Yacc

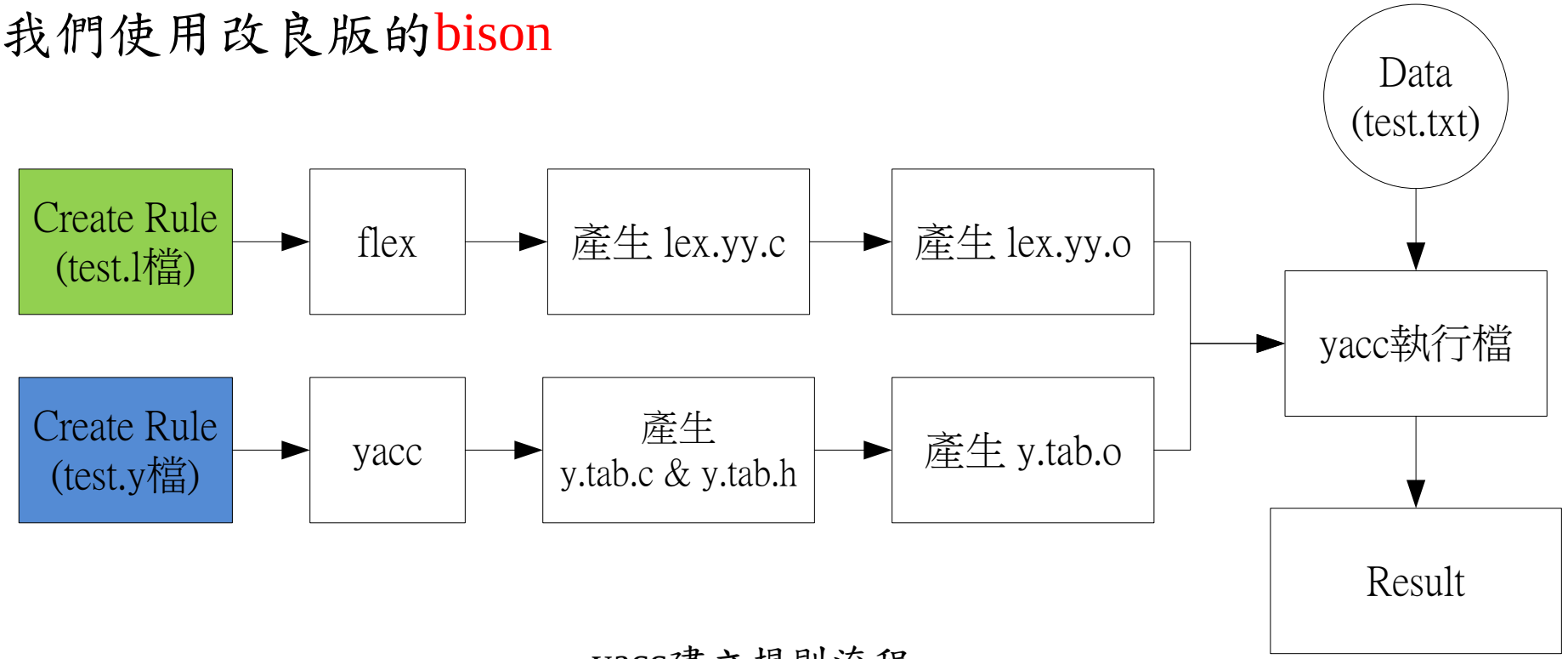
-Yet Another Compiler Compiler

Outline

- 什麼是Yacc
- 怎麼使用Yacc
- 怎麼編譯Yacc
- 作業要求
- 加分題
- 參考資料

What is yacc.

