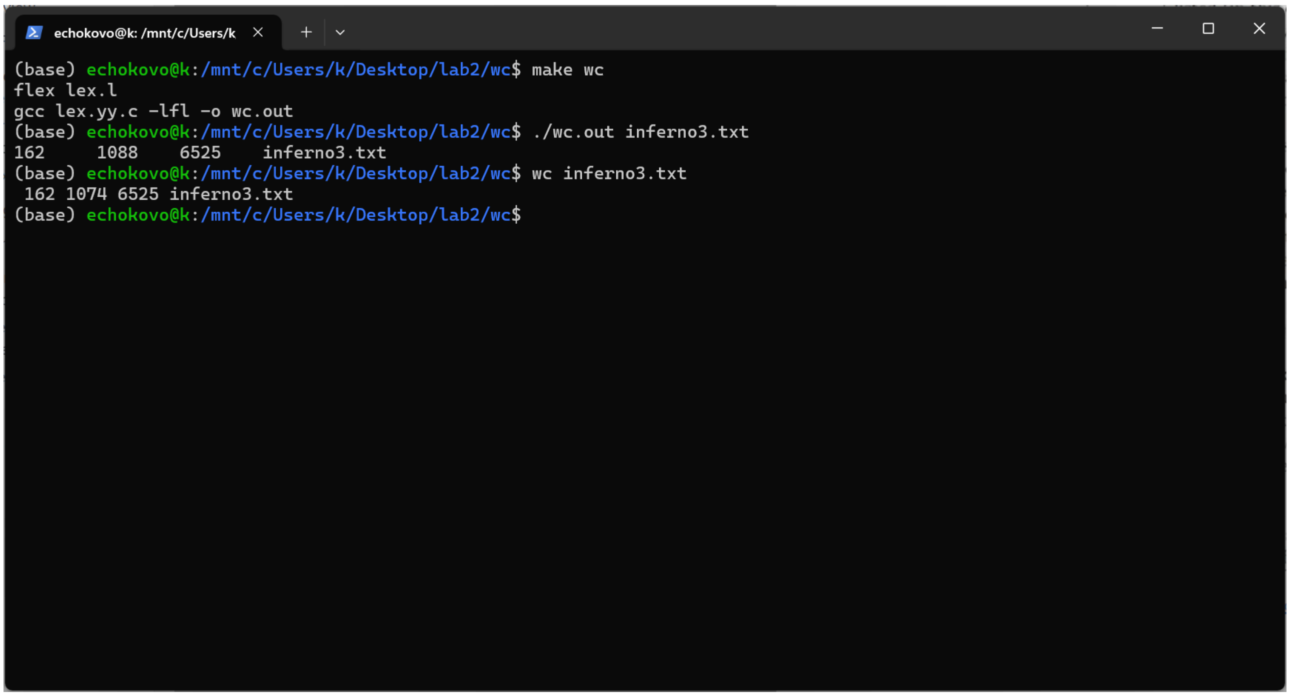
实验报告

学号：

班级：

1. wc

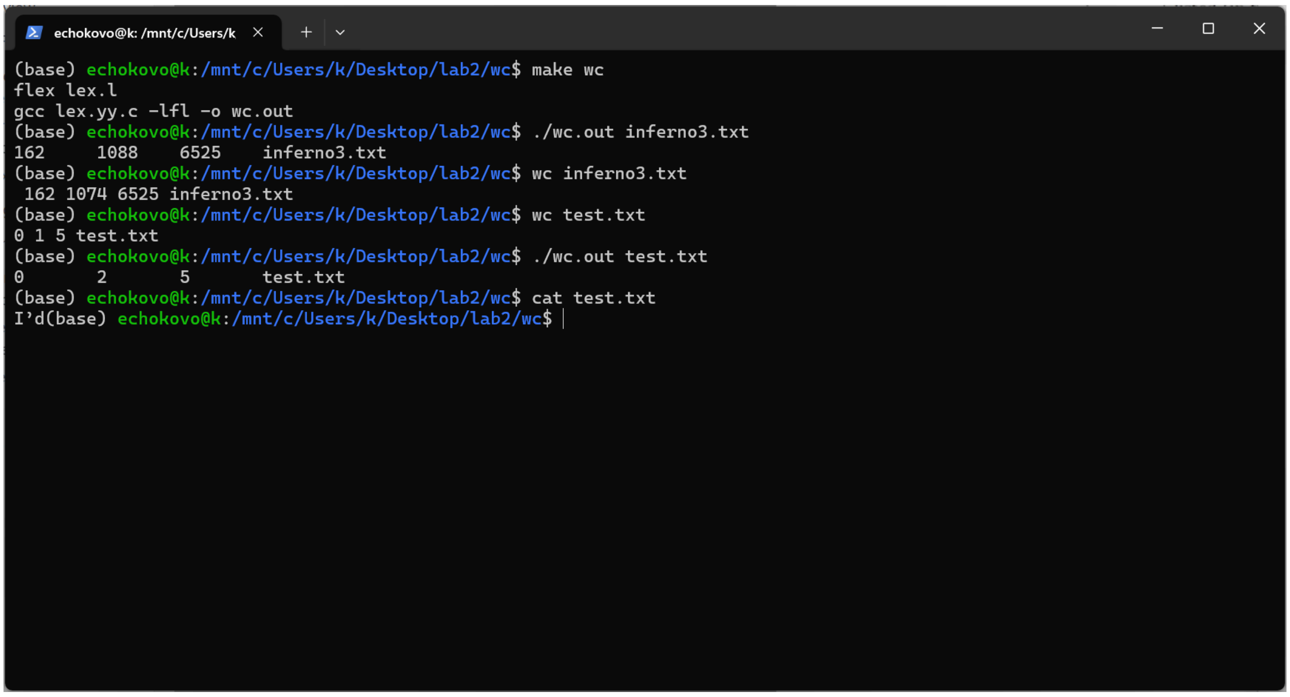
1. 安装 flex 后运行截图如下



```
(base) echokovo@k:/mnt/c/Users/k/Desktop/lab2/wc$ make wc
flex lex.l
gcc lex.yy.c -lfl -o wc.out
(base) echokovo@k:/mnt/c/Users/k/Desktop/lab2/wc$ ./wc.out inferno3.txt
162     1088    6525    inferno3.txt
(base) echokovo@k:/mnt/c/Users/k/Desktop/lab2/wc$ wc inferno3.txt
 162 1074 6525 inferno3.txt
(base) echokovo@k:/mnt/c/Users/k/Desktop/lab2/wc$
```

