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

- 總共分成三個部分
- definition

  %%

  rules

  %%

  user code
- 每個部分以%%區隔開來

Lex範例

- 此範例在課程網頁中的Lex program example底下的demo.lex
- 目的是可以印出目前所找到的id，最後會印出總共有多少個id、character以及行數

Definition

```
%{
#include<stdio.h>
unsigned charCount = 0, idCount = 0,
lineCount = 0;
%}
id              [^ \t\n]+
space           [ ]
eol             \n
character       .
```

Rules

- 定義token及對應的action
- {id} { idCount++; charCount += **yyleng**;
  printf("word : %s, **yytext**);
  }

  {space} {//do nothing}

  {eol} { lineCount++; }

  {character} { charCount++;}
- 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
  - 假設出現perter123<space>，則scanner會選擇可以匹配perter123的token，而不會去選擇只可以匹配到peter的token
  - 如果匹配長度一樣，則看被定義的先後順序(由上到下)
  - 把space跟character的順序對調的話，則space沒辦法被匹配到

Our code

- 這部分完全是以C語言來撰寫
- //some globla variable, for example, symbol table

```
 main(){
yylex(); //yylex會呼叫scanner來scan程式碼
//yylex()結束表示接收到EOF(檔案結尾)
//接下來是user defined的code
//例如，你可以把一些資訊給印出來
printf("%d %d %d\n",charCount, idCount, lineCount);
return 0;
 }
```

demo.lex完整內容

```
%{
#include<stdio.h>
unsigned charCount = 0, idCount = 0, lineCount = 0;
%}
id          [^ \t\n]+
space       [ ]
eol         \n
character   .
%%
{word}      { idCount++; charCount += yyleng ; printf("This id is %s, yytext);}
{eol}       {lineCount++; }
{space}     {//do nothing}
{character} { charCount++;}
%%
 main(){  yylex();
            printf("%d %d %d\n",charCount, idCount, lineCount);
            return 0;
        }
```

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.lex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.lex編譯成C source file，這個C source file就是我們的scanner
  - flex demo.lex
- 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 **-lfl**
- 執行檔為a.out，假設我們要scan的檔案為test.in
  - ./a.out < test.in
- 也可以直接執行a.out，<Ctrl-D>可以送出EOF

作業繳交注意事項

- **due: April 14 (五), 23:59**
- 程式Demo環境是Ubuntu 14.04，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...
- 請把作業Email給我，jacky83528@gmail.com
- 請準時繳交作業，作業遲交一天打七折
- 作業繳交之後，繳交截止時間過後，就可以來EC5023找助教Demo，