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](#_Toc33781888)

[1.1 实验内容 1](#_Toc33781889)

[1.2 实验步骤 1](#_Toc33781890)

[1.3 Flex源代码说明 1](#_Toc33781891)

[1.4 实验结果 4](#_Toc33781892)

[1.4.1 Windows下实验结果 4](#_Toc33781893)

[1.4.2 Linux环境下运行结果 6](#_Toc33781894)

[1.5 实验总结 9](#_Toc33781895)

[1.5.1 遇到的难题 9](#_Toc33781896)

[1.5.2 对程序的评价 9](#_Toc33781897)

[1.5.3 实验收获 9](#_Toc33781898)

# 实验二 词法分析器

## 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C1的词法扫描器。

## 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。

## 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

|  |  |
| --- | --- |
|  | **%{** |
|  | #include <stdio.h> |
|  | #define LT 1 |
|  | #define LE 2 |
|  | #define GT 3 |
|  | #define GE 4 |
|  | #define EQ 5 |
|  | #define NE 6 |
|  | #define ID 20 |
|  | #define NUMBER 21 |
|  | #define RELOP 22 |
|  | #define MAIN 44 |
|  | #define INT 45 |
|  | #define FLOAT 46 |
|  | #define RETURN 48 |
|  | #define CONST 49 |
|  | #define WS 51 |
|  | #define INCLUDE 59 |
|  | #define NEWLINE 23 |
|  | #define OTHER 24 |
|  | #define STRING 26 |
|  |  |
|  | **int** yylval**;** |
|  | **int** column**=**0**;** |
|  | **int** row**=**0**;** |
|  | **%}** |
|  |  |
|  | delim **[** \t \n**]** |
|  | ws **{**delim**}+** |
|  | letter **[**A**-**Za**-**z\_**]** |
|  | schar \'(\\.|[^"\\])\' |
|  | string \**"(\\.|[^"**\\**])\***\**"** |
|  | digit **[**0**-**9**]** |
|  | H **[**a**-**fA**-**F0**-**9**]** |
|  | id **({**letter**}|**\\_**)(**\\_**|{**letter**}|{**digit**})\*** |
|  | number **{**digit**}+(**\**.{**digit**}+)?([**eE**][+-]?{**digit**}+)?([**uUlL**]|([**uU**][**lL**])|([**lL**][**uU**]))?** |
|  |  |
|  | **%%** |
|  | **{**ws**}** **{return** WS**;}** |
|  | **"/\*"([^**\**\*]|(**\**\*)\*[^**\**\*/])\*(**\**\*)\*"\*/"** **{;}** |
|  | **"/""/"[^**\n**]\*** **{;}** |
|  |  |
|  | main **{**yylval**=**MAIN**;return(**MAIN**);}** |
|  | **int** **{**yylval**=**INT**;return(**INT**);}** |
|  | **float** **{**yylval**=**FLOAT**;return(**FLOAT**);}** |
|  | **return** **{**yylval**=**RETURN**;return(**RETURN**);}** |
|  | 0**[**0**-**7**]\*** **{**yylval**=**NUMBER**;return(**NUMBER**);}** |
|  | 0**[**xX**]{**H**}+** **{**yylval**=**NUMBER**;return(**NUMBER**);}** |
|  | **"#"**include **{**yylval**=**INCLUDE**;return(**INCLUDE**);}** |
|  |  |
|  | **{**id**}** **{** **return** **(**ID**);}** |
|  | **{**number**}** **{** **return** **(**NUMBER**);}** |
|  | **{**string**}** **{return** **(**STRING**);}** |
|  | **"<<"** **{return(**RELOP**);}** |
|  | **"<"** **{**yylval **=** LT**;** **return** **(**RELOP**);}** |
|  | **"<="** **{**yylval **=** LE**;** **return** **(**RELOP**);}** |
|  | **"="** **{**yylval **=** EQ**;** **return** **(**RELOP**);}** |
|  | **"<>"** **{**yylval **=** NE**;** **return** **(**RELOP**);}** |
|  | **">"** **{**yylval **=** GT**;** **return** **(**RELOP**);}** |
|  | **">="** **{**yylval **=** GE**;** **return** **(**RELOP**);}** |
|  | **"+"** **{return(**RELOP**);}** |
|  | **"/"** **{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**);** |
|  | **}** |
|  | count**();** |
|  | **return;** |
|  | **}** |
|  | **int** main **(int** argc**,** **char** **\*\*** argv**){** |
|  | **int** c**=**0**;** |
|  | **while** **(**c **=** yylex**()){** |
|  | writeout**(**c**);** |
|  | **}** |
|  | **return** 0**;** |
|  | **}** |
|  | **int** yyerror**(char** **\***s**){** |
|  | fprintf**(**stderr**,"%s\n",**s**);** |
|  | **return** 1**;** |
|  | **}** |

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.

