112 學年 第 2 學期 編譯器設計 小專題報告(2) 日期:2024/5/17

班級: 資工三 學號: B1043003 姓名: 陳麒安

1. 語法分析器(Parser) (共 1 題，100 分，滿分 100 分)

請於 2024/5/17 晚上 12:00 前透過數位學習園區繳交

參考指定教材 4.9 小節內容做為參考進行以下的實作，每小題須**貼關鍵程式碼，截圖呈現結果並文字說明。(注意: 請先使用 sudo hostname [學號]指令修改主機名稱為學號再截圖，否則不予計分。)**

A. 請以小專題報告(1)為例(可視狀況調整輸入原始程式碼)，編寫一個 YaCC/bison Specification，並以 Flex 建立 YaCC/bison 的詞彙分析器，以建立語法分析器(Parser)，語法分析器需能在分析語法後於螢幕上列印出分析的原始程式碼行數。(70 分)

專題報告(1)之 Python 程式碼：

```
print('Hello world')
num1 = 20
num2 = 3
add = num1 + num2
sub = num1 - num2
mul = num1*num2
div = num1/num2
print(add)
print(sub)
print(mul)
print(div)
```

Flex 之 lex.l 程式碼：

```
%{
#include <stdio.h>
int line_count = 0;
%}

%%
print                       { printf("Keyword: print\n"); }
[0-9]+                      { printf("Integer: %s\n", yytext); }
[a-zA-Z_][a-zA-Z0-9_]*      { printf("Identifier: %s\n", yytext); }
\".*\"                      { printf("String: %s\n", yytext); }
[+\-*/=]                    { printf("Operator: %s\n", yytext); }
[()]                        { printf("Parentheses: %s\n", yytext); }
[\n]                        { line_count++; printf("\n");}
[ \t]                       ;
```

```
.                                   { printf("Unrecognized: %s\n", yytext); }
%%

int main() {
    yyin = fopen("test.py", "r");
    if (!yyin) {
        fprintf(stderr, "Failed to open input file.\n");
        return 1;
    }

    yylex();
    fclose(yyin); // Close input file

    printf("********** Output Count **********\n");
    printf("Total lines: %d\n", line_count); // Print total lines

    return 0;
}
```

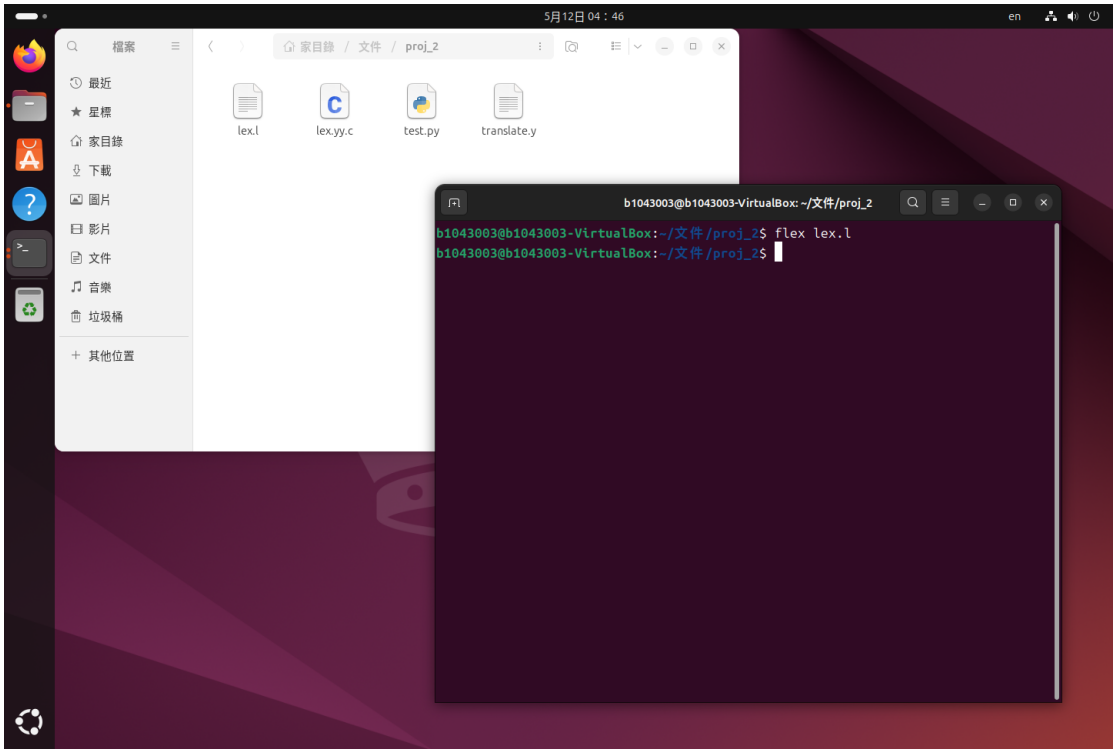
YaCC/bison Specification 之 translate.y 程式碼：

```
%{
#include <stdio.h>
extern int yylex();
extern int yyerror(const char *);
%}

%token PRINT INTEGER IDENTIFIER STRING OPERATOR

%%

program:
    statements
    | /* empty */
    ;

statements:
    statement
    | statements statement
    ;

statement:
    PRINT expression
```

