第十三章 flex 使用

- flex 是用 C 程式語言撰寫的一支程式，用來產生一個語彙分析的程式，它的工作方式類似 UNIX 系統的公用程式 lex，它們都從一個含有語彙分析規格（specification）的本文檔案輸入本文，然後產生一個相對
- 應的語彙分析原始程式，所產生的是一個語彙分析原始程式，必須編譯過後才成為語彙分析器（lexical analyzer）。

13.1 flex 如何工作

- flex 所產生的語彙分析原始程式，內定名稱為 lex.yy.c，其中包含一個取得下一個符記（token）的函式，該函式內定簽名（signature）如下：
- ***int yylex(void);***

flex 的輸入本文檔稱為規格檔

- flex 的輸入本文檔稱為規格檔，習慣上副檔名都為「.l」。規格檔的格式如下：
- ***定義段落***
- ***%%***
- ***規則段落***
- ***%%***
- ***程式碼段落***

規格檔 counts.l 中的「定義段落」

- 規格檔 counts.l 中的「定義段落」如下：
- ***DIGIT [0-9]***
- ***LETTER [A-Za-z]***
- ***%{***
- ***int ids=0, lines=0;***
- ***%}***

規格檔 counts.l 中的「規則段落」

- 規格檔 counts.l 中的「規則段落」如下：
- ***{LETTER}({LETTER}|{DIGIT})* {***
- ***ids++;***
- ***printf("\t%3d\t\t\"%s\"\n",yyleng,yytext);***
- ***}*** ***/*識別字*/***
- ***\n { lines++; }*** ***/*列結束*/***
- ***. ;*** ***/*其他字元*/***

一個規則包含一個樣式及一個動作

- 一個規則（rule）包含一個樣式（pattern）及一個動作（action）。動作通常是在大括號內的 C 語言程式碼。
- 「{LETTER}({LETTER}|{DIGIT})*」為樣式，表示識別字，其相對應的動作將 ids 變數增一，然後將 flex 所提供的識別字長度 yyleng 以及識別字名稱 yytext 以指定的格式顯示在螢幕上。
- 「\n」為樣式，表示新列符號，其相對應的動作 { lines++; } 只是將列計數增一而已。
- 「.」為樣式，表示其他無法匹配（match）的字元，相對應的動作「;」，表示沒有動作。

規格檔counts.l中的「程式碼段落」

```
int yywrap(void)
{
  return 1;
}
int main(int argc, char *argv[])
{
  char cmd[80]="TYPE ";
  printf("    輸入規格檔\"%s\"內容如下：\n",argv[1]);
  system(strcat(cmd, argv[1]));
  printf("\t識別字長度\t識別字名稱\n");
  yyin = fopen(argv[1], "r");
  yylex();
  printf("\n\t總共列數=%d\t識別字數=%d\n",lines,ids);
  return 0;
}
```

規格檔counts.l中的「程式碼段落」

- 函式 main() 是主程式。
- 首先使用作業系統的 TYPE 命令將 argv[1] 輸入檔名的檔案內容顯示出來。yyin 是 flex 內定的輸入檔案指標，將它指向從命令列輸入的檔名。yylex() 函式負責取得下一個符記，每一個符記其相對應的動作直接將符記的相關資料顯示在螢幕，輸出如執行結果。

13.2 sym 符號表

- flex 所取得的符記不會這麼單純的只顯示在螢幕而已，也不會這麼單純的只辨認識別字而已，我們就以一個桌上型計算機專案來說明它的使用較為實際。
- 依照慣例，每一個符記都給一個相對應的整數編號，處理起來較為容易。這些符記的整數編號定義於 sym.h 表頭檔裡。
- 【sym.h】
- /************** sym.h ***************/
- #define symSYMMAX 25
- #define symEOF 0
- #define symerror 1
- #define symIDENTIFIER 2
- #define symNUMBER 3
- //略

13.3 symname 符號名稱

- 表頭檔 symname.h 提供相對應的符號名稱。
-
- 【symname.h】
-

```
/****************** symname.h *****************/
#include <stdlib.h>
#include "sym.h"
char names[symSYMMAX][32];
void symnameinit()
{
  strcpy(names[symEOF],"EOF");
  strcpy(names[symerror],"error");
  strcpy(names[symIDENTIFIER],"IDENTIFIER");
  strcpy(names[symNUMBER],"NUMBER");
  //略
}
```

