**中国矿业大学计算机学院**

**2018 级本科生课程设计报告**

课程名称 系统软件开发实践

实验名称 Flex理论与练习实验2

报告时间 2021年3月6日

学生姓名 丁旭行

学 号 14184501

专 业 计算机科学与技术

任课教师 徐东红

**目 录**

[1 实验目的 1](#_Toc65968816)

[2 实验要求 1](#_Toc65968817)

[3 Lex程序源代码分析 1](#_Toc65968818)

[3.1 Lex程序功能描述 1](#_Toc65968819)

[3.2 Lex程序代码分析 2](#_Toc65968820)

[4 Windows环境下lex.l源代码编写与执行 3](#_Toc65968821)

[4.1 lex.l源代码编写与保存 3](#_Toc65968822)

[4.2 Visual Studio 2019命令行执行命令 4](#_Toc65968823)

[4.3 C语言代码编写与测试 7](#_Toc65968824)

[4.4 分析程序输出结果 9](#_Toc65968825)

[5 Linux环境下lex.l源代码编写与执行 12](#_Toc65968826)

[5.1 lex.l源代码编写与保存 12](#_Toc65968827)

[5.2 C语言代码编写与保存 14](#_Toc65968828)

[5.3 lex代码运行测试 14](#_Toc65968829)

[5.4 分析程序输出结果 17](#_Toc65968830)

[6 实验总结 17](#_Toc65968831)

# 1 实验目的

1. 编写LEX源文件，实现C语言子集的词法分析功能，最后上机调试。
2. 要求编写一个测试程序，以给定的测试文件作为输入，输出运行结果到输出文件中。

# 2 实验要求

给定C语言的一个子集，具体内容如下：

1. 下面是语言的关键字：

else if switch for int float return void while所有的关键字都是保留字，并且必须是小写。

1. 下面是专用符号：

+ - \* / < <= > >= == != = ; , ( ) [ ] { } /\* \*/

1. 其他标记是标识符(ID)和数字(NU），通过下列正则表达式定义：

ID = letter letter\*

NUM = digit digit\*

letter = a|..|z|A|..|Z

digit = 0|..|9

注：小写和大写字母是有区别的。

1. 空格由空白、换行符和制表符组成。空格通常被忽略，除了它必须分开ID、NUM关键字。
2. 注释用通常的C语言符号/\* . . . \*/围起来。注释可以放在任何空白出现的位置(即注释不能放在标记内)上，且可以超过一行。注释不能嵌套。