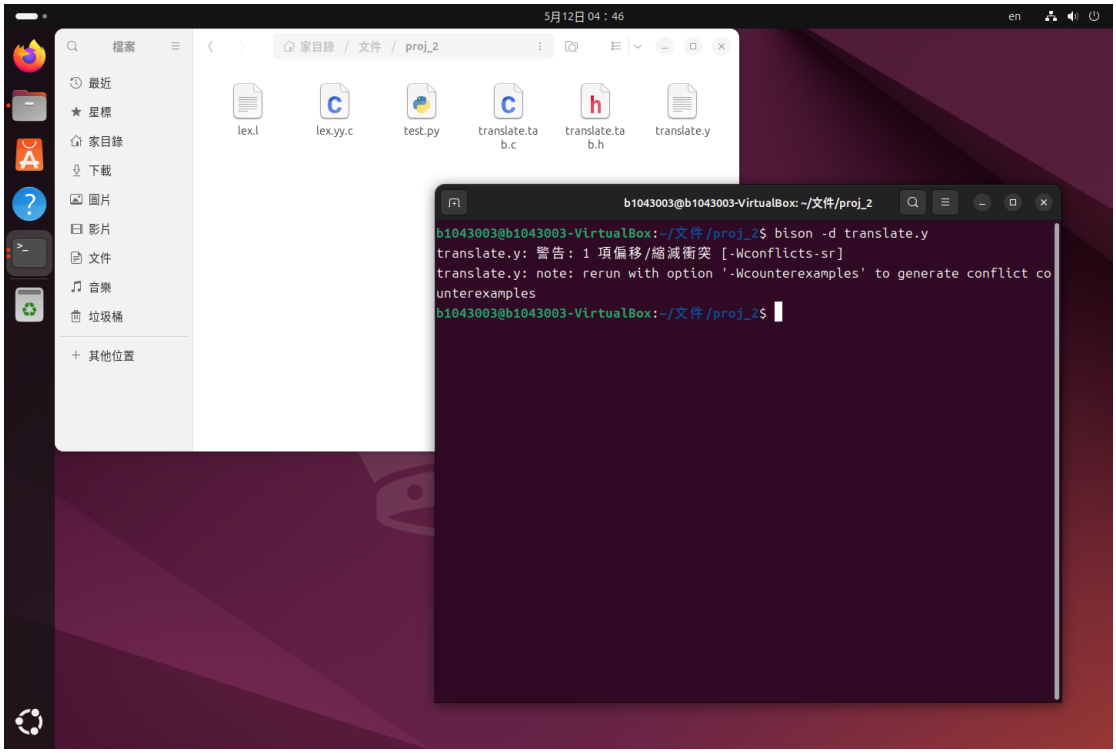
```
    | assignment
    ;

assignment:
    IDENTIFIER '=' expression
    ;

expression:
    INTEGER
    | IDENTIFIER
    | STRING
    | OPERATOR
    //| expression OPERATOR expression
    ;

%%

int yyerror(const char *s) {
    fprintf(stderr, "Parse error: %s\n", s);
    return 0;
}
```

