编译原理实验报告

实验四、 语法分析实验报告

姓　　名：刘宇诺

班　　级：计算机 20-1

学　　号：20201210207

任课教师：韩连金

目录

一、实验目的

理解语法分析在整个编译过程中的作用，掌握词法分析器生成工具 Flex 与语法分析器生成工具 Bison 的使用方法，并使用 Flex 和 Bison 实现简单的语法分析功能。

二、实验内容

1. 使用 Flex 和 Bison 实现可以对整型数据正确进行带有括号的四则运算的计算器程序。

输入输出示例：

输入：2*(1+3)/(6-3)

输出：= 2

2. 使用 Flex 和 Bison 实现一个可以分析符合以下规则的表达式的语法分析器。

词法规则：

(1)标识符：[a-zA-Z][0-9a-zA-Z]+

(2)数字：[0-9]+

文法规则：

<表达式> ::= <项> { <加法运算符> <项> }

<项> ::= <因子> { <乘法运算符> <因子> }

<因子> ::= <标识符> | <数字> | ‘(’ <表达式> ‘)’

<加法运算符> ::= + | -

<乘法运算符> ::= * | /

输入输出示例：

输入： a*2-3/4+b

a-*b+c

输出： syntax correct

syntax error

三、实验环境

Windows 7 系统或更高版本

gcc 5.4.0 或更高版本

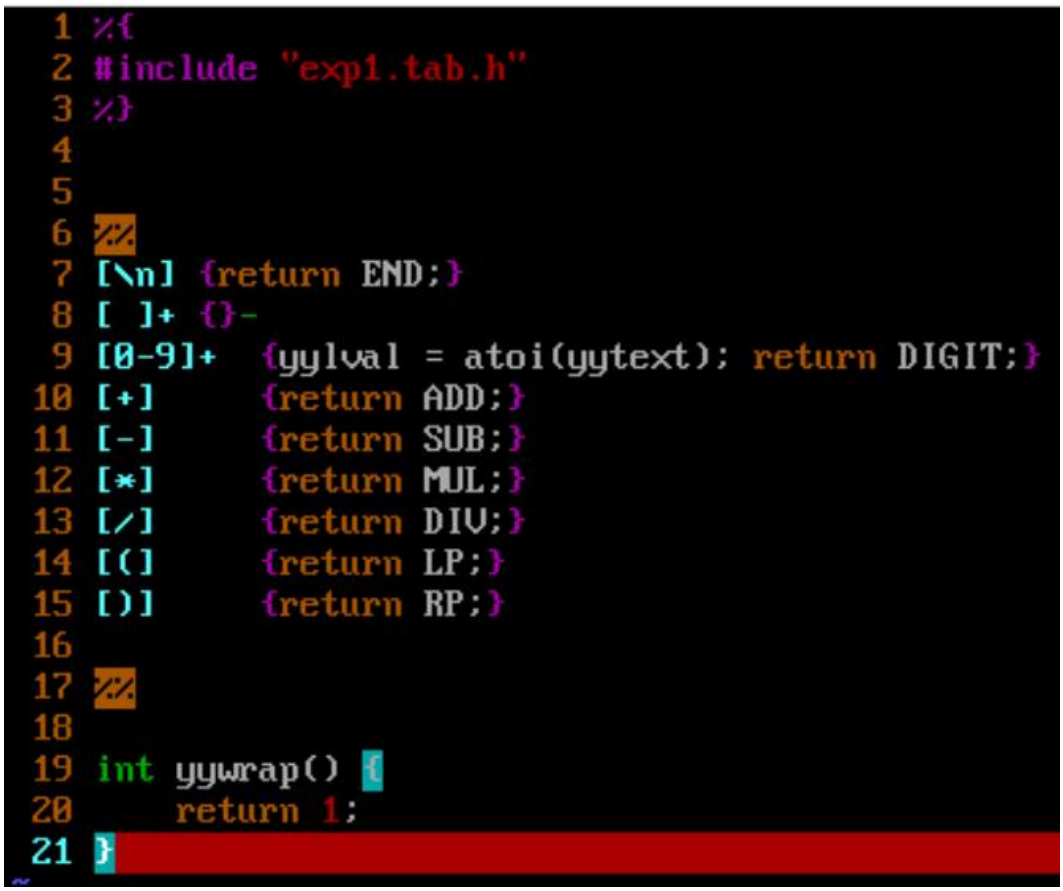
winflex & winbison 2.5.4 或更高版本

四、数据准备

data 文件夹下的 calc.txt 文件为实验内容 1 所需要计算的算术表达式，每行为一个带括号的四则运算；data 文件夹下的 expr.txt 文件为实验容 2 所需要进行语法分析的表达式，每行为一个合法或非法的表达式。