## 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。

### Windows下实验结果

#### Lex2-1.l运行结果

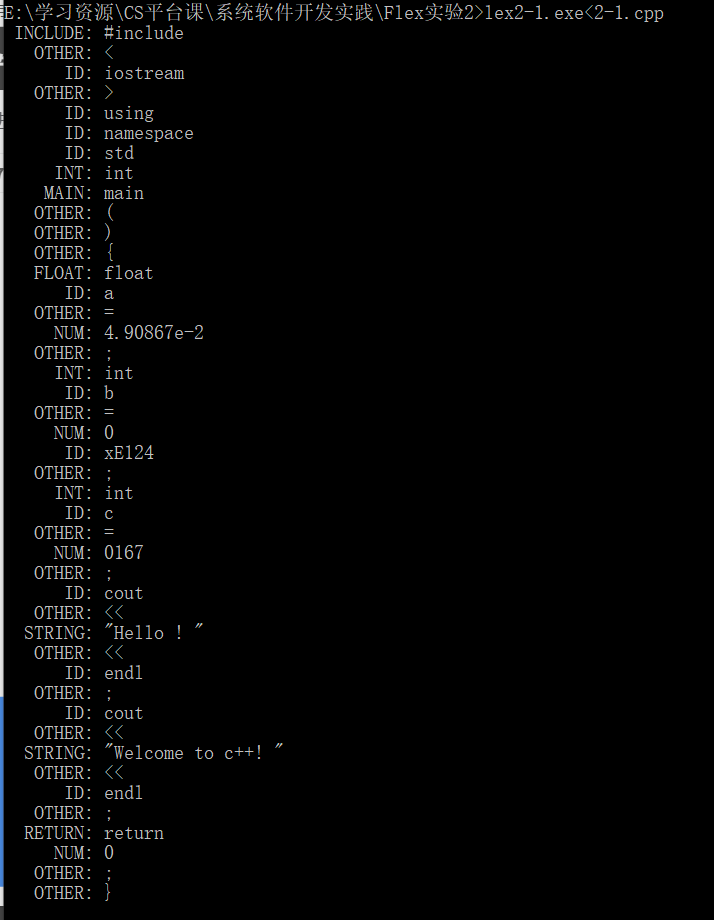


图 ‑1 lex2-1结果

可以看到16进制数0xE124尚未正确识别。

#### Lex2-2结果

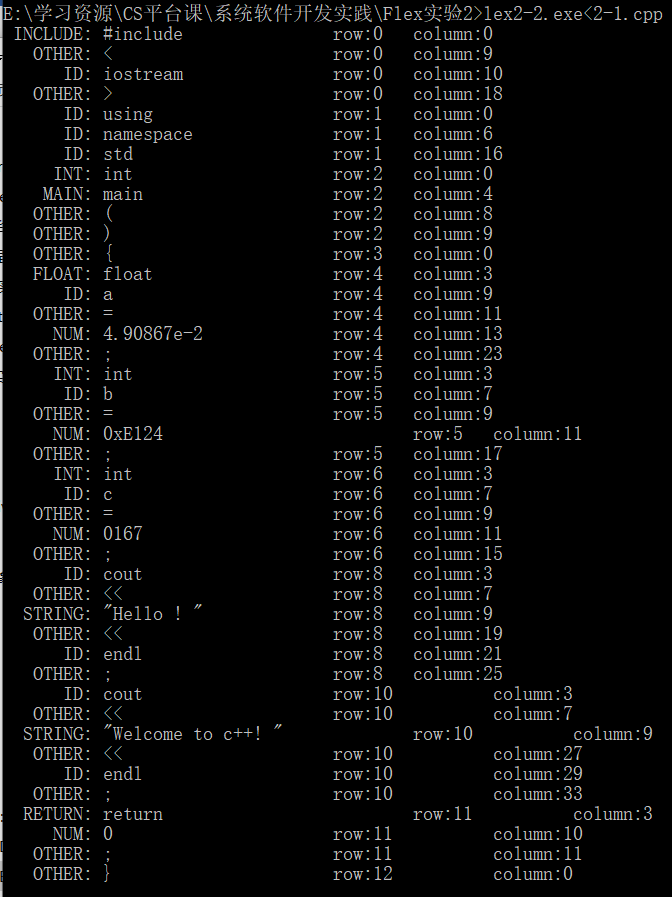


图 ‑2 lex2-2结果

可以看到行列值、字符串、不同进制数都得到了正确识别。

### 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。

