



中国矿业大学
CHINA UNIVERSITY OF MINING AND TECHNOLOGY

中国矿业大学计算机学院 2017级本科生课程设计报告

实验一 Flex理论与练习1

课程名称	系统软件开发实践
报告时间	2020.02.17
学生姓名	李治远
学 号	07172757
专 业	计算机科学与技术
任课教师	席景科老师

目录

1	实验目的	3
2	分析flex代码	3
2.1	分析lex1.1代码	3
2.2	分析lex1.2代码	4
3	分析程序输出结果	4
3.1	分析lex1.1对1-1.cpp结果	5
3.2	分析lex2.1对1-1.cpp结果	5
4	在Windows环境下实验	5
4.1	安装flex	5
4.2	安装Visual Studio 2019开发工具	7
4.3	文件保存	7
4.4	lex1.1结果	8
4.5	lex2.1结果	8
5	在ubuntu环境下实验	10
5.1	安装flex	10
5.2	文件保存	10
5.3	lex1.1结果	11
5.4	lex2.1结果	11
6	实验感想	11

实验一 Flex理论与练习1

1 实验目的

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。

2 分析flex代码

2.1 分析lex1.l代码

```
1  %{
2      int nchar,nword,nline; // 定义三个变量，分别存放字符数，单词数，行数
3  %}
4
5  %%
6  \n {nline++;nchar++;} // 匹配到回车，则行数加1，字符数加1
7  [^ \t\n] {nword++,nchar += yyleng;}
8  // 不是空格或者制表符或者换行符的1到多次，则单词数加1，字符数加上匹配到的长度
9  . {nchar++;} // 若以上两个正则未匹配到，则匹配到一个字符，字符数就加1
10 %%
11
12 void main(){
13     yylex(); // 开始开始正则分析源程序
14     printf("%d\t%d\t%d\n",nchar,nword,nline);
15 }
16
17 int yywrap(){ // 当词法分析到达结束位置时，将调用yywrap函数，
18               // 返回1表示完成分析
19     return 1;
20 }
```

整个程序包括三大部分，各部分之间通过“%%”分割。

第一部分如代码1-3行，表示声明和选项设置，并且“%{”与“%}”之内的代码会被原样的复制到生成的C文件开头。

第二部分如代码6-8行，每一行开头都是匹配模式，紧接着是匹配到所要执行的C代码，用{}包含所要执行的代码。

第三部分如代码11-18行，是程序主要的部分，负责调用flex提供的词法分析例程yylex()，并在分析结束后，返回正确的值，也可以进行声明变量值的输出。

2.2 分析lex1.2代码

```

1  %{
2      int wordCount = 0; // 记录单词数
3  %}
4
5  chars [A-Za-z\_\'\".] // 定义chars匹配字母，下划线，单引号，点，双引号
6  numbers ([0-9])+ // 匹配整数
7  delim [\"\\n\\t] // 匹配双引号，空格，换行符，制表符
8  whitespace {delim}+ // 匹配1 至多个delim 所匹配的文本
9  words {chars}+ // 匹配1 至多个chars 所匹配的文本
10
11 %%
12 {words} { wordCount++; /*increase the word count by one*/ }
13 // 匹配到单词，单词数加1
14 {whitespace} { /* donothing*/ }
15 {numbers} { /* one may want to add some processing here*/ }
16 %%
17
18 void main(){
19     yylex(); /* start theanalysis*/
20     printf(" No of words:%d\\n", wordCount); // 输出整个程序中匹配到的单词数
21 }
22
23 int yywrap(){
24     return 1;
25 }
```

3 分析程序输出结果

实验所用测试源代码1-1.cpp如下所示：

```

1  #include iostream
2  using namespace std
3  int main
4  cout "Hello! "<<endl
5  cout "Welcome to c++! " endl
6  return
```

3.1 分析lex1.l对1-1.cpp结果

初始值：nchar=0,nword=0,nline=0，分析结果如表1所示。需要注意，代码每行组成的字符串末尾都有一个换行符，所以每行的nchar都多1。

表 1: lex1.l结果分析	
程序	结果
#include iostream	nchar=+18,nword+=2,nline+=1
using namespace std	nchar=+20,nword+=3,nline+=1
int main	nchar=+9,nword+=2,nline+=1
cout <<"Hello! "<<endl	nchar=+21,nword+=3,nline+=1
cout <<"Welcome to c++! "<<endl	nchar=+29,nword+=6,nline+=1
return	nchar=+7,nword+=1,nline+=1
	nchar=104,nword=17,nline=6