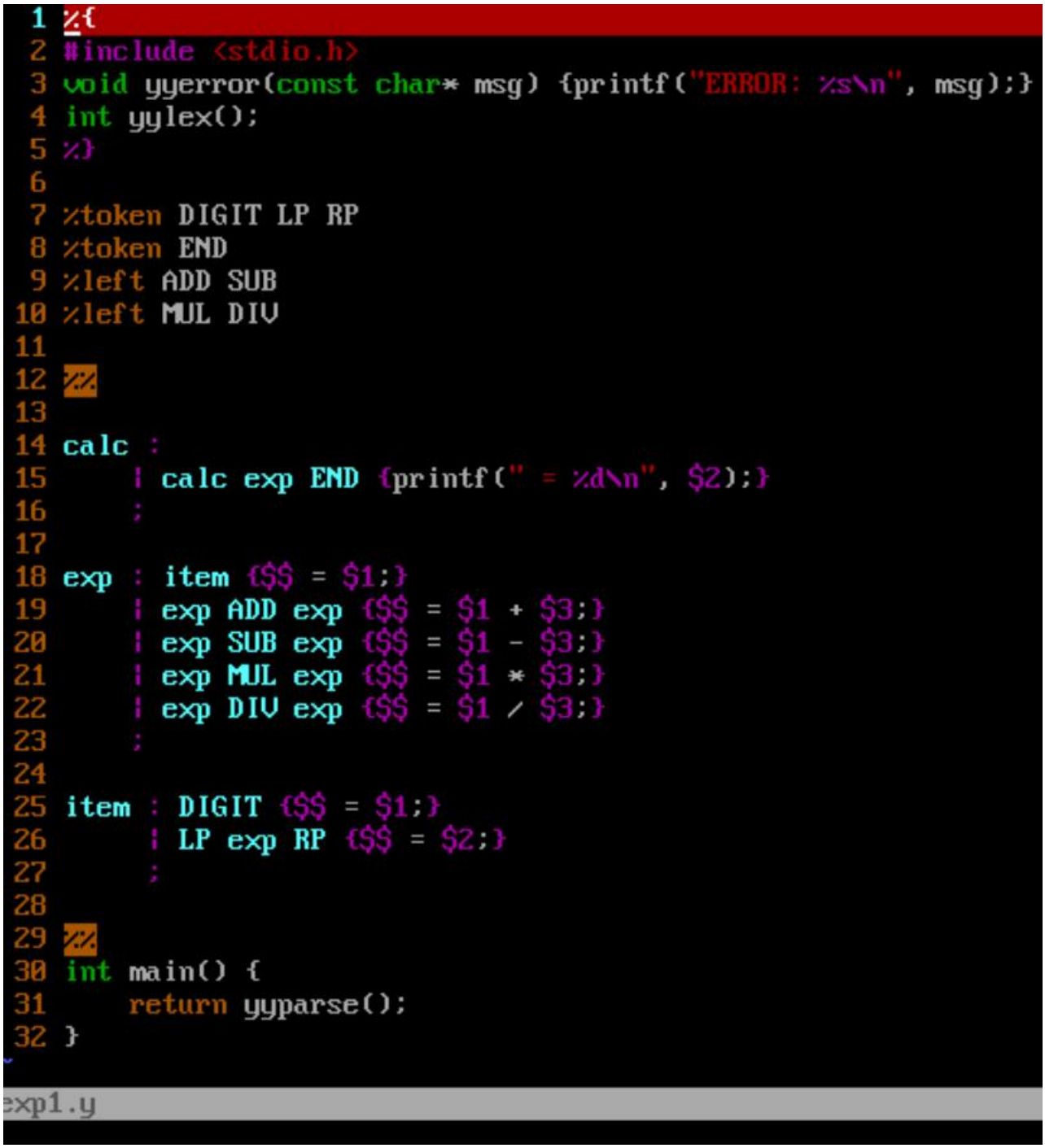
五、实验过程描述

第一计算 calc 实验的.l 文件，exp1.l



```
%{
#include "exp1.tab.h"
%}


%%
[\n] {return END;}
[ ]+ {}-
[0-9]+  {yylval = atoi(yytext); return DIGIT;}
[+]     {return ADD;}
[-]     {return SUB;}
[*]     {return MUL;}
[/]     {return DIV;}
[(]     {return LP;}
[)]     {return RP;}

%%

int yywrap() {
    return 1;
}
```

第一计算 calc 实验的.y 文件 exp1.y 文件



```
%{
#include <stdio.h>
void yyerror(const char* msg) {printf("ERROR: %s\n", msg);}
int yylex();
%}

%token DIGIT LP RP
%token END
%left ADD SUB
%left MUL DIV

%%

calc :
     | calc exp END {printf(" = %d\n", $2);}
     ;

exp : item {$$ = $1;}
    | exp ADD exp {$$ = $1 + $3;}
    | exp SUB exp {$$ = $1 - $3;}