- yacc是linux系統中用來生成編譯器的編譯器
  - 我們使用改良版的bison



yacc建立規則流程

How to use yacc

- Parser.y :
  - 與Lex類似

```
definition    //宣告
%%
rules         //規則建立
%%
user code //主程式
```

How to use yacc

- Definition
  - 用來宣告token，也可以宣告變數、include
  - Ex.

```
1 %token LPAREN RPAREN PLUS REALNUMBER
2 %{
3     #include <stdio.h>
4     double global_value;
5 %}
```

How to use yacc

➢Rules

A : BODY;
名字：規則

- 名字可以是不以數字開頭的任意長度英文、底線”_”、點“.”的組合。
- Ex.

```
1 A   :   B C D ;
2 A   :   E F ;
3 A   :   G ;
4 // ================
5 A   :   B C D
6     |   E F
7     |   G
8     ;
```

How to use yacc

- Rules
  - 規則有對應的動作

```
   $$       $1       $2    $3
1 expr : DIGIT PLUS DIGIT
2     {
3         $$ = $1 + $3;
4         printf("Result: %d.\n", $$);
5     }
```

How to use yacc

- Lex.l
  - 需要Lex來分割token
  - 在definition中，需要include我們寫的parser
    - 檔名會根據你的.y檔變
    - Ex.

```
1 %{
2         #include <stdio.h>
3         #include <stdlib.h>
4         #include "y.tab.h"
5 %}
```

How to use yacc

- Lex.l
  - Rule中，要改用return的

```
1 [0-9]+  {   return(DIGIT); }
2 “(“     {   return(LPAREN); }
3 “)“     {   return(RPAREN); }
4 “+”     {   return(PLUS); }
```

How to use yacc

- Lex.l
  - UserCode
  - Ex.

```
1 int main()
2 {
3   yyparse();
4   return 0;
5 }
6
7 int yyerror(char *msg)
8 {
9     printf("Error:%s \n", msg);
10 }
```

How to use yacc

- Sample: 加法

```
1  //calc.y
2  %token DIGIT
3  %token PLUS
4
5  %left '+'
6  %start list
7
8  %%
9  list :
10       | list expr
11       | list expr '\n'
12       ;
13 expr : DIGIT '+' DIGIT
14     {
15         $$ = $1 + $3;
16         printf("Result: %d.\n", $$);
17     }
18     | expr '+' DIGIT
19     {
20         $$ = $1 + $3;
21         printf("Result: %d.\n", $$);
22     }
23 %%
```