3.2 分析lex2.l对1-1.cpp结果

初始值：wordCount=0

表 2: lex1.l结果分析	
程序	结果
#include iostream	wordCount=+2
using namespace std	wordCount=+3
int main	wordCount=+2
cout <<"Hello! "<<endl	wordCount=+3
cout <<"Welcome to c++! "<<endl	wordCount=+5
return	wordCount=+1
	wordCount=16

4 在Windows环境下实验

4.1 安装flex

flex安装页面如图1所示，安装结果如图2-3所示。

将C:\GnuWin32\bin添加至环境变量，这样就无需进入安装目录，就可以运行可执行文件flex.exe。



图 1: flex安装页面

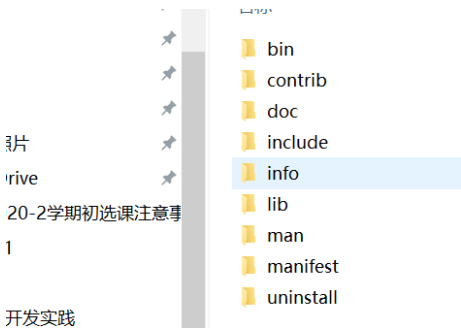


图 2: flex安装完成目录

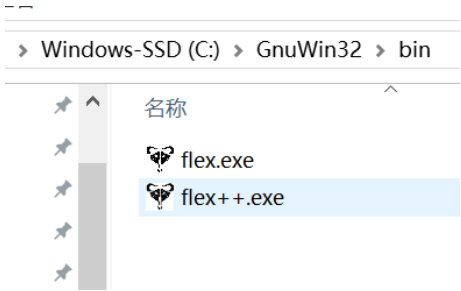


图 3: flex可执行文件

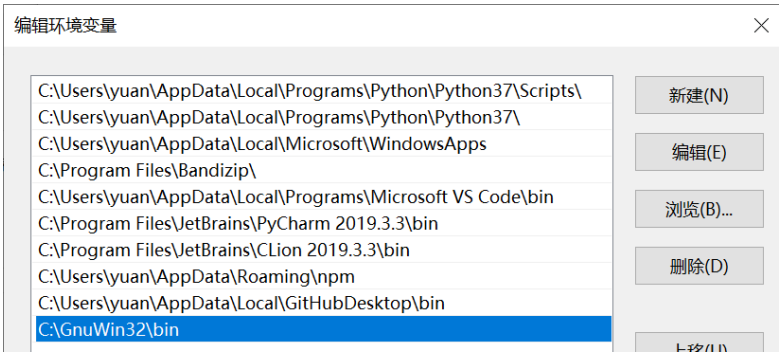


图 4: 添加环境变量

4.2 安装Visual Studio 2019开发工具

因为需要对flex生成的C源码进行编译，所以需要在电脑上安装C语言编译环境，下载Visual Studio 2019，选择C++生成工具即可，如图5所示。

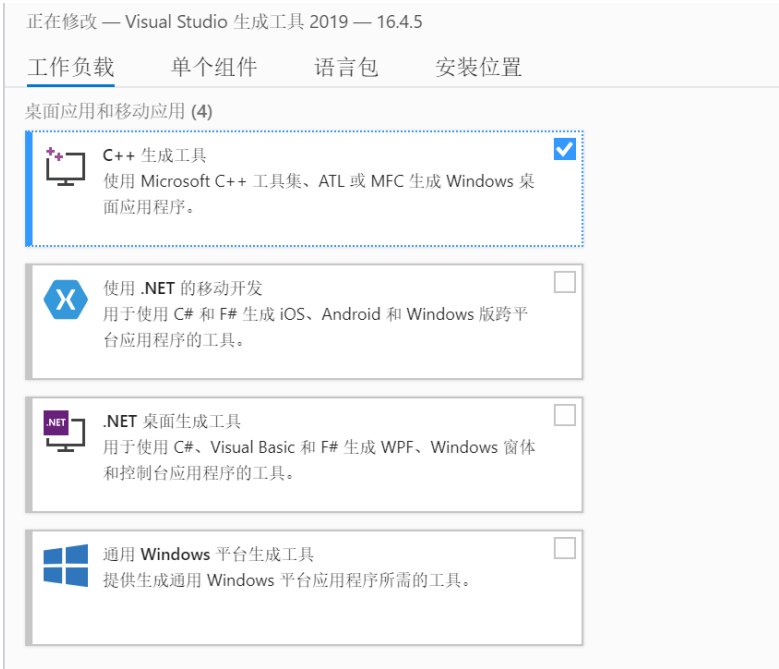


图 5: Visual Studio 2019安装

4.3 文件保存

将lex1.l与lex2.l保存在实验目录下，测试源代码保存为1-1.cpp，如图6所示。