    | exp MUL exp {$$ = $1 * $3;}
    | exp DIV exp {$$ = $1 / $3;}
    ;

item : DIGIT {$$ = $1;}
     | LP exp RP {$$ = $2;}
     ;

%%
int main() {
    return yyparse();
}
```

exp1.y

第二计算 expr 实验的.l 文件 exp2.l 文件

第二计算 expr 实验的.y 文件 exp2.y 文件

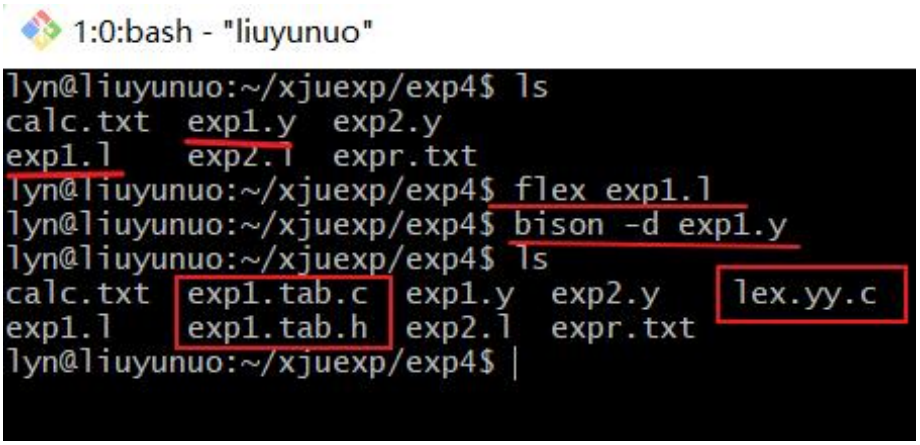
```
%{
#include <stdio.h>
void yyerror(const char* msg) {printf("ERROR: %s\n", msg);}
int yylex();
%}

%token DIGIT LP RP
%token END VAR
%left ADD SUB
%left MUL DIV

%%

line : exp END { printf("systax correct\n");}
exp : item
    | exp ADD item
    | exp SUB item
    ;
item : factor
     | item MUL factor
     | item DIV factor
     ;
factor : DIGIT
       | VAR
       | LP exp RP
       ;

%%
int main() {
    return yyparse();
}
```

