112 學年 第 2 學期 編譯器設計 小專題報告(1) 日期:2024/4/14

班級: 資工三 學號: B1043003 姓名: 陳麒安

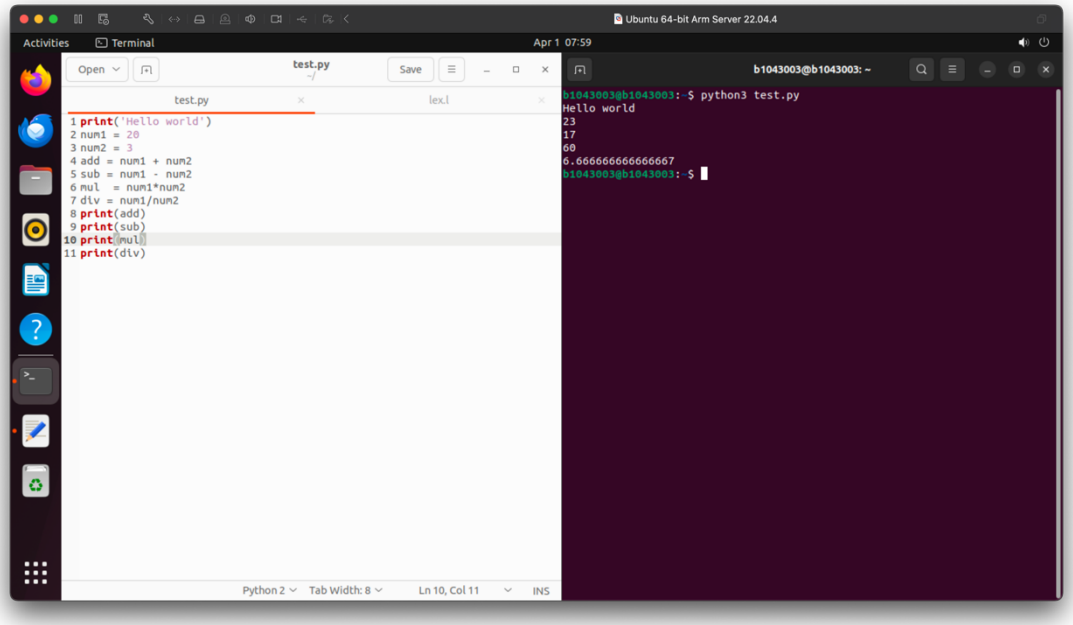
1. 詞彙分析器(Scanner) (共 1 題，100 分，滿分 100 分)

請於 2024/4/14 晚上 12:00 前透過數位學習 M 園區繳交

請以指定教材 3.5 小節內容做為參考進行以下的實作，每小題須**貼關鍵程式碼或腳本，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. 請自選一段至少 10 行以上的任何語言程式碼作為詞彙分析器之輸入(source program)。 (25 分)

```
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)
```



