z]
- Operator [\+\-*]

如何使用Lex file

- 我們的目的要將demo.l編譯成可以執行的scanner
- 首先必須安裝flex這個程式來編譯我們的lex file，以ubuntu為例
  - sudo apt-get install flex
- 透過flex將demo.l編譯成C source file，這個C source file就是我們的scanner
  - flex demo.l
- C source file預設檔名為lex.yy.c，最後我們可以利用gcc將其編譯成可執行檔
  - gcc lex.yy.c –o demo **-lfl**
- 執行檔為demo，假設我們要scan的檔案為test1.java
  - ./demo < test1.java
- 也可以直接執行demo，<Ctrl-D>可以送出EOF

對 Regular Expression不熟的同學

- 網路上對正則表達式的資源非常豐富

- Online regular expression tester
  - https://regexr.com/

作業繳交注意事項

- **due: 4/18 11:59 p.m.**
- 程式Demo環境是Ubuntu 18.04，因此請保證你們的程式碼能夠在Ubuntu上面編譯執行
- 請參考課程網頁中的測試檔案來驗證你的程式
- 助教會自行設計額外的測試檔案，因此請保證你所寫的Regular Expression可以匹配到大部分的case
  - 例如一些複雜的變數名稱、浮點數必須要可以是負數...
- 請準時繳交作業，作業遲交一天打七折
- 請把作業包成一個壓縮包，上傳至網大，檔名命為「學號_hw1」，學號輸錯，此項作業分數-10，沒輸學號分數-50，請同學注意
- 作業繳交之後，在繳交截止隔周會安排時間，到EC5023找助教Demo。