這兩個檔案分別定義了一個簡單的語言的 lexer 和 parser。lexer 負責將輸入的程式碼分割成 token，而 parser 則負責檢查這些 token 是否符合語言的語法規則。

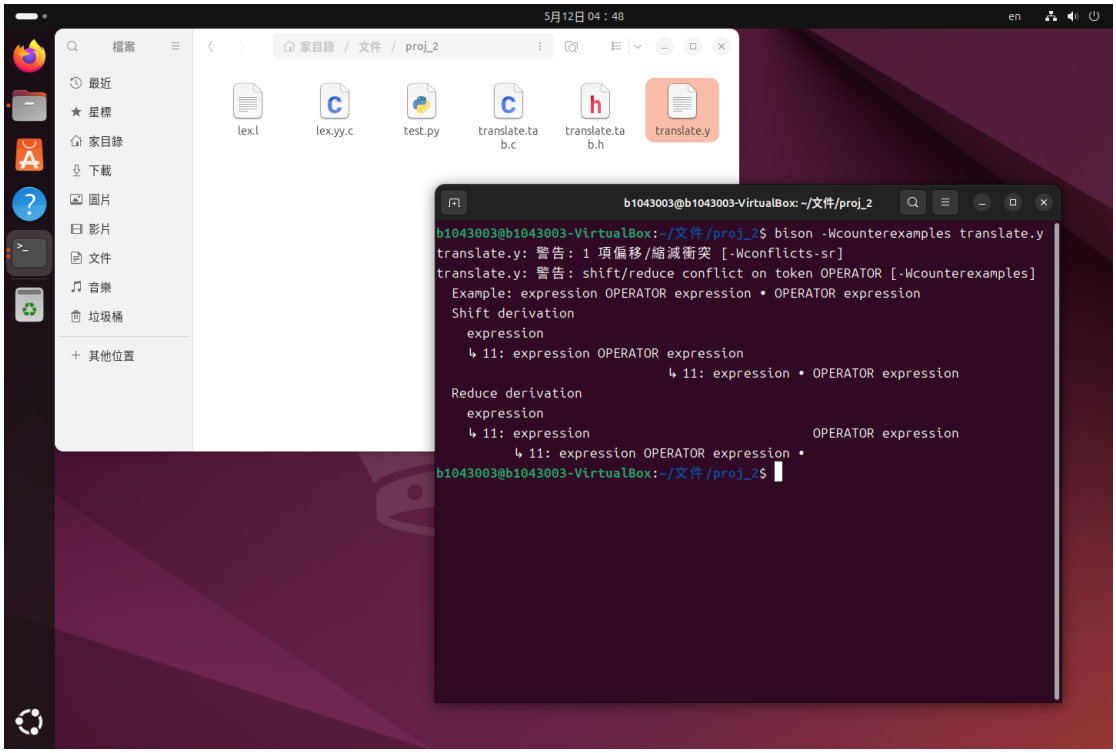
首先是 'lex.l' 檔案，它是 lexer 的定義。這個文件使用了 Flex（快速的 lexical 分析器產生器），它定義了匹配輸入中各種模式的規則。
接著，'translate.y' 檔案，它是 parser 的定義，使用的是 Bison。這個文件定義了語法規則以及如何解析 token。這兩個檔案結合起來實現了一個詞彙分析器與語法分析器。