圖一、python 程式碼及執行畫面

B. 以程式碼語言的特性撰寫 FLEX 程式以定義詞彙單位(lex.l)。 (25 分)

選擇 Ubuntu 系統進行實作。

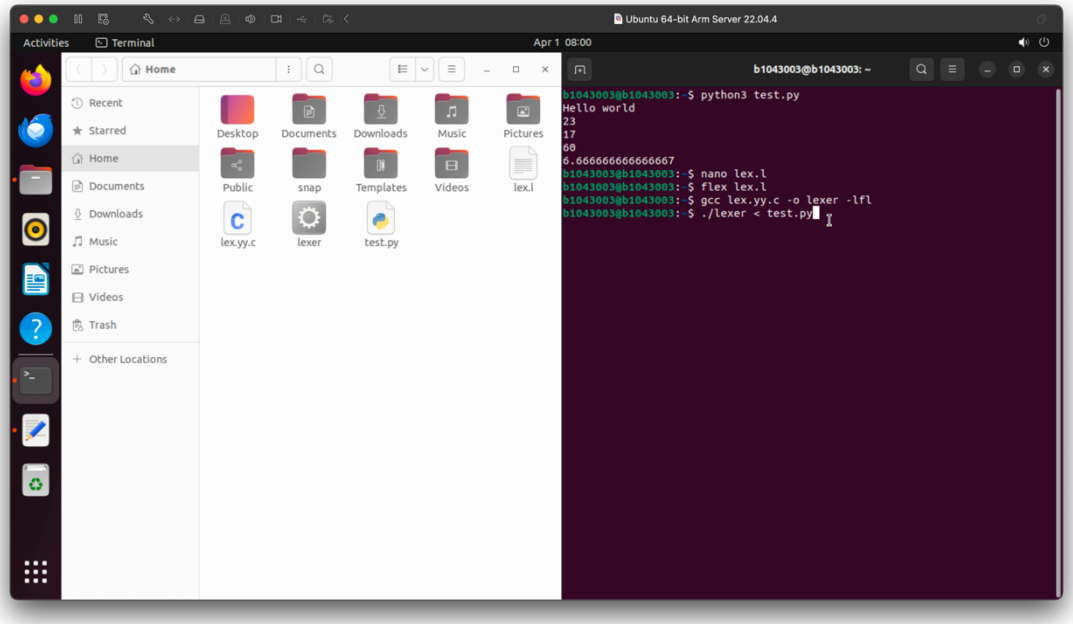
Lex 檔案內容：

```
%{
#include <stdio.h>

int keyword_count = 0;
int identifier_count = 0;
int integer_count = 0;
int string_count = 0;
int operator_count = 0;
int unrecognized_count = 0;
%}

%%
print                          { keyword_count++; printf("Keyword: print\n"); }
[0-9]+                         { integer_count++; printf("Integer: %s\n", yytext); }
[a-zA-Z_][a-zA-Z0-9_]*    { identifier_count++; printf("Identifier: %s\n",
yytext); }
\".*\"                         { string_count++; printf("String: %s\n", yytext); }
[+\-*/=]                       { operator_count++; printf("Operator: %s\n",
yytext); }
[ \t\n]                       ; // Ignore whitespace and newline
.                                { unrecognized_count++; printf("Unrecognized:
%s\n", yytext); }
%%

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

    yylex(); // Perform lexical analysis

    fclose(yyin); // Close input file
    printf("********** Output Count **********\n");
    printf("Keyword count: %d\n", keyword_count);
    printf("Identifier count: %d\n", identifier_count);
    printf("Integer count: %d\n", integer_count);
    printf("String count: %d\n", string_count);
    printf("Operator count: %d\n", operator_count);
    printf("Unrecognized count: %d\n", unrecognized_count);

    return 0;
}
```

C. 試用 FLEX 編譯器產生 lex.yy.c 程式原始碼，並以你的理解來分析說明。(25 分)

首先，在終端機中執行 “flex lex.l” 命令來編譯 Flex 程式並產生 C 語

言原始檔。接著使用 “gcc lex.yy.c -o lexer -lfl” 命令將 lex.yy.c 編譯成可執行檔。最後，執行 “flex lex.l” 命令將 Python 程式碼檔案作為輸入，得到運行 lexer 的結果。



圖二、撰寫 FLEX 程式並產生執行檔

當使用“gcc lex.yy.c -o lexer -lfl”處理 lex`檔案時，它會生成一個 C 程式檔案，其中`yy`是由`lex`工具產生的前綴，而`.c`則表示這是一個 C 程式檔案。

在`lex.yy.c`中，看到一些以`yy`為前綴的函式和變數，這些是由`lex`工具自動生成的。其中一些函式是`yylex()`，它是 lexical analyzer 的入口函式，用於從輸入文本中讀取字符並根據我在`lex`檔案中定義的規則進行匹配和處理。另外一些函式則是用來輔助`yylex()`函式的，比如用來返回匹配到的 token 的內容的`yytext`，以及用來向`yylex()`提供輸入字符的`yyinput()`。在`lex.yy.c`中，還會看到由`%{ ... %}`區塊中的程式碼直接複製到頂部，以及由`%%`區塊中的程式碼轉換為`yylex()`函式的規則和動作。

以我的觀察來看，`lex.yy.c`是一個可以用來對輸入文本進行 lexical analysis 的 C 程式。它包含了根據我在`lex`檔案中定義的規則自動生成的程式碼，這些規則用於識別和處理輸入文本中的不同 token，並且可以根據這些 token 進行後續的處理。

D. 試以 lex.yy.cc 編譯而成的程式作為詞彙分析器，，並以自選程式碼片段作為輸入，產出詞彙單位序列與詞彙統計。(25 分)

圖三、執行 ./lexer < test.py 之結果

執行結果：

b1043003@b1043003:~$ ./lexer < test.py
Keyword: print
Unrecognized: (
Unrecognized: '
Identifier: Hello
Identifier: world
Unrecognized: '
Unrecognized:)
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
Unrecognized: (
Identifier: add
Unrecognized:)
Keyword: print
Unrecognized: (
Identifier: sub
Unrecognized:)
Keyword: print
Unrecognized: (
Identifier: mul
Unrecognized:)
Keyword: print
Unrecognized: (
Identifier: div
Unrecognized:)
********** Output Count **********
Keyword count: 5
Identifier count: 20
Integer count: 2
String count: 0
Operator count: 10
Unrecognized count: 12