# 3 Lex程序源代码分析

## 3.1 Lex程序功能描述

此Lex程序实现了C语言子集的词法分析功能，并将分析结果输入到结果文件result.txt中。

## 3.2 Lex程序代码分析



图1 定义段和规则段代码分析



图2 用户代码段代码分析

# 4 Windows环境下lex.l源代码编写与执行

由于在实验一中已经将Flex安装配置完毕，其具体过程在此就不再赘述。

## 4.1 lex.l源代码编写与保存

新建文本文档编写lex.l源代码，编写完后，将其保存到Flex的安装目录下。

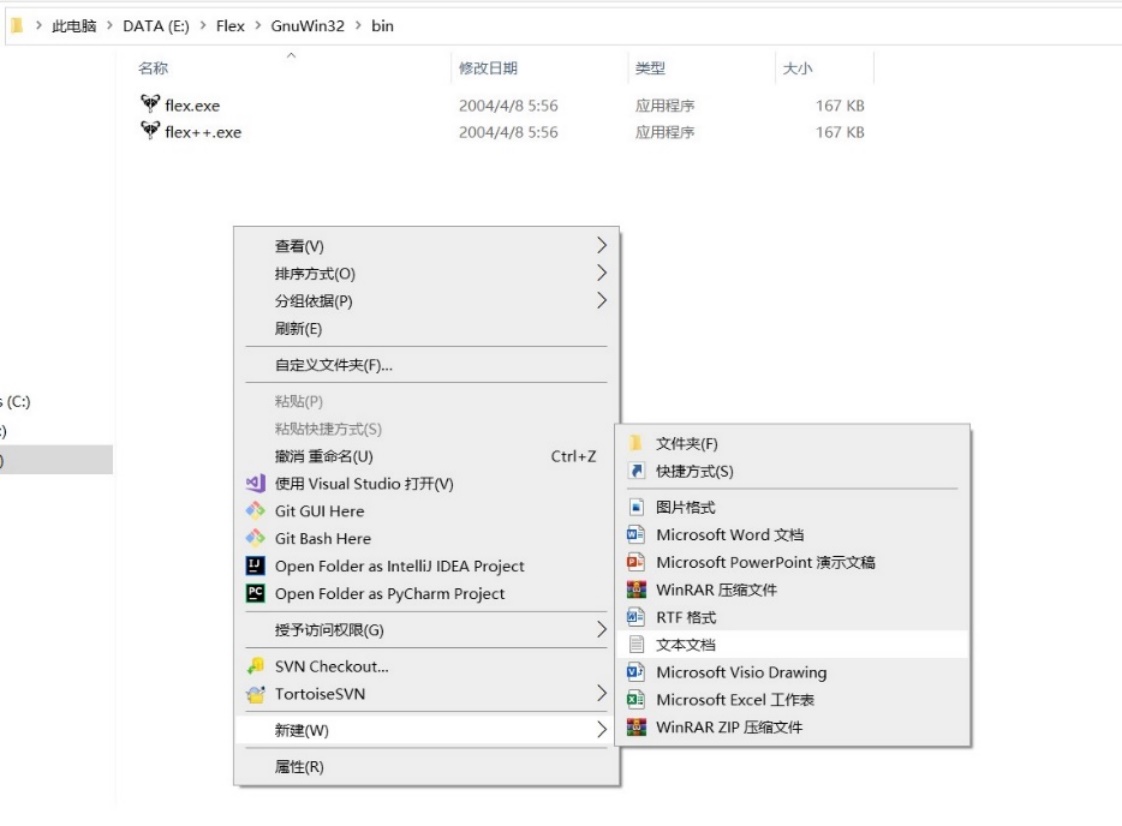


图3 新建文本文档

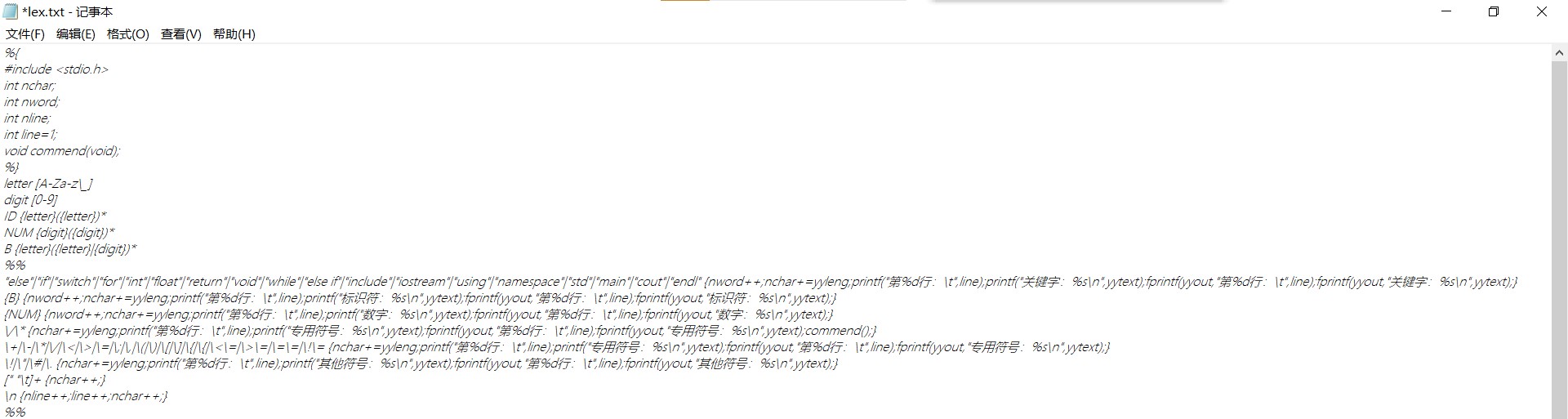


图4 lex.l源代码 1

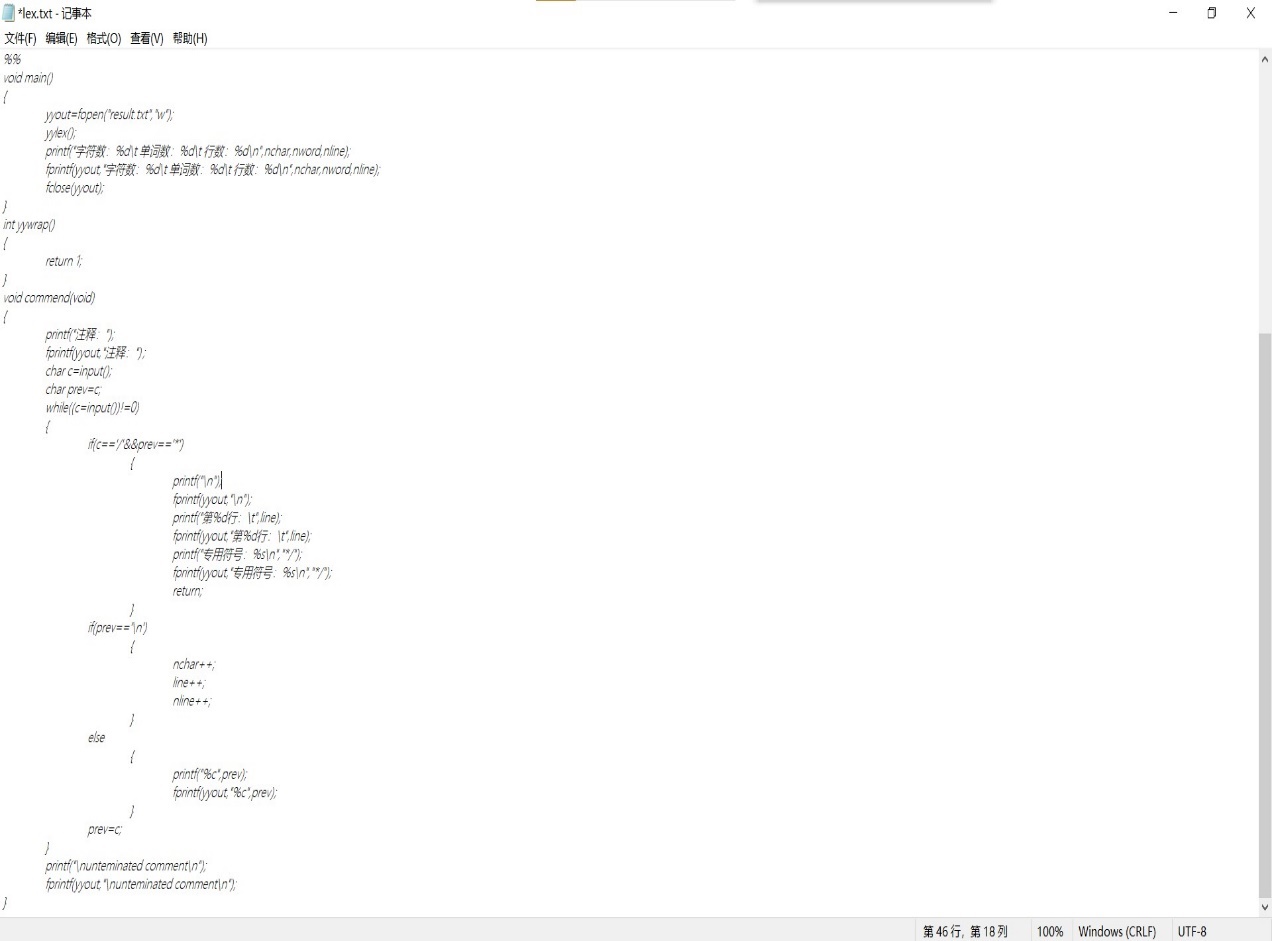


图5 lex.l源代码 2

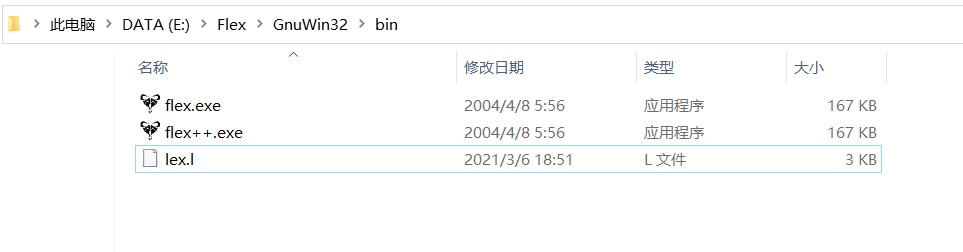


图6 lex.l保存目录