規格檔 retint.l

- 規格檔 retint.l 傳回每一個符記的整數編號，它定義於 sym.h 表頭檔裡，其相對應的符記編號名稱定義於 symname.h 表頭檔，這兩個表頭檔在一開始時就透過 C 語言的 #include 指令引入規格檔裡，然後拷貝至 flex 的輸出，即語彙分析程式 lex.yy.c 裡頭。

13.4 符記結構

- flex 所產生的 lex.yy.c 原始程式，其中的 yylex() 函式所取得的下一個符記，常用的為 C 語言的 struct 結構，其程式碼 symbol.h 如下。
- 結構 struct symbolTag 包含下列三個欄位。
- 第一個欄位 sym 符記，定義於表頭檔sym.h者。
- 第二個欄位 len 符記字串長度。
- 第三個欄位 value 符記名稱。
- 函式 newSymbol() 透過傳進來的參數建立一個符記結構，並將其結構指標「struct symbolTag *」傳回。
- 函式 symbolToString() 將指定的符記結構指標所指結構的三個欄位值以字串型態傳回。

規格檔 retsym.l

- 規格檔 retsym.l 傳回每一個符記的結構指標，定義於 symbol.h 表頭檔裡，sym.h 及 symbol.h 這兩個表頭檔在一開始時就引入規格檔裡，然後拷貝至 flex 的輸出，即語彙分析程式 lex.yy.c 裡頭。
- %{
- /********** retsym.l **********/
- #include <stdlib.h>
- #include "sym.h"
- #include "symbol.h"
- #define YY_DECL struct symbolTag *yylex(void)
- %}
- %%
- [a-z] return newSymbol(symIDENTIFIER, yytext);
- [0-9]+ return newSymbol(symNUMBER, yytext);
- "-" return newSymbol(symMINUS, "-");
- //略

13.5 flex 使用說明

- flex 的輸入本文規格檔是一個合乎 flex 規格的檔案，其內容分三個段落（section），以兩個百分比符號隔開，它的結構如下：
- 定義段落
- %%
- 規則段落
- %%
- 程式碼段落
- 兩個百分比符號的指引隔開三個段落，百分比符號必須出現在一列的開頭，其後最好保持空白。

13.6 定義段落

- 「定義段落」屬於規格檔的第一個段落，包含一些單純的名稱宣告，用於簡化掃描程式的規格，名稱宣告格式如下：
- ***名稱　定義***
- 「名稱」第一個字元為英文字母或底線，其後根著零個或多個英文字母（A-Za-z）、阿拉伯數字（0-9）、底線（_）、連字符號（-）。
- 「定義」起於「名稱」後非空白的開始字元，一直到該列結束為止。「定義」可以參考前面定義過的名稱，以大括號括起來，也就是 {名稱}，會將它擴展成「(定義)」，事實上「名稱」就是一個巨集（macro）。如下例：
- **DIGIT [0-9]**
- **ID　[a-z][a-z0-9]***
- 「名稱」為 DIGIT，「定義」為一個阿拉伯數字。「名稱」為 ID，「定義」為一個識別字，第一個字元為小寫英文字母，其後為小寫英文字母或阿拉伯數字。

- ***{DIGIT}+"."{DIGIT}+***
- 表示小數點前後最少要有一個阿拉伯數字。「+」表示最少出現一次。例如 1.2、12.34、4567.89 等等都合乎規定。它相當於如下的名稱定義：
- ***([0-9])+"."([0-9])+***
- 若改成如下的定義：
- ***{DIGIT}+"."{DIGIT}****
- 表示小數點前最少要有一個阿拉伯數字，但小數點後可以沒有阿拉伯數字。例如 123.45、678.、90123.4567 等都合乎規定。它相當於如下的名稱定義：
- ***([0-9])+"."([0-9])****

13.7 規則段落

- flex 規格檔裡的「規則段落」包含一系列如下列格式的規格：
- ***樣式　動作***
- 「***樣式***」必須從一列的第一個位置開始，「***動作***」必須在同一列。「***樣式***」使用正規運算式（regular expression）表示。
- 正規運算式應該不能包含任何空白（white space）字元，因為空白字元是正規運算式結束的符號。
- flex 所處理的是 ASCII 碼 0-127（含）的字元。

