**中国矿业大学计算机学院**

**2019 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 2022.2.25

学生姓名 王杰永

学 号 03190886

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 | |  | |

Flex实验一

[1 实验内容 1](#_Toc96715706)

[2 Flex环境搭建 1](#_Toc96715707)

[2.1 Windows环境下Flex的配置 1](#_Toc96715708)

[2.2 Linux环境下Flex的配置 1](#_Toc96715709)

[3 Flex源码分析 2](#_Toc96715710)

[3.1 lex1.l源码分析 2](#_Toc96715711)

[3.2 lex2.l源码分析 3](#_Toc96715712)

[4 实验结果及分析 4](#_Toc96715713)

[4.1 lex1结果分析 4](#_Toc96715714)

[4.2 lex2结果分析 5](#_Toc96715715)

[5 实验总结 5](#_Toc96715716)

* 1. 实验内容

1)阅读《Flex/Bison.pdf》第一章，第二章，掌握Flex基础知识。

2)利用Flex设计一个词法扫描器，用于统计输入文件中的字符数，单词数和行数。

* 1. Flex环境搭建
     1. Windows环境下Flex的配置

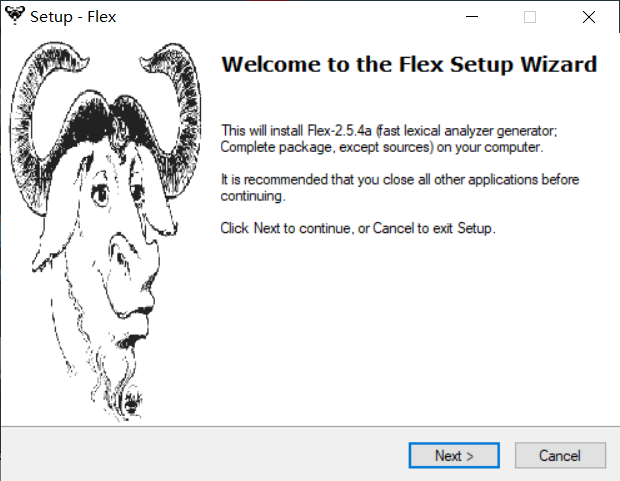
首先以管理员权限打开资料中的flex-2.5.4a-1.exe。

图1 Flex的安装

点击Next，接受许可协议，选择安装路径，即可安装成功。将安装好的flex根目录下的bin文件夹加入系统环境变量，以便在任何路径下都可以使用flex命令。

* + 1. Linux环境下Flex的配置

我选用的Linux发行版是ubuntu18.04，可以使用apt很方便的安装flex。

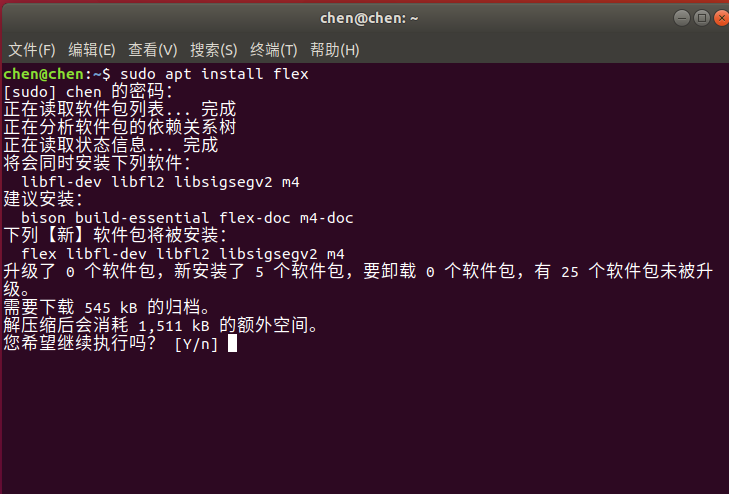
打开终端，使用命令sudo apt install flex，输入管理员密码，等待安装完成。

图2 ubuntu安装flex

* 1. Flex源码分析
     1. lex1.l源码分析

lex1要求利用Flex设计一个词法扫描器，用于统计输入文件中的字符数，单词数和行数。其源代码如下。

1. */\*word count\*/*
2. %{
3. int chars = 0;
4. int words = 0;
5. int lines = 0;
6. %}
7. %%
8. [^ \t\n]+ {words++; chars += yyleng;}
9. \n {
10. chars++;
11. lines++;
12. }
13. . {
14. chars++;
15. }
16. %%
17. int main(int argc, char \*\*argv){
18. yylex();
19. printf("%d**\t**%d**\t**%d**\n**", chars, words, lines);
20. return 0;
21. }
23. int yywrap(){
24. return 1;
25. }