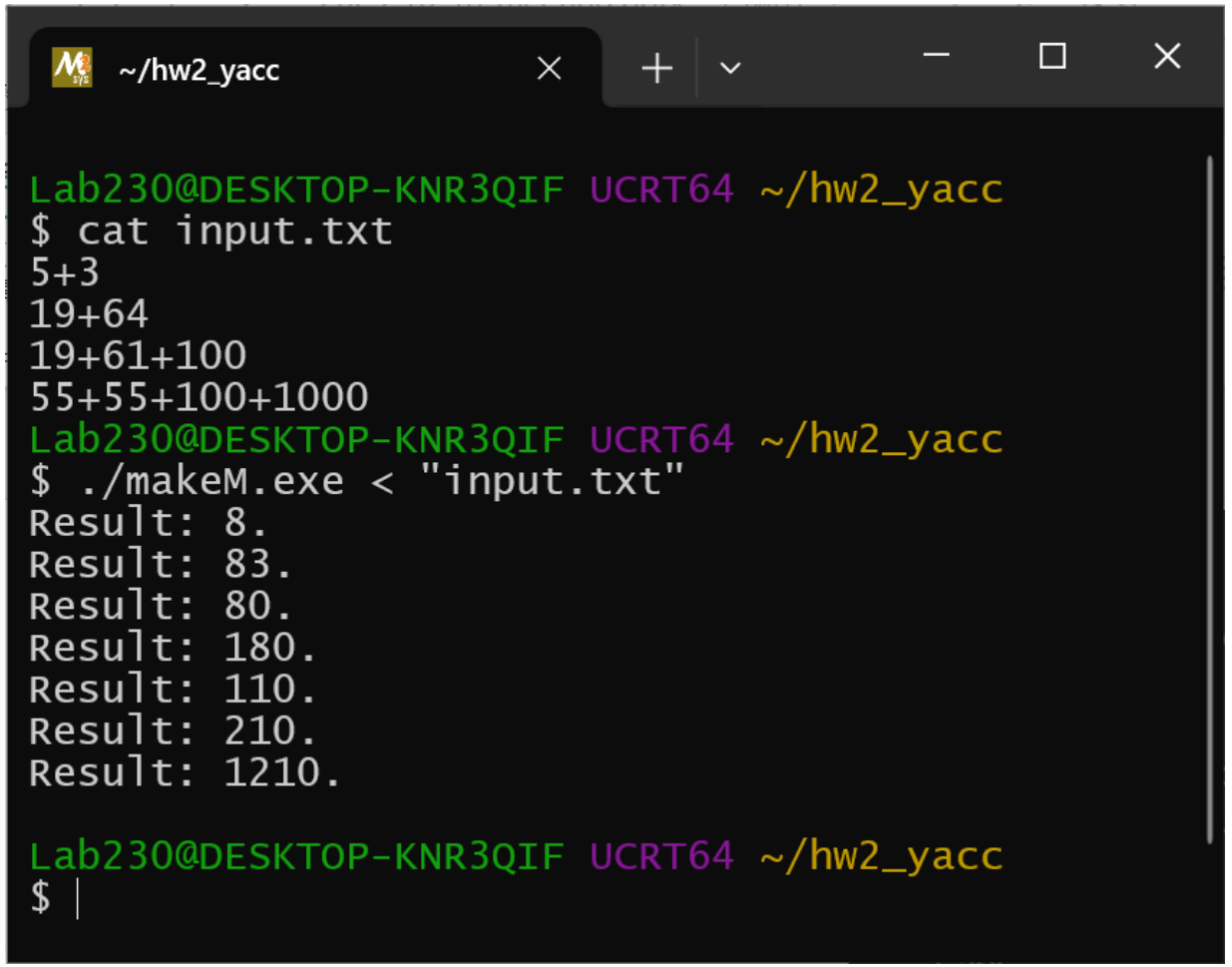
How to use yacc

- Sample: 加法

```
1 //calc.l
2 %option noyywrap
3
4 %{
5 #include <stdio.h>
6 #include "calc.tab.h"
7 %}
8
9 %%
10 [0-9]+ { yylval = atoi(yytext);
11         return(DIGIT);
12 }
13
14 "+" { return(yytext[0]);}
15 \n { return('\n');}
16
17 %%
18 int main()
19 {
20     yyparse();
21     return 0;
22 }
23
24 void yyerror (char const *s) {
25     fprintf (stderr, "%s\n", s);
26 }
```

How to use yacc

➢Sample: 加法



How to compile yacc

- Installation
  - 如果flex是按照ppt安裝，bison也會一起裝

```
1 $ gcc --version
2 $ flex --version
3 $ bison --version
```

  - 如果沒有
    1. 回去看flex的ppt.
    2. $ pacman –S bison

How to compile yacc

- Step 1.
  - 先編譯.l檔與.y檔

```
1 $ flex calc.l
2 $ bison -d calc.y
```

  - 結果

```
-rw-r--r-- 1 Lab230 None   311 十二月  4 21:10 calc.l
-rw-r--r-- 1 Lab230 None 39297 十二月  4 21:12 calc.tab.c
-rw-r--r-- 1 Lab230 None  2570 十二月  4 21:12 calc.tab.h
-rw-r--r-- 1 Lab230 None   316 十二月  4 20:30 calc.y
-rw-r--r-- 1 Lab230 None   509 十二月  4 21:11 CMakeLists.txt
-rw-r--r-- 1 Lab230 None    34 十二月  4 20:35 input.txt
-rw-r--r-- 1 Lab230 None 44506 十二月  4 21:11 lex.yy.c
```

How to compile yacc

- Step 2.
  - 將兩個.c檔編譯成obj檔

```
1 $ gcc -c calc.tab.c
2 $ gcc -c lex.yy.c
```

  - 結果

```
-rw-r--r-- 1 Lab230 None   311 十二月  4 21:10 calc.l
-rw-r--r-- 1 Lab230 None 39297 十二月  4 21:12 calc.tab.c
-rw-r--r-- 1 Lab230 None  2570 十二月  4 21:12 calc.tab.h
-rw-r--r-- 1 Lab230 None  4725 十二月  4 21:19 calc.tab.o
-rw-r--r-- 1 Lab230 None   316 十二月  4 20:30 calc.y
-rw-r--r-- 1 Lab230 None   509 十二月  4 21:11 CMakeLists.txt
-rw-r--r-- 1 Lab230 None    34 十二月  4 20:35 input.txt
-rw-r--r-- 1 Lab230 None 44506 十二月  4 21:11 lex.yy.c
-rw-r--r-- 1 Lab230 None 17560 十二月  4 21:19 lex.yy.o
```