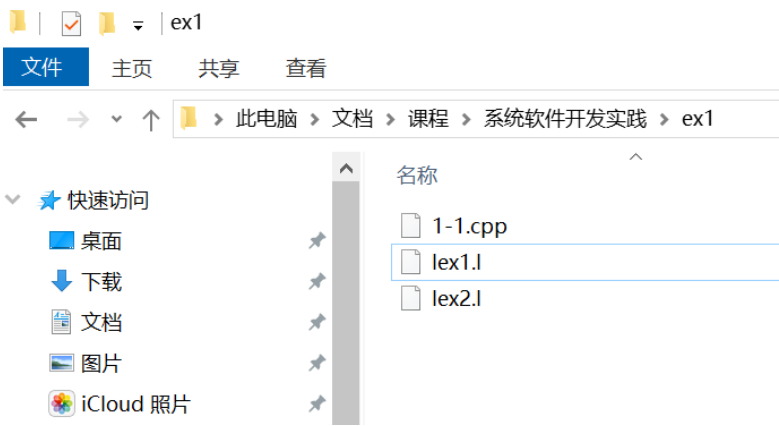


图 6: 文件目录

4.4 lex1.1结果

打开安装好的Developer Command Prompt for VS 2019，如图7所示。

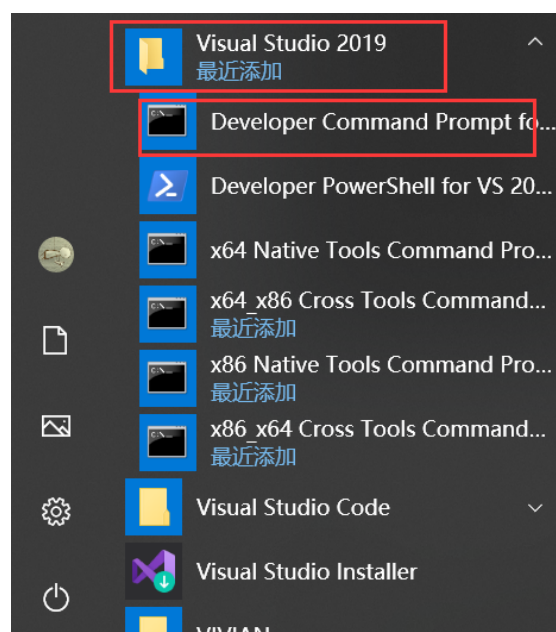


图 7: Developer Command Prompt for VS 2019

进入lex1.1文件目录：`C:\Users\yuan\Documents\课程\系统软件开发实践\ex1`。
使用命令 `flex -o"flex1.yy.c" lex1.1` 生成flex1.yy.c文件，再用cl命令将此文件编译成可执行文件，如图8-9所示。

```
C:\Program Files (x86)\Microsoft Visual Studio\2019\BuildTools>cd C:\Users\yuan\Documents\课程\系统软件开发实践\ex1

C:\Users\yuan\Documents\课程\系统软件开发实践\ex1>flex -o"flex1.yy.c" lex1.1

C:\Users\yuan\Documents\课程\系统软件开发实践\ex1>cl flex1.yy.c
用于 x86 的 Microsoft (R) C/C++ 优化编译器 19.24.28316 版
版权所有 (C) Microsoft Corporation。保留所有权利。

flex1.yy.c
Microsoft (R) Incremental Linker Version 14.24.28316.0
Copyright (C) Microsoft Corporation. All rights reserved.

/out:flex1.yy.exe
flex1.yy.obj
```