第2行至第6行被 %{ 与 }% 包裹，属于全局声明，其代码会复制到生成的c代码的最前端。在这里定义的三个变量分别用于统计字符数，单词数与行数。

在第7行使用%%分割，其下的行8至行15属于词法分析模式匹配段，用于定义词法规则及匹配到特定规则后执行的动作。

行8定义了遇到非空格非换行符非制表符之外其他任意串（即单词）的动作——单词数加一，字符数增加匹配到的串的长度。

行9-12定义了遇到换行符后的动作——字符数与代码行数均加一。

行13-15定义了遇到任意字符的动作——字符数加1。由于正则匹配的原则，被上两条匹配到的串不会被该规则匹配。

最后，在行16的%%下的部分，会直接填补到生成的c文件的末尾。主函数中调用了flex词法分析函数的入口yylex()，并在分析完成后输出全局变量的值。

* + 1. lex2.l源码分析

lex2仍然是要求统计单词数，其源代码如下。

1. %{
2. int wordCount = 0;
3. %}
4. chars [A-za-z\\_\'\.\"]
5. numbers ([0-9])+
6. delim [" "\n\t]
7. whitespace {delim}+
8. words {chars}+
9. %%
10. {words} {wordCount++;}
11. {whitespace} {/\*donothing\*/}
12. {numbers} {/\*one may want to add some processing here\*/}
13. . {/\*donothing\*/}
14. %%
15. void main(){
16. yylex();
17. printf("%d**\n**", wordCount);
18. }
19. int yywrap(){
20. return 1;
21. }

行4-8定义了一些正则表达式的变量。

行4定义变量chars，指代所有大小写字母以及下划线单引号双引号和小数点之一。

行5定义变量numbers，指代任意长度大于1的数字串。

行6定义变量delim，指代空格、换行符、制表符中的一个。

行7定义变量whitespace，指代任意长度大于1的delim。

行8定义变量words，指代任意长度大于1的chars。

最后，在行10，当词法分析过程中匹配到words，则进行单词数加一的动作。

* 1. 实验结果及分析
     1. lex1结果分析

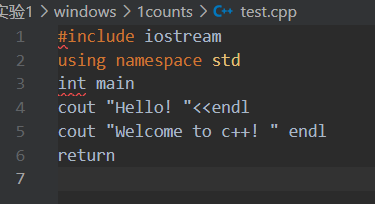
用于lex1词法分析的示例文本如下图。

图3 用于lex1词法分析的文本

从源码中可以看出，源码以空格、换行符和制表符作为单词的分隔符，因此#include会作为一个单词。在源码中的单词匹配模式处，加入如下语句

最终可以看出是哪17个单词被匹配了。

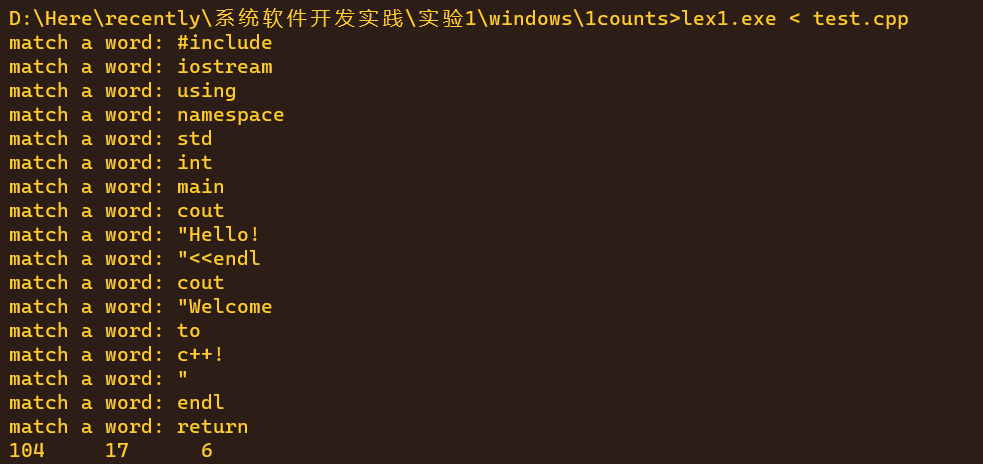
Windows与Linux系统的运行结果如图。

图4 Windows系统下lex1的运行结果

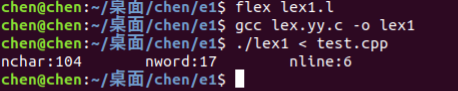


图5 Linux系统下lex1的运行结果

* + 1. lex2结果分析

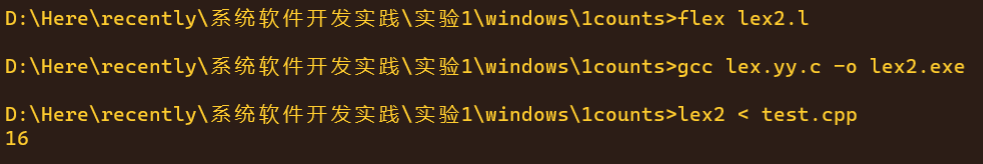
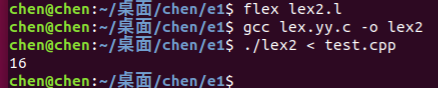
lex2源程序在Windows与Linux系统下的运行结果如下图。

图6 Windows系统下lex2的运行结果

图7 Linux系统下lex2的运行结果

* 1. 实验总结

通过本次实验，初步掌握了Flex的.l文件的基本构成，学习了各种flex允许的各种正则表达式的规则，加深了正则表达式的理解并得以更加熟练的运用正则表达式去解决实际问题。受益匪浅。