圖一、執行‘flex lex.l’得到 lex.yy.c

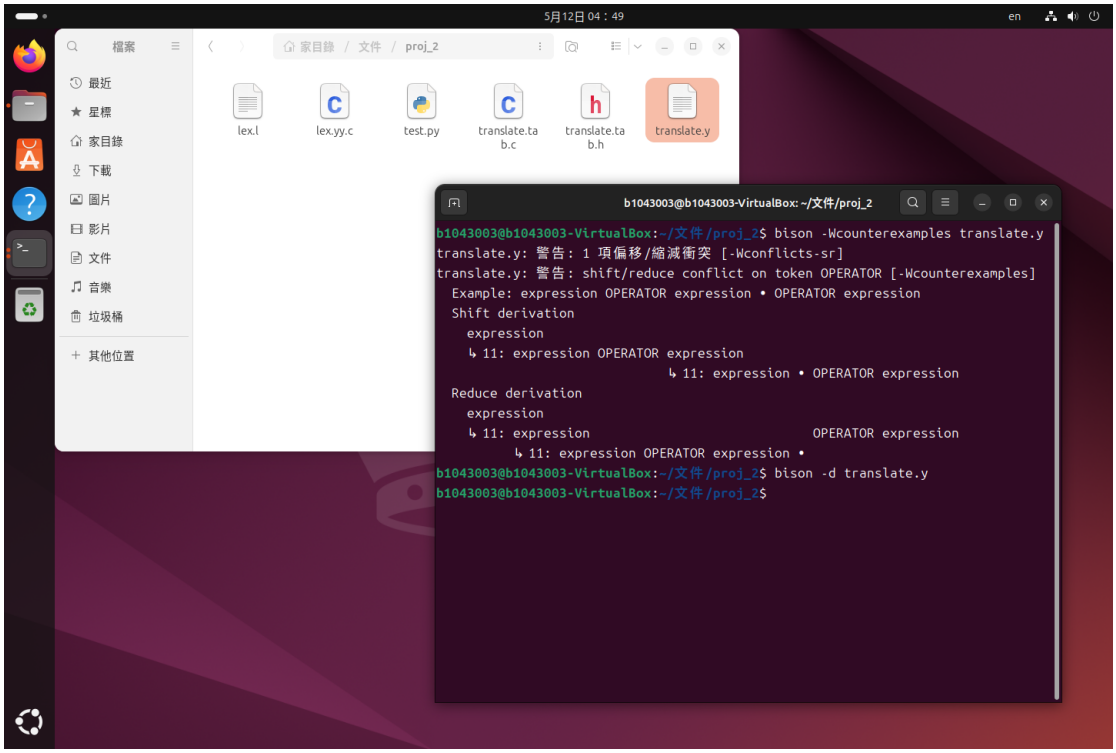


圖二、執行‘bison -d translate.y’出現警告

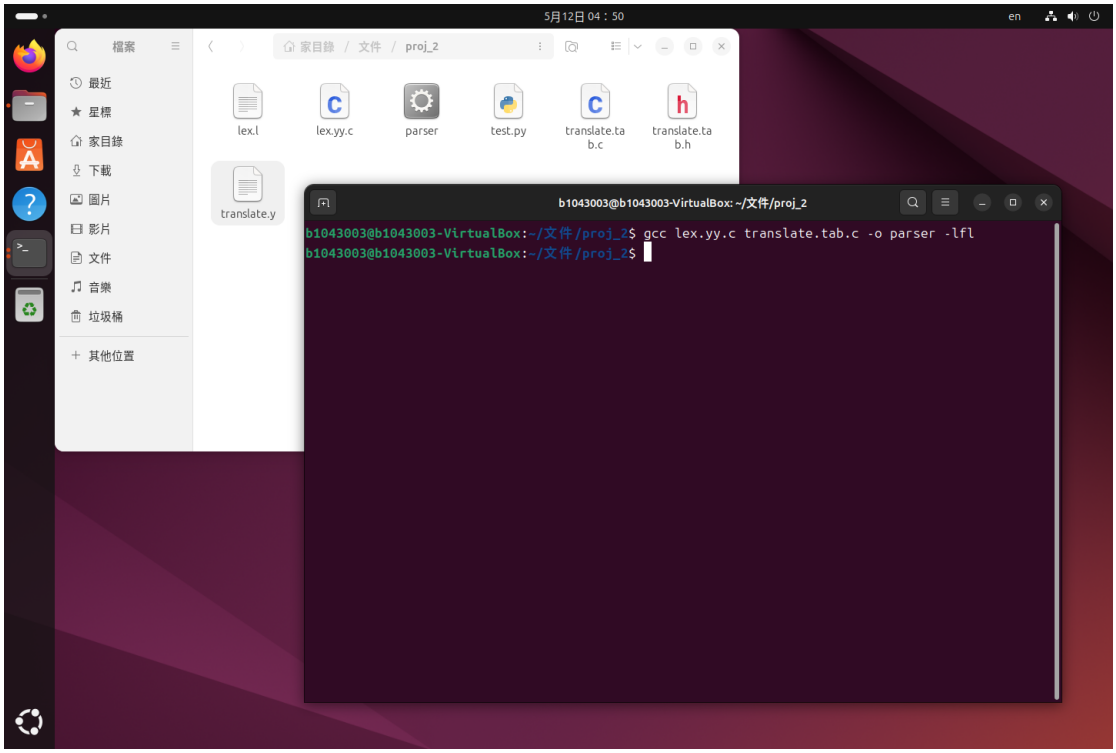


