中国矿业大学计算机学院

2017 级本科生课程设计报告

课程名称 系统软件开发实践

报告时间 2020 年 2 月 28 日

学生姓名 陆玺文

学　　号 03170908

专　　业 计算机科学与技术

任课教师 张博

成绩考核

编号	课程教学目标	占比	得分
1	**目标 1：**针对编译器中词法分析器软件要求，能够分析系统需求，并采用 FLEX 脚本语言描述单词结构。	15%	
2	**目标 2：**针对编译器中语法分析器软件要求，能够分析系统需求，并采用 Bison 脚本语言描述语法结构。	15%	
3	**目标 3：**针对计算器需求描述，采用 Flex/Bison 设计实现高级解释器，进行系统设计，形成结构化设计方案。	30%	
4	**目标 4：**针对编译器软件前端与后端的需求描述，采用软件工程进行系统分析、设计和实现，形成工程方案。	30%	
5	**目标 5：**培养独立解决问题的能力，理解并遵守计算机职业道德和规范，具有良好的法律意识、社会公德和社会责任感。	10%	
总成绩			
指导教师		评阅日期	

目 录

1、实验二　词法分析器

1.1 实验内容

1. 阅读《Flex/Bison.pdf》第一、二章，掌握 Flex 基础知识。
2. 利用 Flex 实现用于 C 语言子集 C_1 的词法扫描器。

1.2 实验步骤

在第一次实验的基础上，同时参考书籍《ANSI C grammar(Lex)》，首先针对关键字、专用符号、标识符、整型常熟、空白、注释给定相应的模式匹配规则，接着编写计算行列的函数 Count（）完成匹配字符的行列输出。

1.3 Flex 源代码说明

为了尝试使用返回整型常数的方式来完成模式匹配并执行相应动作，宏定义各个类型的字符码，并在模式匹配的动作中返回相应的码值，主函数中在调用函数去根据码值执行相应的动作。