## 4.2 Visual Studio 2019命令行执行命令

打开Developer Command Prompt for VS 2019命令窗口。

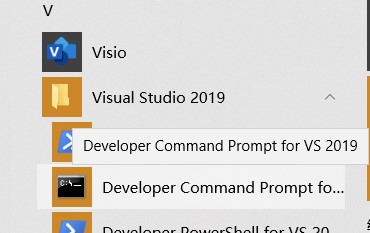


图7 打开命令窗口

依次输入e:、cd E:\Flex\GnuWin32\bin命令进入Flex安装目录下。

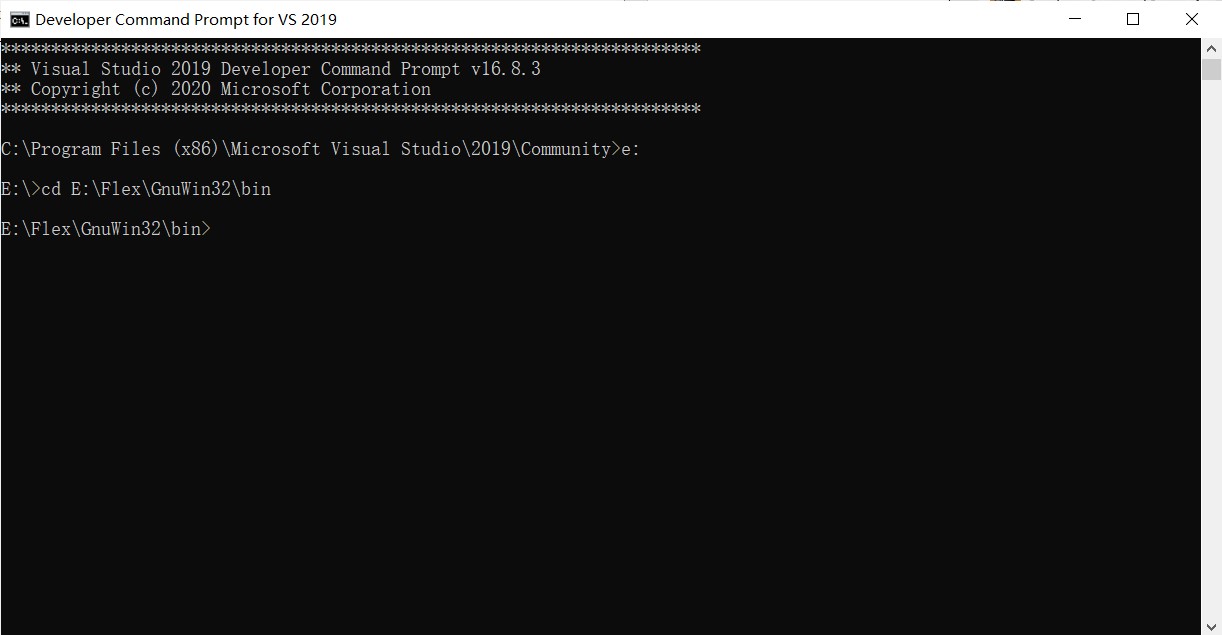


图8 进入Flex安装目录

再输入flex.exe -o “lex.yy.c” lex.l(调用flex.exe)生成lex.yy.c文件。

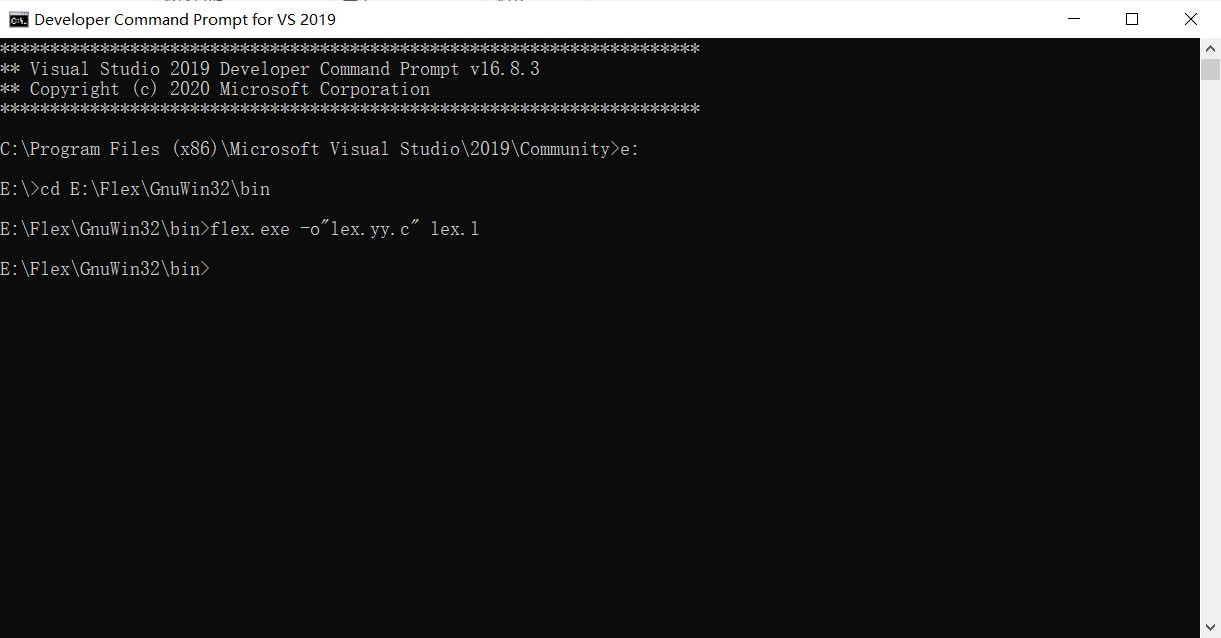


图9 生成lex.yy.c输入的命令

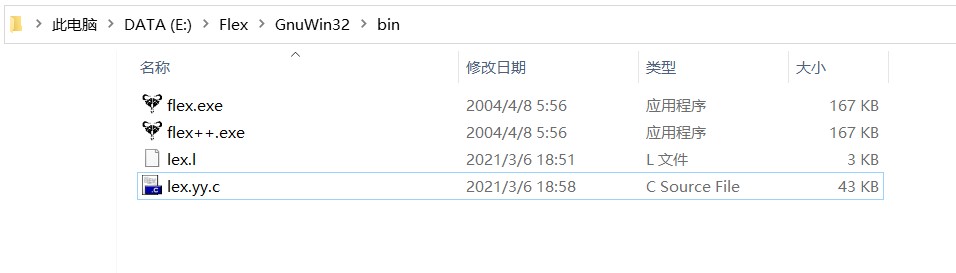


图10 生成lex.yy.c文件所在目录

打开Cygwin命令窗口，进入lex.yy.c所在目录，输入命令gcc lex.yy.c生成其对应的a.exe文件。（Cygwin安装过程https://www.jianshu.com/p/b940cc575e57）