圖三、使用 ‘bison -Wconterexamples translate.y’ 查看詳細錯誤消息

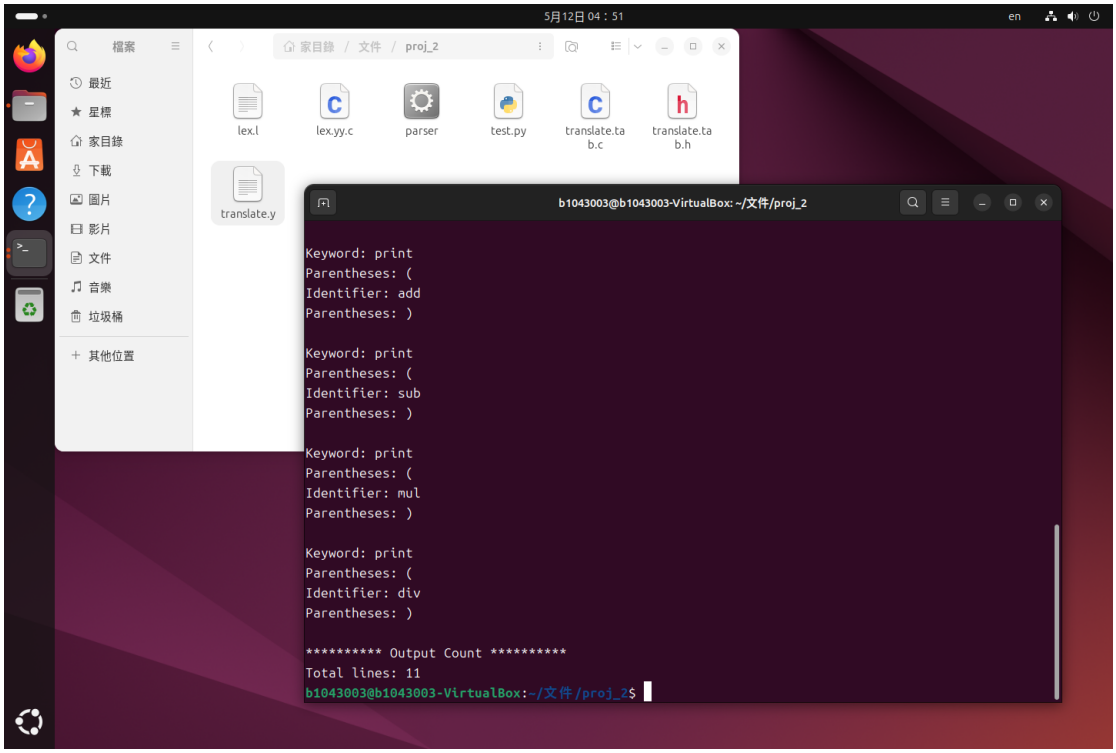
警告提到我的語法中存在移位/歸納衝突 (shift/reduce conflict)。發生在 OPERATOR 這個 token 上，這意味著在解析 expression 過程中存在不明確的操作。