Lex2-2.l 在 lex2-1.l 的基础上增加计算行列数的函数 Count（）而成，详细代码如代码 1-1 所示。

代码 1-1　lex2-2.l

```
1.  %{
2.  #include <stdio.h>
3.  #define LT              1
4.  #define   LE            2
5.  #define GT              3
6.  #define   GE            4
7.  #define   EQ            5
8.  #define NE              6
9.  #define ID              20
10. #define NUMBER          21
11. #define RELOP           22
12. #define MAIN            44
13. #define INT              45
14. #define FLOAT                   46
15. #define RETURN                  48
16. #define CONST                   49
17. #define WS                       51
18. #define INCLUDE                    59
19. #define NEWLINE            23
20. #define OTHER           24
21. #define STRING             26
```

```
22.
23.  int yylval;
24.  int column=0;
25.  int row=0;
26.  %}
27.
28.  delim        [ \t \n]
29.  ws        {delim}+
30.  letter       [A-Za-z_]
31.  schar        \'(\\.|[^"\\])\'
32.  string       \"(\\.|[^"\\])*\"
33.  digit        [0-9]
34.  H         [a-fA-F0-9]
35.  id        ({letter}|\_)(\_|{letter}|{digit})*
36.  number
        {digit}+(\.{digit}+)?([eE][+-]?{digit}+)?([uUlL]|([u
     U][lL])|([lL][uU]))?
37.
38.  %%
39.  {ws}            {return WS;}
40.  "/*"([^\*]|(\*)*[^\*/])*(\*)*"*/"  {;}
41.  "/""/"[^\n]*          {;}
42.
43.  main                  {yylval=MAIN;return(MAIN);}
44.  int                   {yylval=INT;return(INT);}
45.  float                 {yylval=FLOAT;return(FLOAT);}
46.  return                {yylval=RETURN;return(RETURN);}
47.  0[0-7]*               {yylval=NUMBER;return(NUMBER);}
48.  0[xX]{H}+             {yylval=NUMBER;return(NUMBER);}
49.  "#"include
                          {yylval=INCLUDE;return(INCLUDE);}
50.
51.  {id}            { return (ID);}
52.  {number}        { return (NUMBER);}
53.  {string}     {return (STRING);}
54.  "<<"      {return(RELOP);}
55.  "<"           {yylval = LT; return (RELOP);}
56.  "<="           {yylval = LE; return (RELOP);}
57.  "="           {yylval = EQ; return (RELOP);}
58.  "<>"           {yylval = NE; return (RELOP);}
59.  ">"           {yylval = GT; return (RELOP);}
60.  ">="           {yylval = GE; return (RELOP);}
61.  "+"    {return(RELOP);}
62.  "/"    {return(RELOP);}
```

```
"{"    {return(RELOP);}
"}"    {return(RELOP);}
";"    {return(RELOP);}
"("    {return(RELOP);}
")"    {return(RELOP);}
.      {yylval = OTHER; return OTHER;}

%%
int yywrap (){
  return 1;
}
void count(){
    int i;
    for(i=0;yytext[i]!='\0';++i){
        if(yytext[i]=='\n'){
            column=0;
            row++;
        }
        else if(yytext[i]=='\t')
            column+=8-(column%8);
        else
            column++;
    }

}
void writeout(int c){
  switch(c){
    case OTHER: printf("   OTHER: %s", yytext);break;
    case RELOP: printf("   OTHER: %s", yytext);break;
    case NUMBER: printf("     NUM: %s", yytext);break;
    case ID:printf("      ID: %s", yytext);break;
    case NEWLINE:break;
    case STRING:printf("  STRING: %s",yytext);break;
    case MAIN:printf("    MAIN: %s",yytext);break;
    case INT:printf("     INT: %s",yytext);break;
    case FLOAT:printf("   FLOAT: %s",yytext);break;
    case RETURN:printf("  RETURN: %s",yytext);break;
    case WS:break;
    case INCLUDE:printf(" INCLUDE: %s",yytext);break;
    default:break;
  }
  if(c!=WS){
    if(yyleng<8)printf("\t");
    printf("\t\t row:%d \t column:%d \n",row,column);
```

```
107.   }
108.   count();
109.   return;
110. }
111. int main (int argc, char ** argv){
112.    int c=0;
113.    while (c = yylex()){
114.       writeout(c);
115.    }
116.    return 0;
117. }
118. int yyerror(char *s){
119.    fprintf(stderr,"%s\n",s);
120.    return 1;
121. }
```

第一段行 1~26 中，给定了各个类型的整型返回值以及所使用的全局变量，yylval 是为之后的学习准备，给出每个字符串的类型值。第二段行 28 到 69 中，给定各个模式匹配的规则以及相应的动作，其中注释部分参考了网络博文资料（地址：https://blog.ostermiller.org/finding-comments-in-source-code-using-regular-expressions/ ）能够支持更加负责的形如“/*…/n…*…*/”等的注释。

8 进制与 16 进制数的识别参考参考书目给定，在行 47,48 中。

行 36 根据自己的理解给定了整合浮点数和带后缀数的识别规则。

行 74~87 为参考参考资料给定的计算行列值的代码，在每一次模式匹配完成之后加以调用，更新计算下一次的起始行列值。

行 105 针对匹配到的不同的字符串长度不一，做了一个简单的输出格式处理，使得制表符能够让输出相对整齐一些。

在行 88 开始的 writeout 函数中进行主要的动作执行，其中执行到空格与换行符时直接跳出不打印字符，并在行 104 进行判断，进而跳过打印行列值。但是计算行列值的 count 函数仍然正常执行。

考虑到定义上的适当简便，整个程序的关键字基本做到了**一符一码**，操作符基本都使用了 Relop.