图 8: 编译结果

之后再使用生成的flex1.yy.exe文件对1-1.cpp文件进行分析测试，得出结果，如图10所示。

4.5 lex2.1结果

同4.4方法，对lex2.1进行flex生成c源程序，并进行编译，结果如图11-12所示。

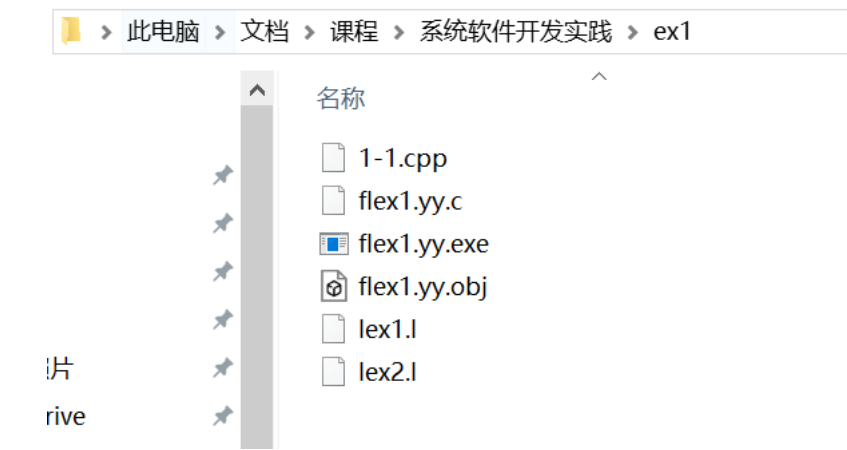


图 9: 生成文件结果

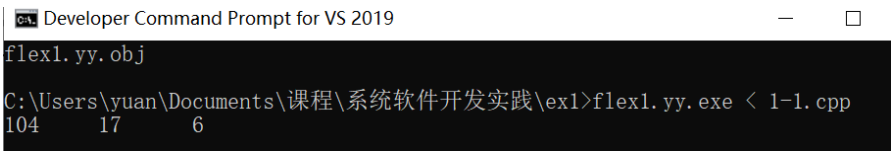


图 10: lex1.l结果



图 11: 编译结果

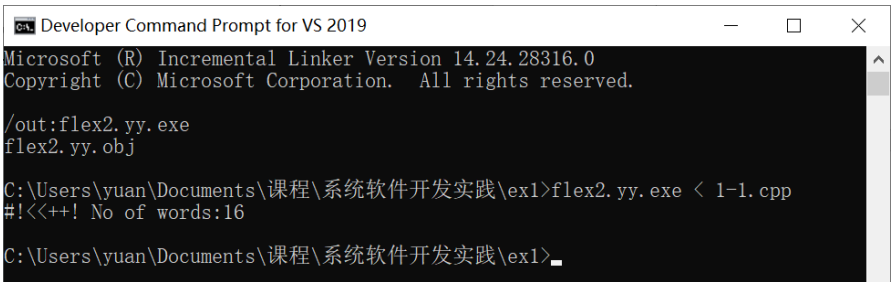
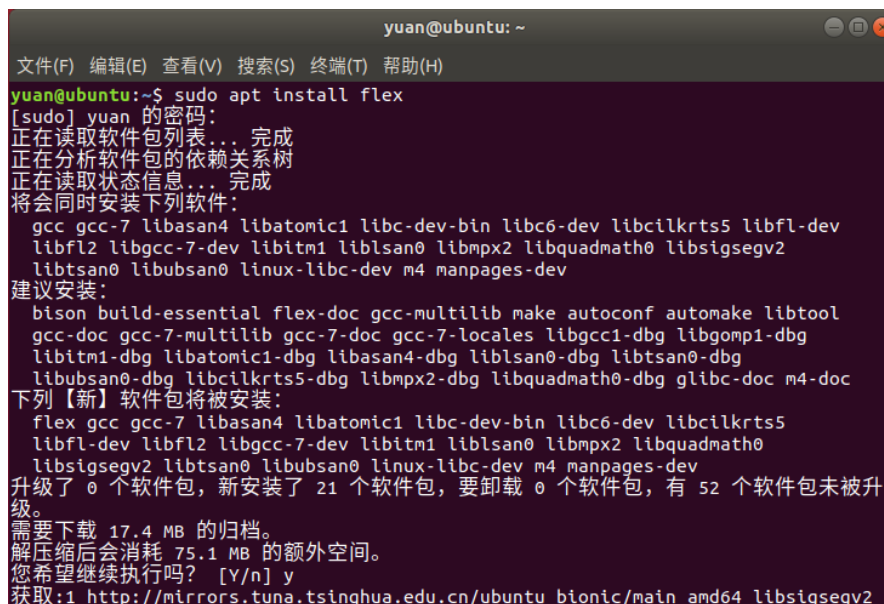