#### Lex2-1

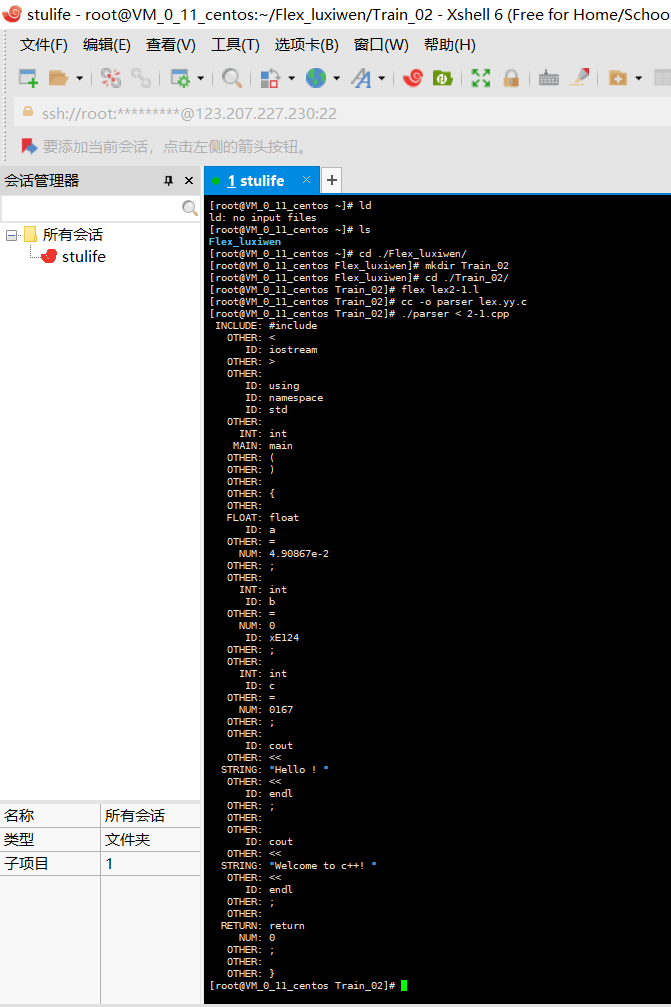


图 ‑3 Linux下lex2-1运行结果

#### Lex2-2

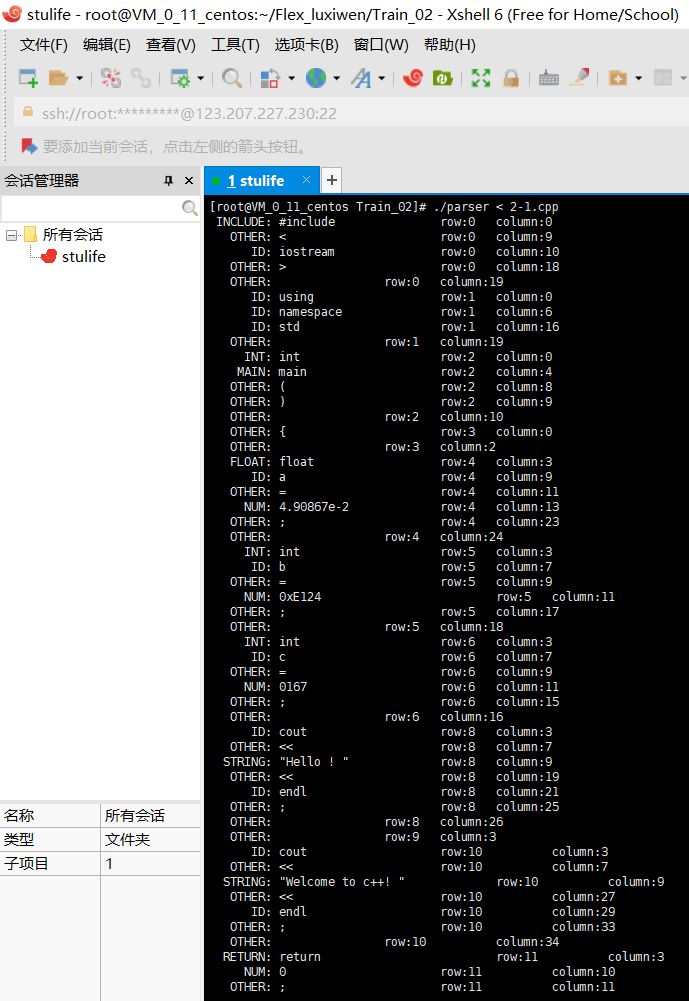


图 ‑4 Linux下lex2-2运行结果

## 实验总结

### 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。

### 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。

### 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。