1.4 实验结果

Lex2-2 为完备实验，lex2-1 相比之下没有处理行列值，以及 8 进制数。

1.4.1 Windows 下实验结果

1.4.1.1 Lex2-1.l 运行结果

```
E:\学习资源\CS平台课\系统软件开发实践\Flex实验2>lex2-1.exe<2-1.cpp
 INCLUDE: #include
   OTHER: <
      ID: iostream
   OTHER: >
      ID: using
      ID: namespace
      ID: std
     INT: int
    MAIN: main
   OTHER: (
   OTHER: )
   OTHER: {
   FLOAT: float
      ID: a
   OTHER: =
     NUM: 4.90867e-2
   OTHER: ;
     INT: int
      ID: b
   OTHER: =
     NUM: 0
      ID: xE124
   OTHER: ;
     INT: int
      ID: c
   OTHER: =
     NUM: 0167
   OTHER: ;
      ID: cout
   OTHER: <<
  STRING: "Hello ! "
   OTHER: <<
      ID: endl
   OTHER: ;
      ID: cout
   OTHER: <<
  STRING: "Welcome to c++! "
   OTHER: <<
      ID: endl
   OTHER: ;
  RETURN: return
     NUM: 0
   OTHER: ;
   OTHER: }
```

图 1-1　lex2-1 结果

可以看到 16 进制数 0xE124 尚未正确识别。

1.4.1.2 Lex2-2 结果

```
E:\学习资源\CS平台课\系统软件开发实践\Flex实验2>lex2-2.exe<2-1.cpp
 INCLUDE: #include                row:0   column:0
   OTHER: <                       row:0   column:9
      ID: iostream                row:0   column:10
   OTHER: >                       row:0   column:18
      ID: using                   row:1   column:0
      ID: namespace               row:1   column:6
      ID: std                     row:1   column:16
     INT: int                     row:2   column:0
    MAIN: main                    row:2   column:4
   OTHER: (                       row:2   column:8
   OTHER: )                       row:2   column:9
   OTHER: {                       row:3   column:0
   FLOAT: float                   row:4   column:3
      ID: a                       row:4   column:9
   OTHER: =                       row:4   column:11
     NUM: 4.90867e-2              row:4   column:13
   OTHER: ;                       row:4   column:23
     INT: int                     row:5   column:3
      ID: b                       row:5   column:7
   OTHER: =                       row:5   column:9
     NUM: 0xE124                          row:5   column:11
   OTHER: ;                       row:5   column:17
     INT: int                     row:6   column:3
      ID: c                       row:6   column:7
   OTHER: =                       row:6   column:9
     NUM: 0167                    row:6   column:11
   OTHER: ;                       row:6   column:15
      ID: cout                    row:8   column:3
   OTHER: <<                      row:8   column:7
  STRING: "Hello ! "              row:8   column:9
   OTHER: <<                      row:8   column:19
      ID: endl                    row:8   column:21
   OTHER: ;                       row:8   column:25
      ID: cout                    row:10          column:3
   OTHER: <<                      row:10          column:7
  STRING: "Welcome to c++! "              row:10          column:9
   OTHER: <<                      row:10          column:27
      ID: endl                    row:10          column:29
   OTHER: ;                       row:10          column:33
  RETURN: return                          row:11          column:3
     NUM: 0                       row:11          column:10
   OTHER: ;                       row:11          column:11
   OTHER: }                       row:12          column:0
```