图 12: lex2.l结果

5 在ubuntu环境下实验

5.1 安装flex

打开终端，输入`sudo apt install flex`，输入用户密码后，再输入`y`就可以正确安装，如图13所示。



```
yuan@ubuntu: ~  
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)  
yuan@ubuntu:~$ sudo apt install flex  
[sudo] yuan 的密码:  
正在读取软件包列表... 完成  
正在分析软件包的依赖关系树  
正在读取状态信息... 完成  
将会同时安装下列软件:  
gcc gcc-7 libasan4 libatomic1 libc-dev-bin libc6-dev libcilkrts5 libfl-dev  
libfl2 libgcc-7-dev libitm1 liblsan0 libmpx2 libquadmath0 libsigsegv2  
libtsan0 libubsan0 linux-libc-dev m4 manpages-dev  
建议安装:  
bison build-essential flex-doc gcc-multilib make autoconf automake libtool  
gcc-doc gcc-7-multilib gcc-7-doc gcc-7-locales libgcc1-dbg libgomp1-dbg  
libitm1-dbg libatomic1-dbg libasan4-dbg liblsan0-dbg libtsan0-dbg  
libubsan0-dbg libcilkrts5-dbg libmpx2-dbg libquadmath0-dbg glibc-doc m4-doc  
下列【新】软件包将被安装:  
flex gcc gcc-7 libasan4 libatomic1 libc-dev-bin libc6-dev libcilkrts5  
libfl-dev libfl2 libgcc-7-dev libitm1 liblsan0 libmpx2 libquadmath0  
libsigsegv2 libtsan0 libubsan0 linux-libc-dev m4 manpages-dev  
升级了 0 个软件包，新安装了 21 个软件包，要卸载 0 个软件包，有 52 个软件包未被升  
级。  
需要下载 17.4 MB 的归档。  
解压缩后会消耗 75.1 MB 的额外空间。  
您希望继续执行吗？ [Y/n] y  
获取:1 http://mirrors.tuna.tsinghua.edu.cn/ubuntu bionic/main amd64 libsigsegv2
```

图 13: ubuntu安装flex

5.2 文件保存

将实验所需文件保存在`/home/yuan/桌面/System Software/ex1`目录下，如图14所示。

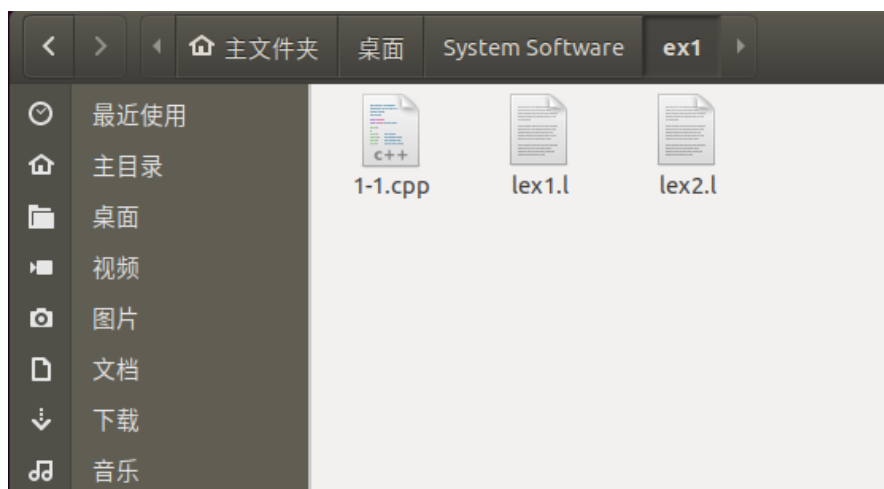
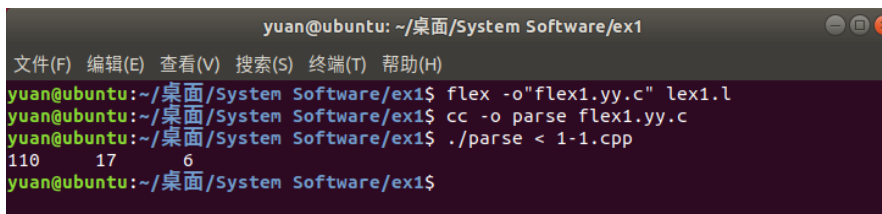