exp2.y

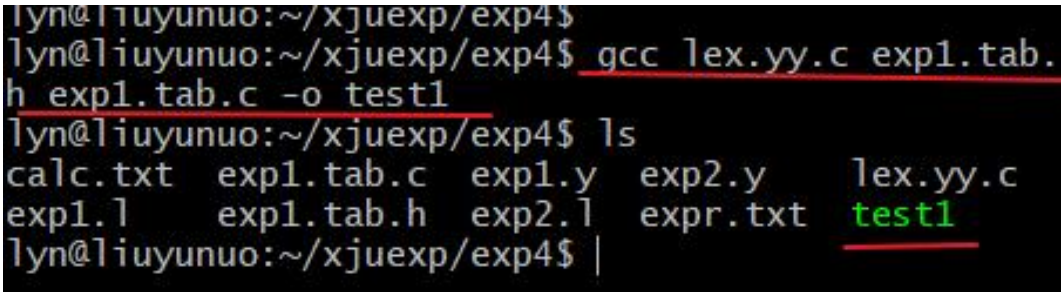
六、实验结果及分析

实验一：

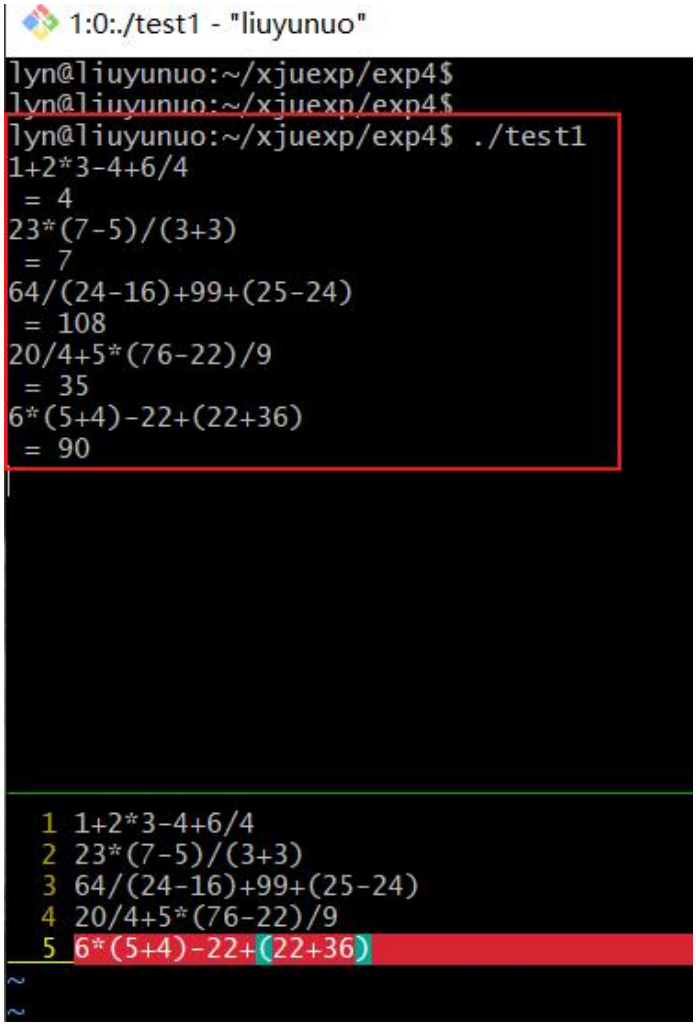
先通过 flex 和 bison 命令处理源文件



利用 gcc，将 lex.yy.c、exp1.tab.h 和 exp1.tab.c 文件一起进行编译，生成可执行文件 test1

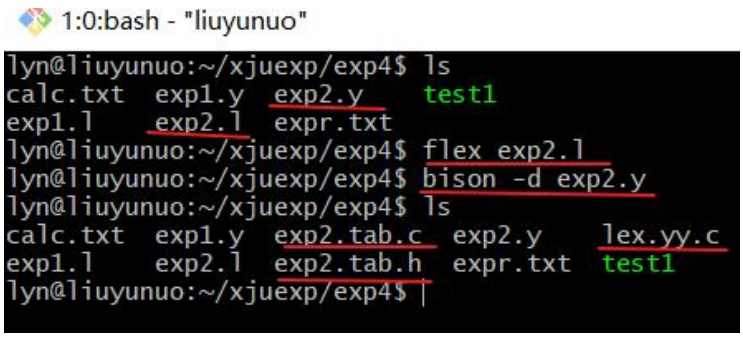


利用 calc.txt 文件中数据对 test1 进行测试，测试结果与预期相同。

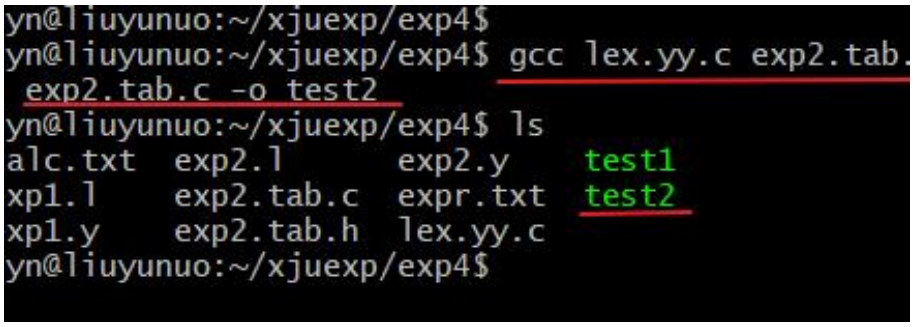


实验二：

通过 flex 和 bison 处理源文件，获得 lex.yy.c、exp2.tab.c、和 exp2.tab.h