How to compile yacc

- Step 3.
  - 將兩個.o檔編譯成可執行檔

```
1 $ gcc calc.tab.o lex.yy.o -o main
```

  - 完成

How to compile yacc

- Another way – CMake
  1. 安裝cmake
     - Pacman –S mingw-w64-ucrt-x86_64-cmake
     - macOS要自己查怎麼裝
     - 紅框改成自己的檔名
  2. $ cmake .
  3. $ cmake --build .

```
cmake_minimum_required(VERSION 3.27)
project(hw2_yacc)

find_package(BISON)
find_package(FLEX)

BISON_TARGET(MyParser calc.y ${CMAKE_CURRENT_BINARY_DIR}/parser.c
COMPILE_FLAGS -d)
FLEX_TARGET(MyScanner calc.l  ${CMAKE_CURRENT_BINARY_DIR}/lexer.c
DEFINES_FILE calc_lex.c)
ADD_FLEX_BISON_DEPENDENCY(MyScanner MyParser)

include_directories(${CMAKE_CURRENT_BINARY_DIR})
add_executable(main
    ${BISON_MyParser_OUTPUTS}
    ${FLEX_MyScanner_OUTPUTS}
)
target_link_libraries(main ${FLEX_LIBRARIES})
```

Homework

➢題目:

給定一個檔案有長度不等的四則運算式，每一個式子以換行隔開，給出過程及答案

- 計算必須遵守括號與先乘除後加減的原則
- 結果必須要印出每個計算式的計算過程與答案

Homework

- Example:
  - 測資

```
12+32+35+21-20
65+35
32-2
112-(2+32)*3
36/(30+3*2)
12+(12-(12+12))
89+77/(8-1)
```

```
12 plus 32 equals 44
44 plus 35 equals 79
79 plus 21 equals 100
100 minus 20 equals 80
ANS is 80

65 plus 35 equals 100
ANS is 100

32 minus 2 equals 30
ANS is 30

2 plus 32 equals 34
34 multiply 3 equals 102
112 minus 102 equals 10
ANS is 10

3 multiply 2 equals 6
30 plus 6 equals 36
36 divide 36 equals 1
ANS is 1

12 plus 12 equals 24
12 minus 24 equals -12
12 plus -12 equals 0
ANS is 0

8 minus 1 equals 7
77 divide 7 equals 11
89 plus 11 equals 100
ANS is 100
```

Homework

- 限制:
  - 只會有加號 ”+” 、減號 ”-” 、乘號 ”*” 、除號 ”/” 、左括號 ”(” 、右括號 ”)” 、負號 ”-” 。
    - Ex.

```
12+32
56-2*3
13+6/2
(5+2)*3
-5+37
```

  - 答案都為整數解
  - 不會出現無法運算的情況(e.q. 除以0)

Homework

- 作業格式
  1. 封面：作業題目、班級、學號、姓名
  2. 作法：規則建立說明
  3. 程式碼
  4. 執行結果：測試資料及結果
  5. 討論：遭遇困難及解決過程
  6. 心得：自由發揮
- 作業期限
  - 將報告與程式碼打包成zip檔，上傳到ilearn
  - 2023/12/26 23:55前

Homework

- 請益時間：
  - 每周四 20:00 ~ 22:00
  - 資電203 Gprint
  - 有問題可至資電230
  - ilearn私訊或email
    - 連唯軒 M1207879@o365.fcu.edu.tw

延伸作業

- 試著用C/C++或其他語言撰寫parser程式，
  - 輸入及輸出如作業相同。
- 報告寫在同一份上，將程式碼一起上傳便可額外加分(3分)。

參考資料

[1] Yacc: Yet Another Compiler-Compiler (nthu.edu.tw)

[2] cs.ccu.edu.tw/~naiwei/cs5605/YaccBison.html