图11 打开Cygwin命令窗口

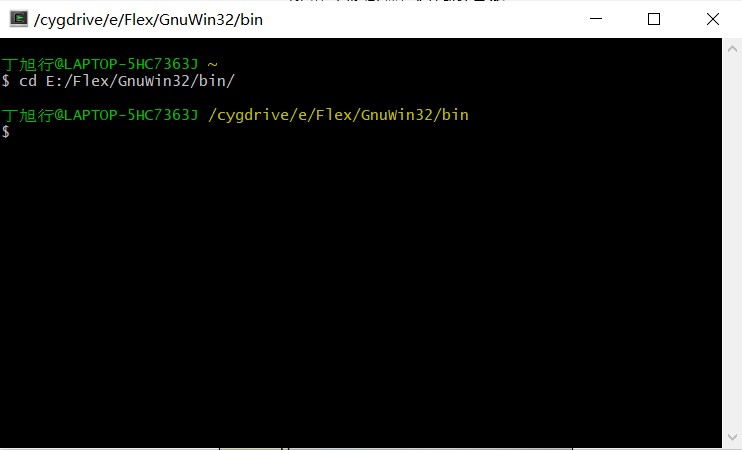


图12 进入lex.yy.c所在目录



图13 输入命令

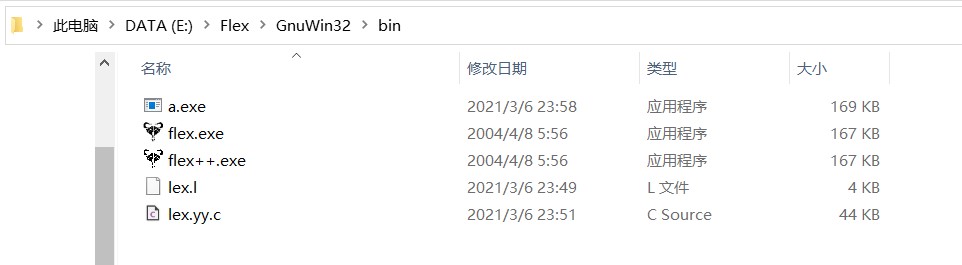


图14 生成a.exe文件

## 4.3 C语言代码编写与测试

在Flex安装目录下新建cpp文件，编写测试代码1.cpp。在命令行中输入a.exe < 1.cpp调用a.exe对1.cpp文件进行词法分析并显示分析结果。

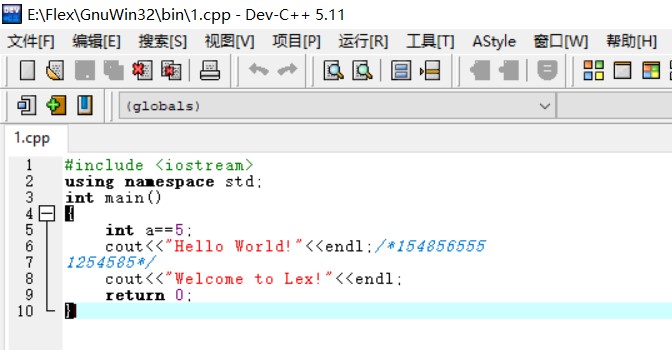
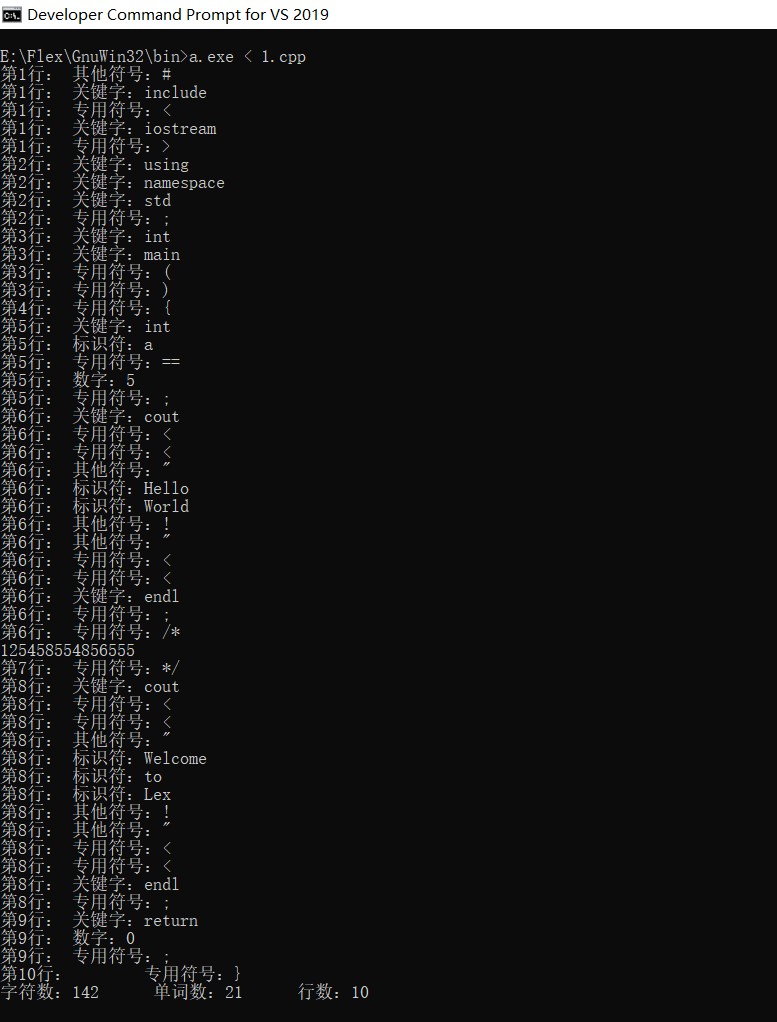