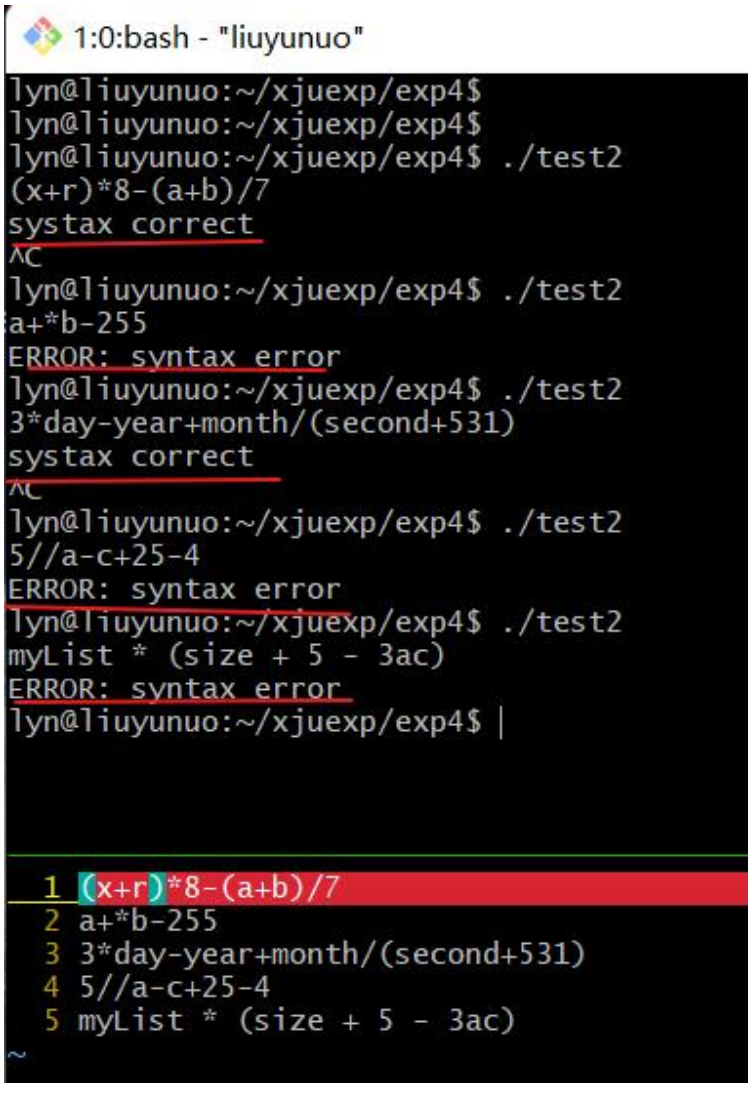
使用 gcc 将生成的三个文件一起进行编译生成 test2 文件



用 expr.txt 中的数据测试 test2。

测试结果和预期相同

第二个+*一起用，错误；第四个//两个除号，错误；第五个 3ac 数字开头的标识符，错误。



七、实验心得体会

学习了语法分析的过程，利用 linux 系统安装了 flex 和 bison以及gcc工具进行语法分析实验。通过实验进行实操练习，更加了解了语法分析器的生成过程，也锻炼了自己的动手能力。在一个有规定的文本文件中，通过 BNF 范式对语法进行定义，Bison 读取文件后，输出有相应语法分析功能的 C 语言程序。然后将 Flex 和 Bison 分别生成的具有词法和语法分析功能的 C 语言源程序一起进行编译后又，最终获得语法分析器。