- 下列的字元稱為中繼字元（metacharacter）， 對於 flex 的正規運算式有特殊的意義。
- **？ * + | ( ) ^ $. [ ] { } " **
- 連續的兩個正規運算式 ef 表示 e 與 f 的連結。
- 以垂直棒隔開的兩個正規運算式 e | f 表示 e 或 f。
- 下列的逸出順序（escape sequence）的意義如下：
-
- \b　反斜線（backspace）
- \n　新列（newline）
- \t　定位（tab）
- \f　跳頁（formfeed）
- \r　返回（carriage return）
- \ddd　表示八進位數 ddd
- \xdd　表示十六進位數 dd
- \c　表示c字元，c為a、b、f、n、r、t、v以外的字元。

- 加字記號（caret）「^」表示一列的開始。若出現在一列開始，此規則只匹配一列前端符記。
- 錢號「$」表示一列的結束。若出現在一列最後端，此規則只匹配列尾符記。
- 句點「.」匹配新列之外的任何字元，相當於 [^\n]。
- {name} 表示一個巨集，name 為巨集名稱。
- 星號「*」匹配前面正規運算式 0 次或多次。
- 加號「+」匹配前面正規運算式 1 次或多次。
- 問號「?」匹配前面正規運算式 0 次或 1 次。
- 小括號「(...)」將正規運算式分組。
- 方括號「[...]」表示字元類別（character class），並匹配其中任一字元。若在左方括號後跟隨著「^」符號，表示相反的意思。

表 13.1 主要的樣式匹配表

- 中繼字元 匹配
- -------- ----------
- . 除新列字元（₩n）外的所有字元
- ₩n 新列字元
- * 前項拷貝零次或多次
- + 前項拷貝一次或多次
- ? 前項拷貝零次或一次
- ^ 一列列頭
- $ 一列列尾
- a|b a或b
- (ab)+ (ab)拷貝一次或多次
- "a+b" 字串 "a+b"
- [ ] 字元類別（character class）

表 13.2 樣式匹配舉例

- **正規運算式　匹配**
- ---------- ------------------------
- abc　　abc
- abc*　　ab abc abcc abccc ...
- abc+　　abc abcc abccc ...
- a(bc)+　　abc abcbc abcbcbc ...
- a(bc)?　　a abc
- [abc]　　a,b,c **其中之一**
- [a-z]　　a-z **其中之一**
- [a₩-z]　　a,-,z **其中之一**
- [-az]　　-,a,z **其中之一**
- [A-Za-z0-9]+　A-Z,a-z,0-9 **其中最少一個**
- [₩t₩n]　　**空白字元**（white space）
- [^ab]　　a,b **以外任何字元**
- [a^b]　　a,^,b **其中之一**
- [a|b]　　a,|,b **其中之一**
- a|b　　a,b **其中之一**

13.8 程式碼段落

- flex 規格檔裡的「程式碼段落」所包含的 C 語言程式碼只是單純的將它拷貝至輸出檔 lex.yy.c 而已。

13.9 輸出語彙分析原始程式

- flex 內定輸出 lex.yy.c 語彙分析原始程式檔案，該檔案內含一個掃描函式 yylex()，其架構如下。
- ***int yylex() { /*略*/ }***
- 這個掃描函式內定的名稱為 yylex，沒有參數，傳回一個整數值。我們可以宣告一個巨集來改變它，格式如下：
- ***#define YY_DECL 傳回值型態 掃描函式名稱 (參數列)***
- 「傳回值型態」原來內定為整數 int，您可改為浮點數 float，或倍精確浮點數 double，或結構 struct，或指標等等，凡 C 語言所認得的型態都可以使用。
- 「掃描函式名稱」您也可以改變，不過大家都已經知道 yylex 是取得下一個符記的函式名稱，建立共識不易，建議還是不改為妙。
- 原來的 yylex() 函式並沒有提供參數，您也可以提供參數，依問題的需求而定。

- 我們在前面的 retsym.l 規格檔裡就改變「傳回值型態」為結構的指標，該巨集宣告如下。
-
- **#define YY_DECL struct symbolTag *yylex(void)**

13.10 flex 可用變數

- **表** 13.3 flex **可用變數表**
-
- **變數** 說明
- ---------- ----------
- char *yytext **符記名稱**
- int yyleng **符記名稱長度**
- FILE *yyin **輸入檔案指標**
- void yyrestart(FILE *newfile) **重新指到指定新檔**
- FILE *yyout **輸出檔案指標**
- YY_CURRENT_BUFFER **傳回** YY_BUFFER_STATE
- YY_START **傳回目前開始狀態**

13.11 flex 與 yacc 的介面

- yacc 所產生的符號表 y.tab.h，您只須將這個檔案引入至 flex 規格檔裡就可以了，如下例：
-
-
```
%{
        #include "y.tab.h"
%}
%%
```