图 1-2　lex2-2 结果

可以看到行列值、字符串、不同进制数都得到了正确识别。

1.4.2 Linux 环境下运行结果

第一次实验使用的虚拟机 Ubuntu 出现了异常崩溃，改用在腾讯云服务器上的 CentOS 上进行了实验。

1.4.2.1 Lex2-1

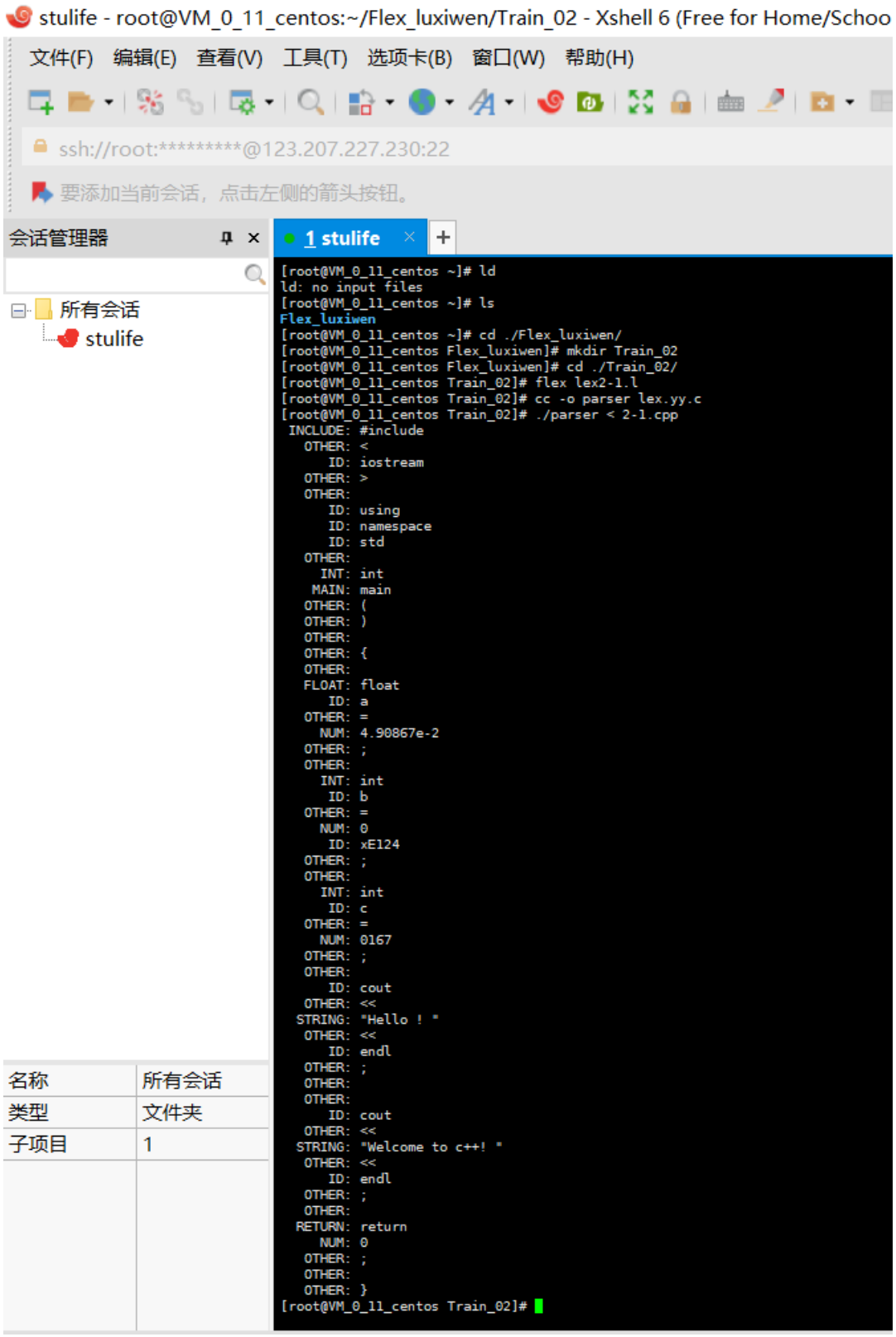

图 1-3　Linux 下 lex2-1 运行结果

1.4.2.2 Lex2-2

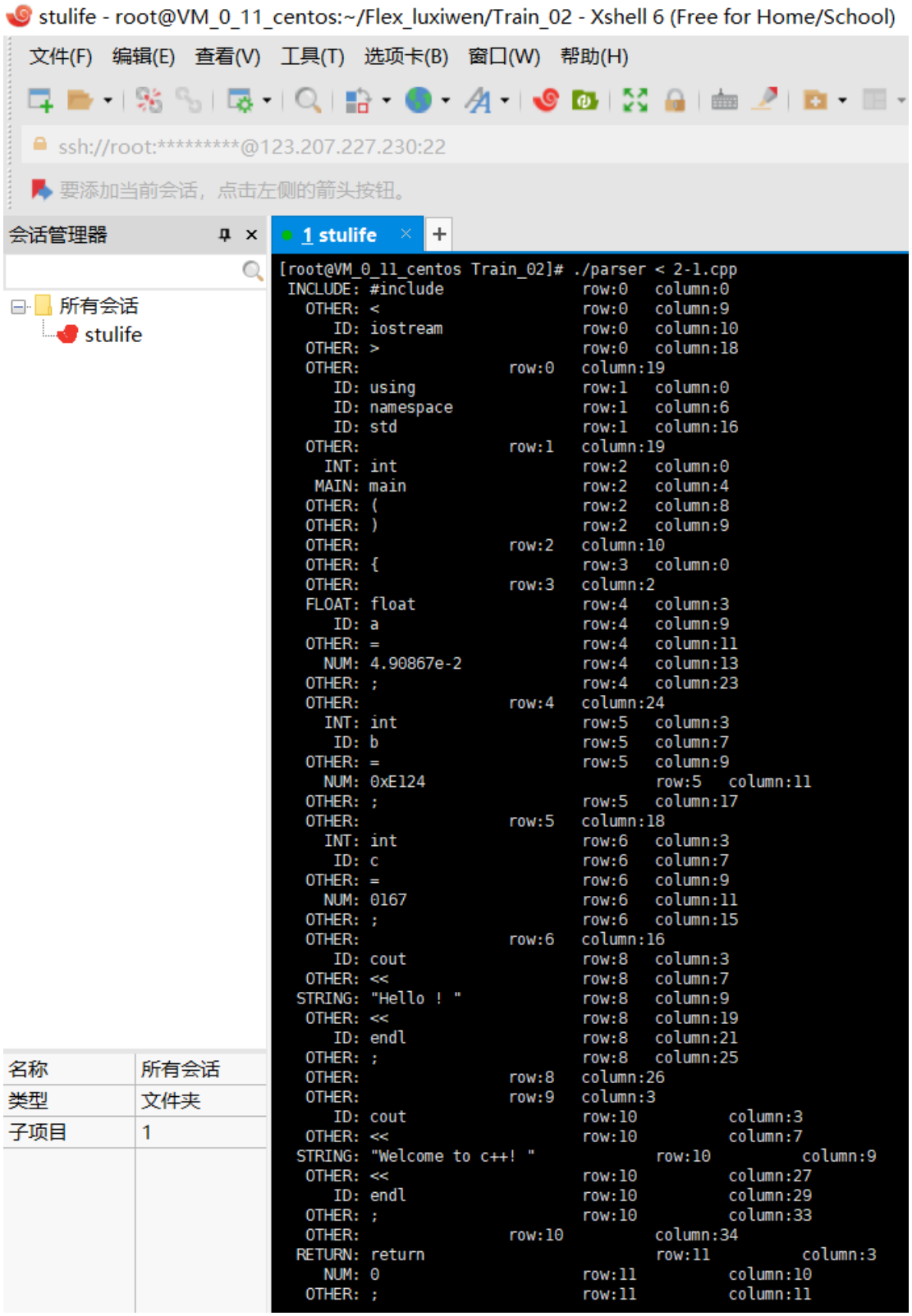

图 1-4 Linux 下 lex2-2 运行结果

1.5 实验总结

1.5.1 遇到的难题

行列值的识别在一开始并没有思路，尝试过单独识别换行符，以及通过任意字符的方式来计算列值，最后阅读参考的代码后在多次尝试下成功完成。

1.5.2 对程序的评价

总体上较好完成了任务，同时使用返回值，然后判断码值执行动作的方式，较之于第一次实验的在每一个模式匹配中嵌入动作有所进步。不过总体上的代码还可以更加简洁，有待提高。

1.5.3 实验收获

这一次实验进一步熟悉了使用 Flex 构造词法分析器的步骤，对于一款编译器的诞生有了更加进一步的感受，同时学习了返回值之后执行动作，为之后学习打表做了准备。