图15 C语言代码文件1.cpp



图表 16 调用a.exe分析结果

可以在当前目录下看到生成的result.txt结果输出文件，打开result.txt文件可以看到输出结果。

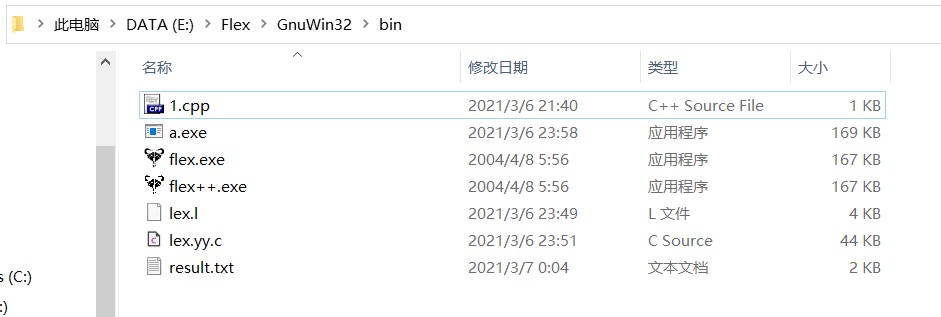


图17 生成result.txt文件

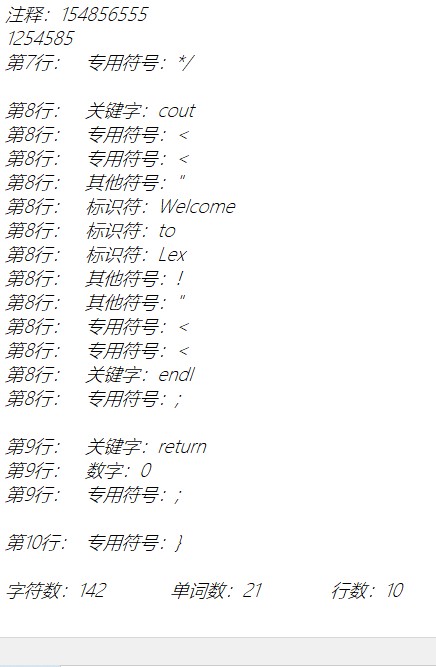
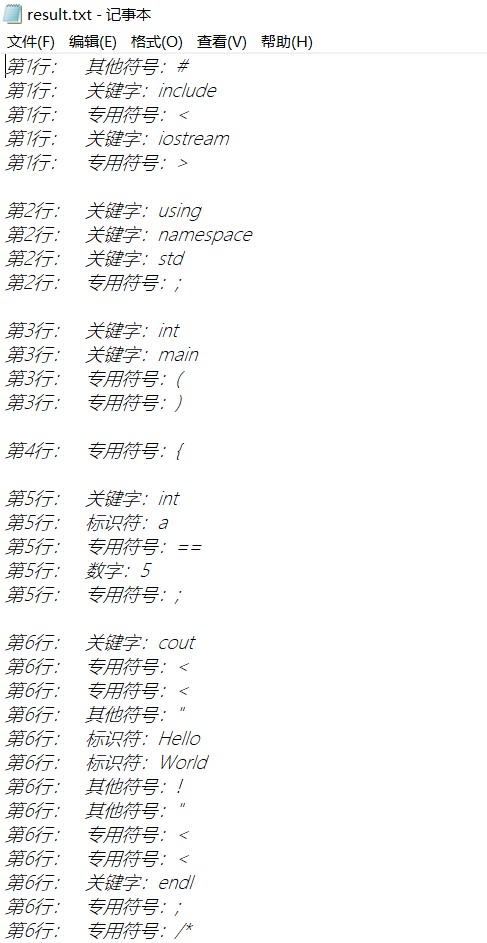