圖四、對 translate.y 進行修正後警告消失



圖五、執行‘gcc lex.yy.c translate.tab.c -o parser -lfl’得到最終的 parser 執行檔



圖六、執行‘./parser’之結果，正確顯示 test.py 之內容與行數，詳細結果如下。

執行結果：

b1043003@b1043003-VirtualBox:~/文件/proj_2$ ./parser

Keyword: print

Parentheses: (

```
Unrecognized: '
Identifier: Hello
Identifier: world
Unrecognized: '
Parentheses: )

Identifier: num1
Operator: =
Integer: 20

Identifier: num2
Operator: =
Integer: 3

Identifier: add
Operator: =
Identifier: num1
Operator: +
Identifier: num2

Identifier: sub
Operator: =
Identifier: num1
Operator: -
Identifier: num2

Identifier: mul
Operator: =
Identifier: num1
Operator: *
Identifier: num2

Identifier: div
Operator: =
Identifier: num1
Operator: /
Identifier: num2

Keyword: print
Parentheses: (
Identifier: add
Parentheses: )
```

Keyword: print
Parentheses: (
Identifier: sub
Parentheses:)

Keyword: print
Parentheses: (
Identifier: mul
Parentheses:)

Keyword: print
Parentheses: (
Identifier: div
Parentheses:)

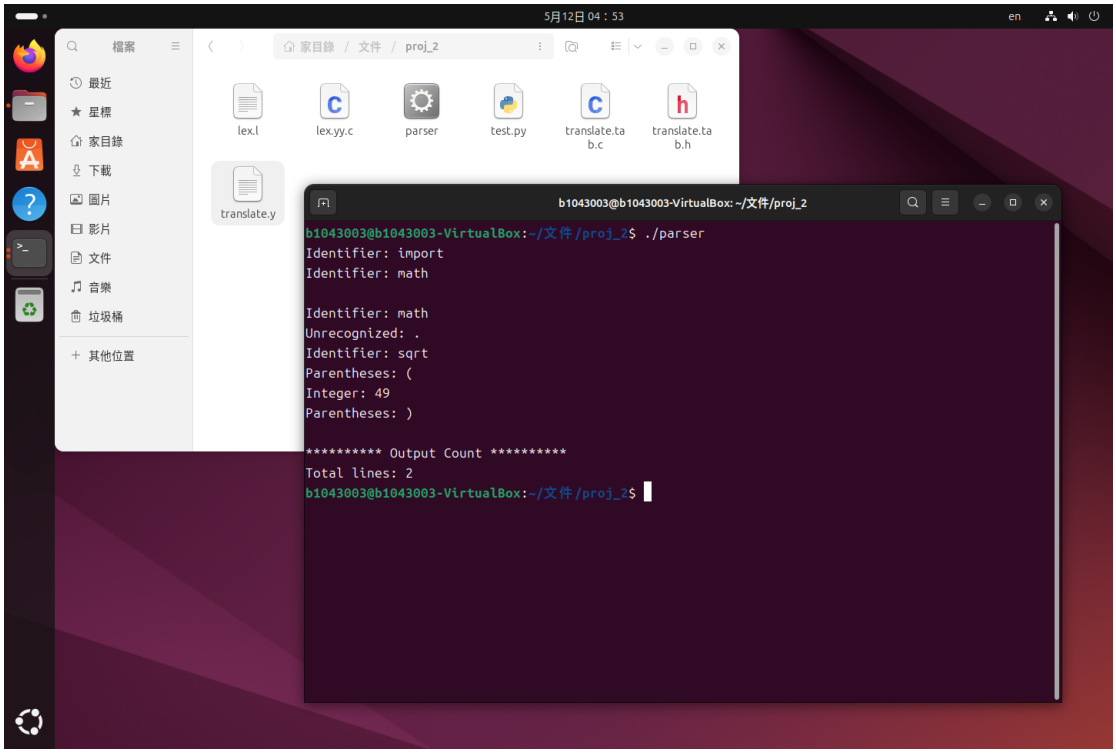
********** Output Count **********
Total lines: 11

B. 改寫 A 小題的語法分析器，使其能在螢幕上列印語法分析錯誤，並修改輸入原始程式碼至少產生兩處錯誤。 (30 分)

修改後之 Python 程式碼：

```
import math
math.sqrt(49)
```

修改後，在執行'./parser'後出現 'Identifier' ，代表輸入不符合語法定義檔案 (translate.y) 的規定。



圖七、修改 python 後之執行‘./parser’之結果