2. 该 wc 程序对单词的定义是连续的大小写字母 故 `I'd` 会被识别为两个单词 而 Linux 系统则会将其识别为一个单词

   通过 `test.txt` 的测试可证实以上内容

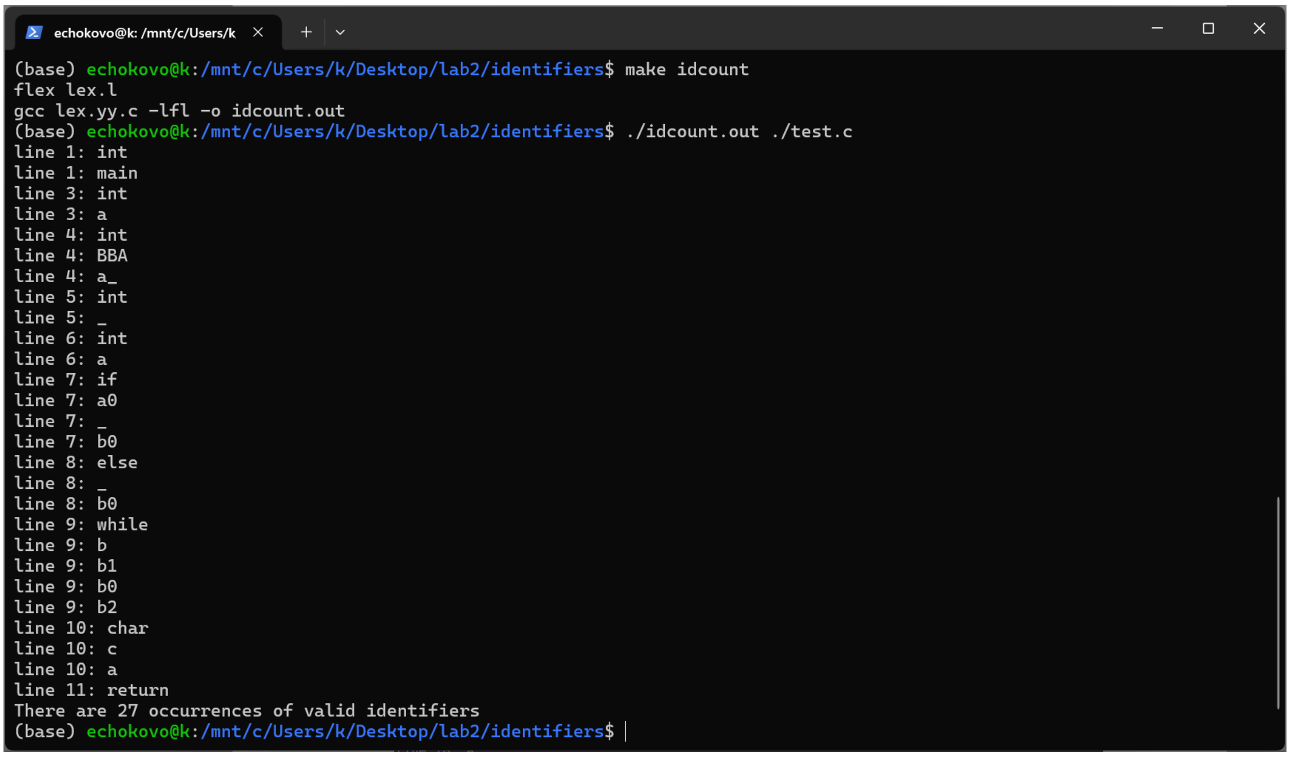


```
(base) echokovo@k:/mnt/c/Users/k/Desktop/lab2/wc$ make wc
flex lex.l
gcc lex.yy.c -lfl -o wc.out
(base) echokovo@k:/mnt/c/Users/k/Desktop/lab2/wc$ ./wc.out inferno3.txt
162     1088    6525    inferno3.txt
(base) echokovo@k:/mnt/c/Users/k/Desktop/lab2/wc$ wc inferno3.txt
 162 1074 6525 inferno3.txt
(base) echokovo@k:/mnt/c/Users/k/Desktop/lab2/wc$ wc test.txt
0 1 5 test.txt
(base) echokovo@k:/mnt/c/Users/k/Desktop/lab2/wc$ ./wc.out test.txt
0       2       5       test.txt
(base) echokovo@k:/mnt/c/Users/k/Desktop/lab2/wc$ cat test.txt
I’d(base) echokovo@k:/mnt/c/Users/k/Desktop/lab2/wc$
```

   故 wc 程序统计的单词数量会比 Linux 系统统计的更多

2. identifiers

1. 修改 `lex.l` 后运行截图如下



```
(base) echokovo@k:/mnt/c/Users/k/Desktop/lab2/identifiers$ make idcount
flex lex.l
gcc lex.yy.c -lfl -o idcount.out
(base) echokovo@k:/mnt/c/Users/k/Desktop/lab2/identifiers$ ./idcount.out ./test.c
line 1: int
line 1: main
line 3: int
line 3: a
line 4: int
line 4: BBA
line 4: a_
line 5: int
line 5: _
line 6: int
line 6: a
line 7: if
line 7: a0
line 7: _
line 7: b0
line 8: else
line 8: _
line 8: b0
line 9: while
line 9: b
line 9: b1
line 9: b0
line 9: b2
line 10: char
line 10: c
line 10: a
line 11: return
There are 27 occurrences of valid identifiers
(base) echokovo@k:/mnt/c/Users/k/Desktop/lab2/identifiers$
```