图18 文件输出结果

## 4.4 分析程序输出结果

第1行，读入#，识别为其他符号，按要求输出，单词数为0，行数为0，字符数为1，当前行数为1。

第1行，读入include，识别为关键字，按要求输出，此时单词数为1，行数为0，字符数为8, 当前行数为1，接着读入空格，字符数为9。

第1行，读入<，识别为专用字符，按要求输出，此时单词数为1，行数为0，字符数为10, 当前行数为1。

第1行，读入iostream，识别为关键字，按要求输出，此时单词数为2，行数为0，字符数为18，当前行数为1。

第1行，读入>，识别为专用字符，按要求输出，此时单词数为2，行数为0，字符数为19, 当前行数为1。接着读入\n，行数为1，字符数为20，当前行数为2。

第2行，读入using，识别为关键字，按要求输出，此时单词数为3，行数为1，字符数为25，接着读入空格，字符数为26，当前行数为2。

第2行，读入namespace，识别为关键字，按要求输出，此时单词数为4，行数为1，字符数为35，接着读入空格，字符数为36，当前行数为2。

第2行，读入std，识别为关键字，按要求输出，此时单词数为5，行数为1，字符数为39，当前行数为2。

第2行，读入;，识别为专用字符，按要求输出，此时单词数为5，行数为1，字符数为40。接着读入\n，字符数为41，行数为2，当前行数为3。

第3行，读入int，识别为关键字，按要求输出，此时单词数为6，行数为2，字符数为44，接着读入空格，字符数为45，当前行数为3。

第3行，读入main，识别为关键字，按要求输出，此时单词数为7，行数为2，字符数为49，当前行数为3。

第3行，读入（，识别为专用字符，按要求输出，此时单词数为7，行数为2，字符数为50，当前行数为3。

第3行，读入），识别为专用字符，按要求输出，此时单词数为7，行数为2，字符数为51，当前行数为3。接着读入\n，行数为3，字符数为52，当前行数为4。

第4行，读入{，识别为专用字符，按要求输出，此时单词数为7，行数为3，字符数为53，接着读入\n，字符数为54，行数为4，当前行数为5。

第5行，读入空格，字符数为55，行数为4，当前行数为5。

第5行，读入int，识别为关键字，按要求输出，此时单词数为8，行数为4，字符数为58，接着读入空格，字符数为59，行数为4，当前行数为5。

第5行，读入a，识别为标识符，按要求输出，此时单词数为9，行数为4，字符数为60，当前行数为5。

第5行，读入==，识别为专用符号，按要求输出，此时单词数为9，行数为4，字符数为62，当前行数为5。

第5行，读入5，识别为数字，按要求输出，此时单词数为10，行数为4，字符数为63，当前行数为5。

第5行，读入;，识别为专用符号，按要求输出，此时单词数为10，行数为4，字符数为64，接着读入\n，字符数为65，行数为5，当前行数为6。

第6行，读入空格，字符数为66，行数为5，当前行数为6。

第6行，读入cout，识别为关键字，按要求输出，此时单词数为11，行数为5，字符数为70，当前行数为6。

第6行，读入<，识别为专用符号，按要求输出，此时单词数为11，行数为5，字符数为71，当前行数为6。

第6行，读入<，识别为专用符号，按要求输出，此时单词数为11，行数为5，字符数为72，当前行数为6。

第6行，读入”，识别为其他字符，按要求输出，此时单词数为11，行数为5，字符数为73，当前行数为6。

第6行，读入Hello，识别为标识符，按要求输出，此时单词数为12，行数为5，字符数为78，当前行数为6。

第6行，读入空格，字符数为79，行数为5，当前行数为6。

第6行，读入World，识别为标识符，按要求输出，此时单词数为13，行数为5，字符数为84，当前行数为6。

第6行，读入！，识别为其他字符，按要求输出，此时单词数为13，行数为5，字符数为85，当前行数为6。

第6行，读入”，识别为其他字符，按要求输出，此时单词数为13，行数为5，字符数为86，当前行数为6。

第6行，读入<，识别为专用符号，按要求输出，此时单词数为13，行数为5，字符数为87，当前行数为6。

第6行，读入<，识别为专用符号，按要求输出，此时单词数为13，行数为5，字符数为88，当前行数为6。

第6行，读入endl，识别为标识符，按要求输出，此时单词数为14，行数为5，字符数为92，当前行数为6。

第6行，读入;，识别为专用符号，按要求输出，此时单词数为14，行数为5，字符数为93，当前行数为6。

第6行，读入/\*，按要求输出，此时单词数为14，行数为5，字符数为95，当前行数为6。进入comment函数，输出注释，在注释中读入\n，字符数为96，行数为6，当前行数为7。

第7行，读入\*/，为注释结束符，按要求输出，此时单词数为14，行数为6，字符数为96，行数为6，当前行数为7。接着读入\n，字符数为97，行数为7，当前行数为8。

第8行，读入空格，字符数为98，行数为7，当前行数为8。

第8行，读入cout，识别为关键字，按要求输出，此时单词数为15，行数为7，字符数为102，当前行数为8。

第8行，读入<，识别为专用符号，按要求输出，此时单词数为15，行数为7，字符数为103，当前行数为8。

第8行，读入<，识别为专用符号，按要求输出，此时单词数为15，行数为7，字符数为104，当前行数为8。

第8行，读入”，识别为其他字符，按要求输出，此时单词数为15，行数为7，字符数为105，当前行数为8。

第8行，读入Welcome，识别为标识符，按要求输出，此时单词数为16，行数为7，字符数为112，接着读入空格，字符数为113，行数为7，当前行数为8。

第8行，读入to，识别为标识符，按要求输出，此时单词数为17，行数为7，字符数为115，接着读入空格，字符数为116，行数为7，当前行数为8。

第8行，读入Lex，识别为标识符，按要求输出，此时单词数为18，行数为7，字符数为119，当前行数为8。

第8行，读入！，识别为其他符号，按要求输出，此时单词数为18，行数为7，字符数为120，当前行数为8。

第8行，读入”，识别为其他符号，按要求输出，此时单词数为18，行数为7，字符数为121，当前行数为8。

第8行，读入<，识别为专用符号，按要求输出，此时单词数为18，行数为7，字符数为122，当前行数为8。

第8行，读入<，识别为专用符号，按要求输出，此时单词数为18，行数为7，字符数为123，当前行数为8。

第8行，读入endl，识别为关键字，按要求输出，此时单词数为19，行数为7，字符数为127，当前行数为8。

第8行，读入;，识别为专用符号，按要求输出，此时单词数为19，行数为7，字符数为128，当前行数为8。接着读入/n，字符数为129，行数为8，当前行数为9。

第9行，读入空格，字符数为130，行数为8，当前行数为9。

第9行，读入return，识别为关键字，按要求输出，此时单词数为20，行数为8，字符数为136，当前行数为9。

第9行，读入空格，字符数为137，行数为8，当前行数为9。

第9行，读入0，识别为数字，按要求输出，此时单词数为21，行数为8，字符数为138，当前行数为9。

第9行，读入;，识别为专用符号，按要求输出，此时单词数为21，行数为8，字符数为139，当前行数为9。接着读入/n，字符数为140，行数为9，当前行数为10。

第10行，读入{，识别为专用字符，按要求输出，此时单词数为21，行数为9，字符数为141，当前行数为10，接着读入\n，字符数为142，单词数为21，行数为10。