图 14: 实验文件存储

5.3 lex1.l结果

在/home/yuan/桌面/System Software/ex1文件夹中打开终端，输入flex -o "flex1.yy.c" lex1.l，生成C源程序，再输入cc -o parse flex1.yy.c对flex生成的C源程序进行编译，最后输入./parse < 1-1.cpp，对1-1.cpp源程序进行分析，结果如图15所示。

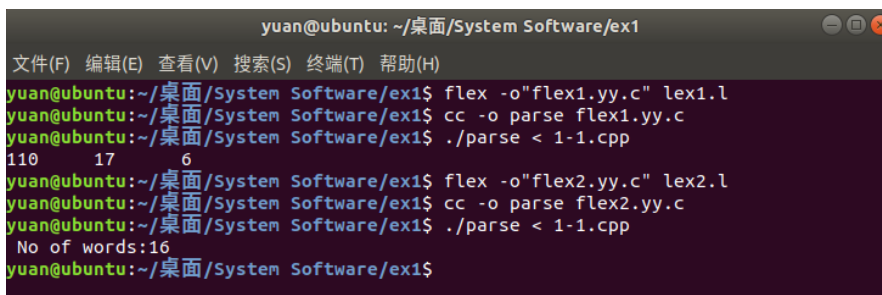


```
yuan@ubuntu: ~/桌面/System Software/ex1
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
yuan@ubuntu:~/桌面/System Software/ex1$ flex -o"flex1.yy.c" lex1.l
yuan@ubuntu:~/桌面/System Software/ex1$ cc -o parse flex1.yy.c
yuan@ubuntu:~/桌面/System Software/ex1$ ./parse < 1-1.cpp
110      17      6
yuan@ubuntu:~/桌面/System Software/ex1$
```

图 15: lex1.l实验结果

5.4 lex2.l结果

同5.3的方法，对lex2.l进行实验，结果如图16所示。



```
yuan@ubuntu: ~/桌面/System Software/ex1
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
yuan@ubuntu:~/桌面/System Software/ex1$ flex -o"flex1.yy.c" lex1.l
yuan@ubuntu:~/桌面/System Software/ex1$ cc -o parse flex1.yy.c
yuan@ubuntu:~/桌面/System Software/ex1$ ./parse < 1-1.cpp
110      17      6
yuan@ubuntu:~/桌面/System Software/ex1$ flex -o"flex2.yy.c" lex2.l
yuan@ubuntu:~/桌面/System Software/ex1$ cc -o parse flex2.yy.c
yuan@ubuntu:~/桌面/System Software/ex1$ ./parse < 1-1.cpp
No of words:16
yuan@ubuntu:~/桌面/System Software/ex1$
```

图 16: lex2.cpp实验结果

6 实验感想

经过实验一的学习与操作，我掌握了flex基本使用方法，也明白了flex分析程序的整体结构，对比上学期的编译实验，当时只是听老师说过flex，在实验中也并没有使用flex，虽然实现了需要的功能，但是方法繁琐，也易错，但是flex并不会这样，它通过C语言与正则匹配的方法，简单清晰的完成了源代码的词法分析或者字符统计操作。

但是，在实验中也遇到了一些问题，比如说对正则匹配的掌握不够，这需要加强练习。也发现了window与linux环境中的一些差别，比如lex1.l在两种环境下统计字符数不同的结果，原因是win中回车为“\r\n”两个字符组成，linux中为“\n”一个字符组成，所以经过6行程序的匹配，造成了字符数差6的结果。