2. 通过提交遇到换行符 `\n` 时的逻辑 将变量 `lines` 进行自增

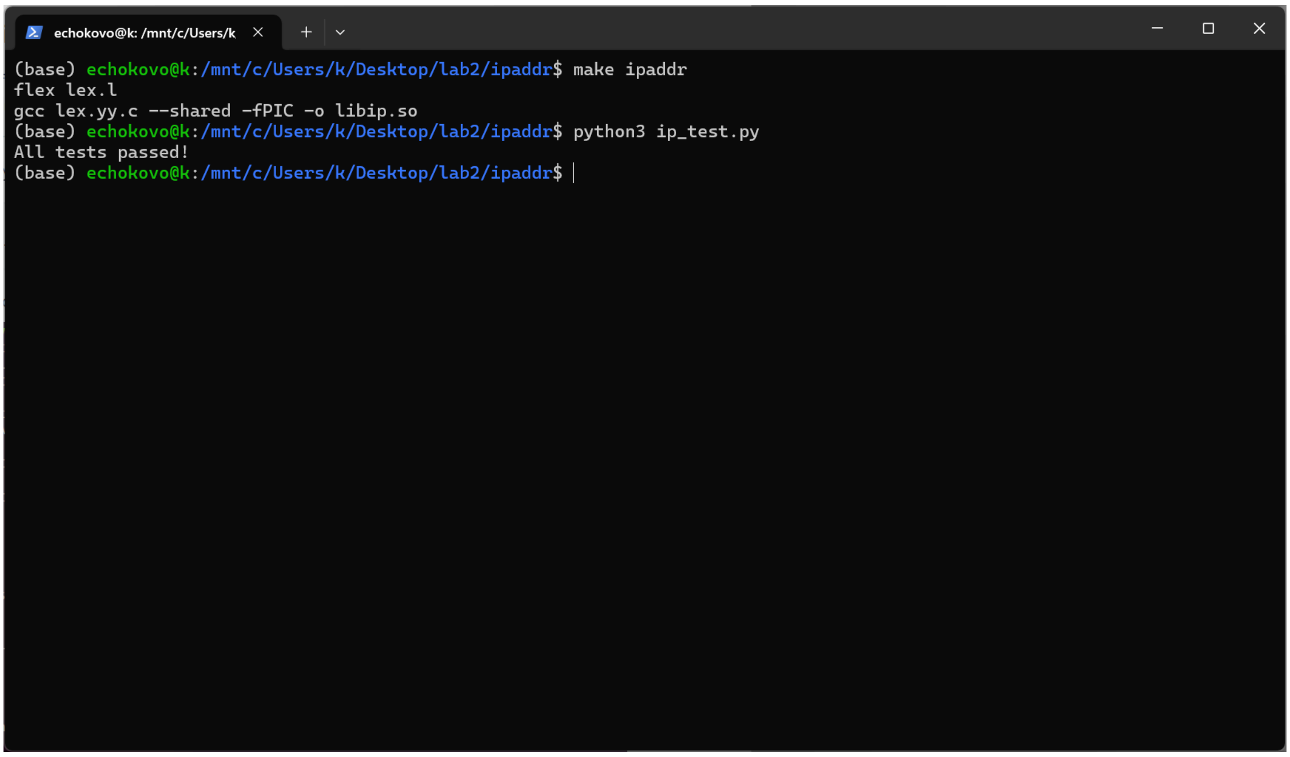
```
%%
{letter_}({letter_}|{digit})* { identifiers++; printf("line %d: %s\n", lines, yytext);}
\n {lines++; }
[\t\r ]+ { /* does nothing when seeing white spaces except new line */ }
. { /* a final rule that matches when seeing any character but new line */ }
<<EOF>> { printf("There are %d occurrences of valid identifiers\n", identifiers); yyterminate(); }

%%
```

同时将变量 `lines` 的初始值修改为 `1`

3. ipaddr

1. 修改 `lex.l` 后运行截图如下



echokovo@k: /mnt/c/Users/k

```
(base) echokovo@k:/mnt/c/Users/k/Desktop/lab2/ipaddr$ make ipaddr
flex lex.l
gcc lex.yy.c --shared -fPIC -o libip.so
(base) echokovo@k:/mnt/c/Users/k/Desktop/lab2/ipaddr$ python3 ip_test.py
All tests passed!
(base) echokovo@k:/mnt/c/Users/k/Desktop/lab2/ipaddr$ 
```

2. 对于 ipv4
   其被逗号分隔开的四个数字的范围为 0-255 故直接定义 digit_v4 为 0-255 的数字 即可识别 ipv4
   对于 ipv6
   其被冒号分隔开的八个部分均为 0-9 a-f A-F 故直接定义 digit_v6 为 [0-9a-fA-F] 即可识别 ipv6

```
%option noyywrap

digit_v4 ([0-9])|([1-9][0-9])|(1[0-9]{2})|(2[0-4][0-9])|(25[0-5])
digit_v6 [0-9a-fA-F]
v4 ^({digit_v4}"."){3}{digit_v4}$
v6 ^({digit_v6}{1,4}":"){7}{digit_v6}{1,4}$

%%
```