# 5 Linux环境下lex.l源代码编写与执行

由于在实验一中已经将Linux虚拟机安装完毕并也已安装了Flex，其具体过程在此就不再赘述。

## 5.1 lex.l源代码编写与保存

打开文本编辑器，输入lex.l源代码并保存到桌面。

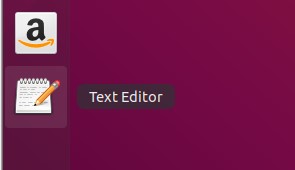


图19 打开文本编辑器

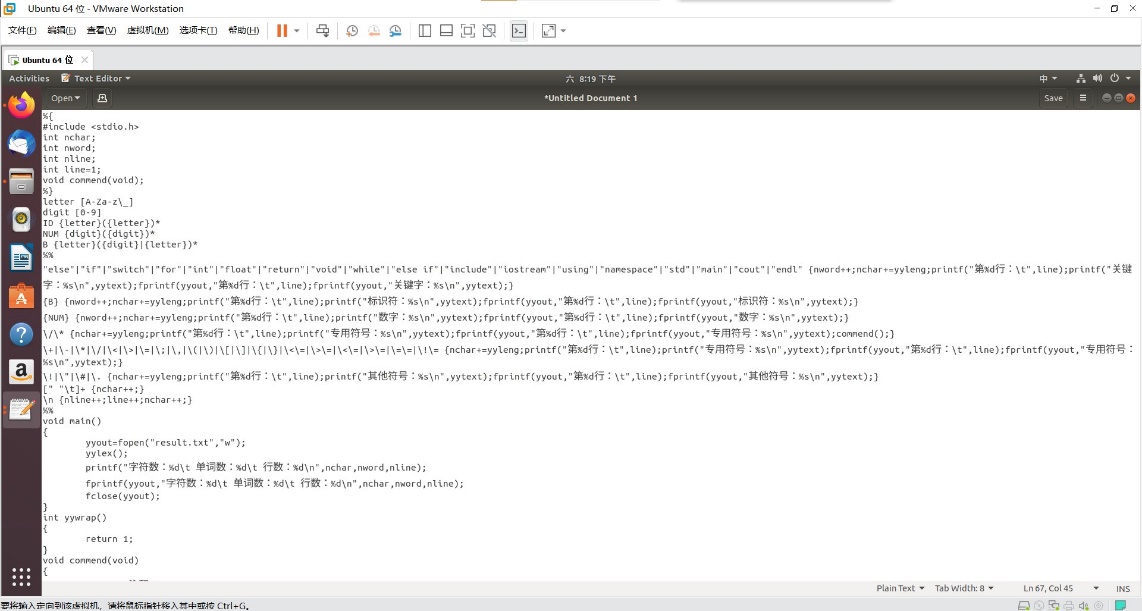


图20 编写lex.l文件 1

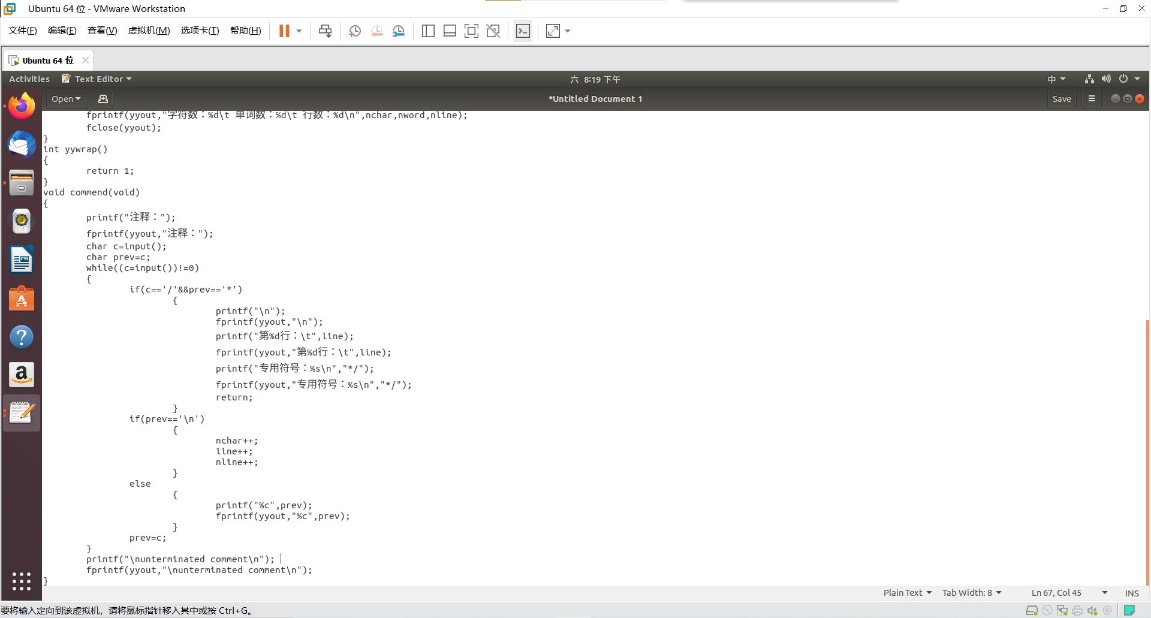


图21 编写lex.l文件 2

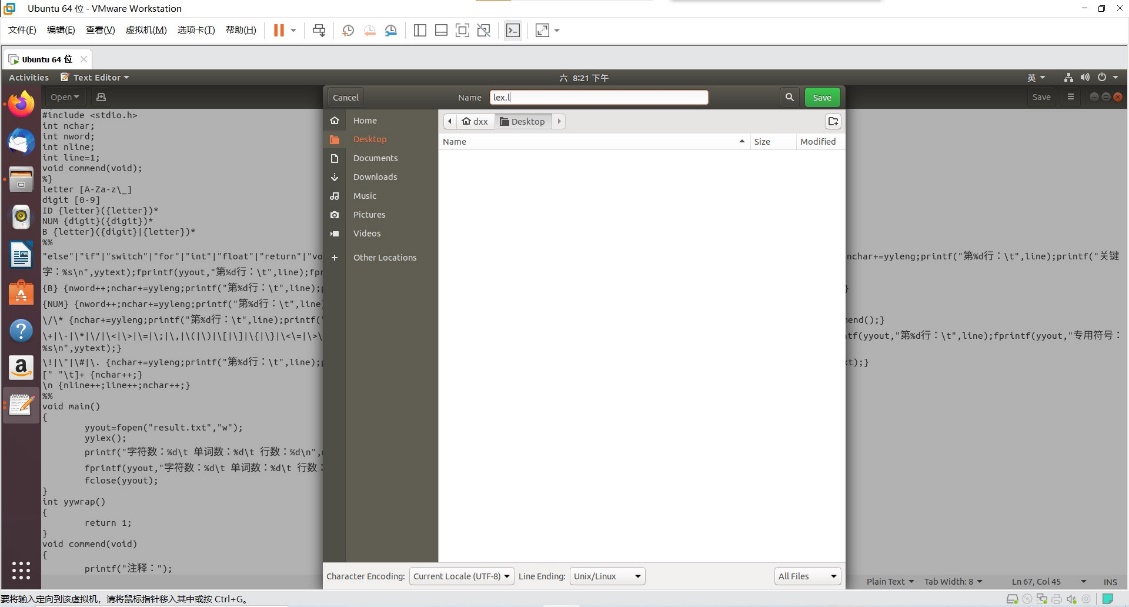


图22 保存lex.l文件至桌面

## 5.2 C语言代码编写与保存

打开文本编辑器，输入C语言代码，将其命名1.txt保存至桌面。

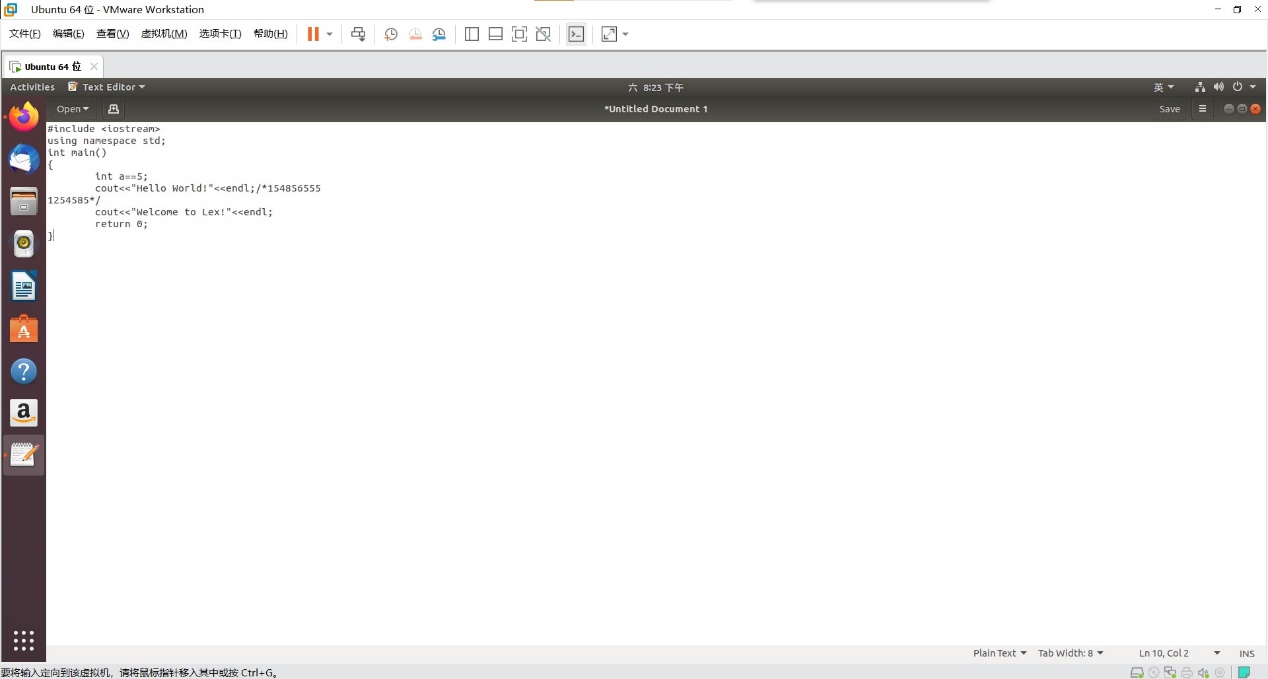


图23 编写C语言代码

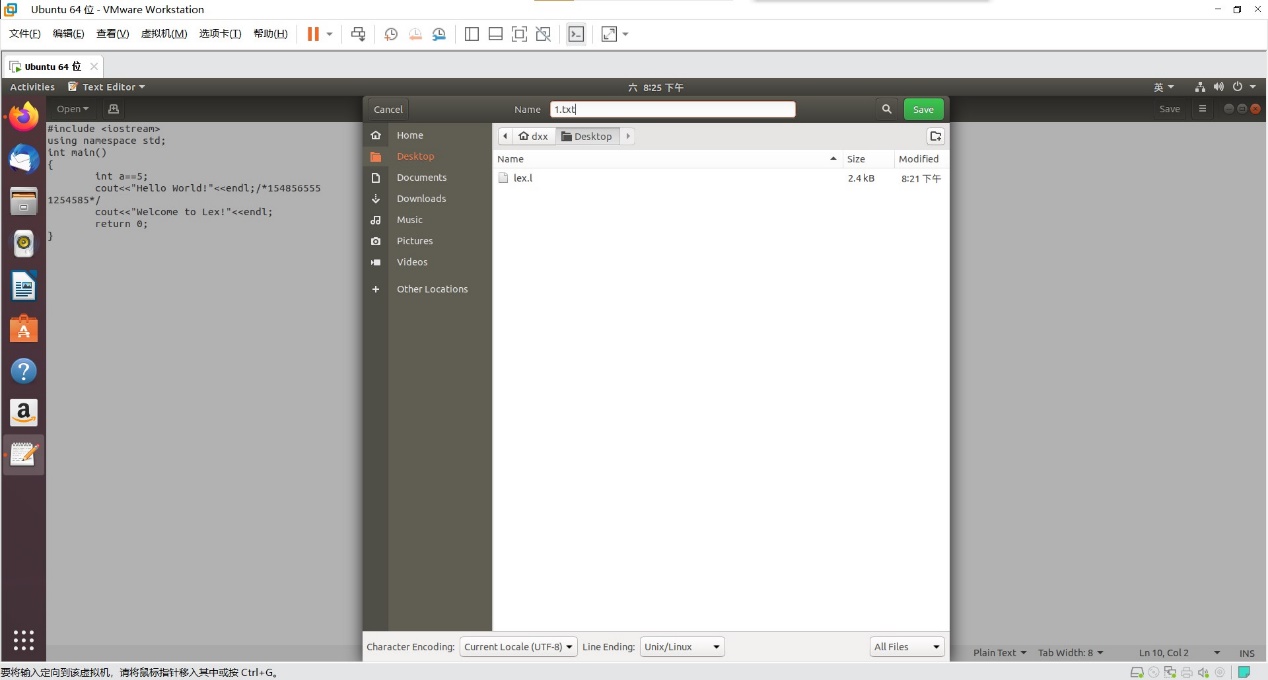


图24 保存1.txt文件至桌面

## 5.3 lex代码运行测试

打开命令窗口，进入桌面目录，依次输入以下命令：flex lex1.l、cc -o parser lex.yy.c、./parser < 1-1.txt，C语言代码将被解析并输出结果。

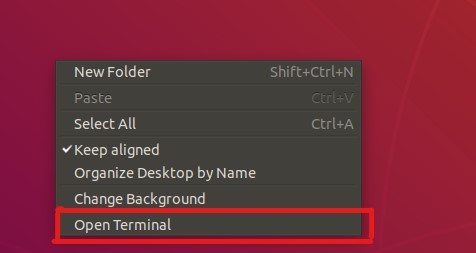


图25 打开命令窗口

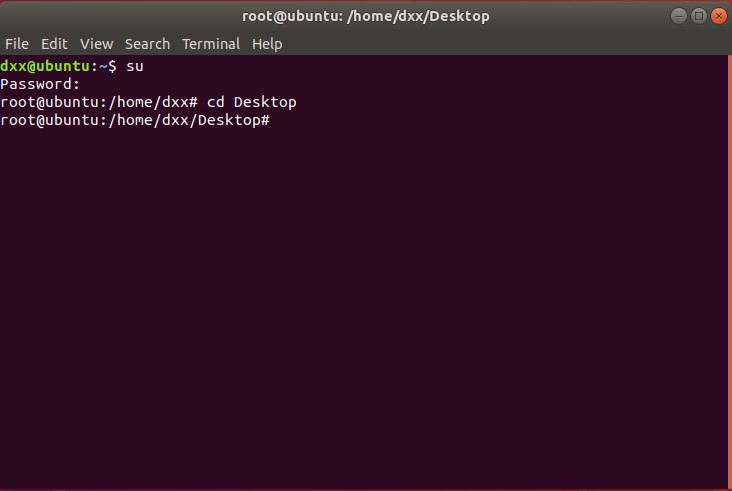


图26 进入桌面目录

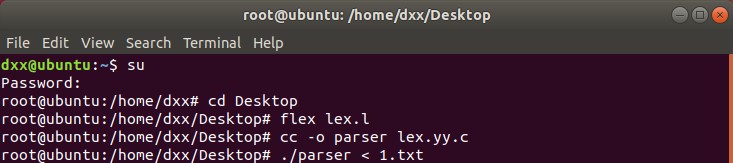


图27 命令输入

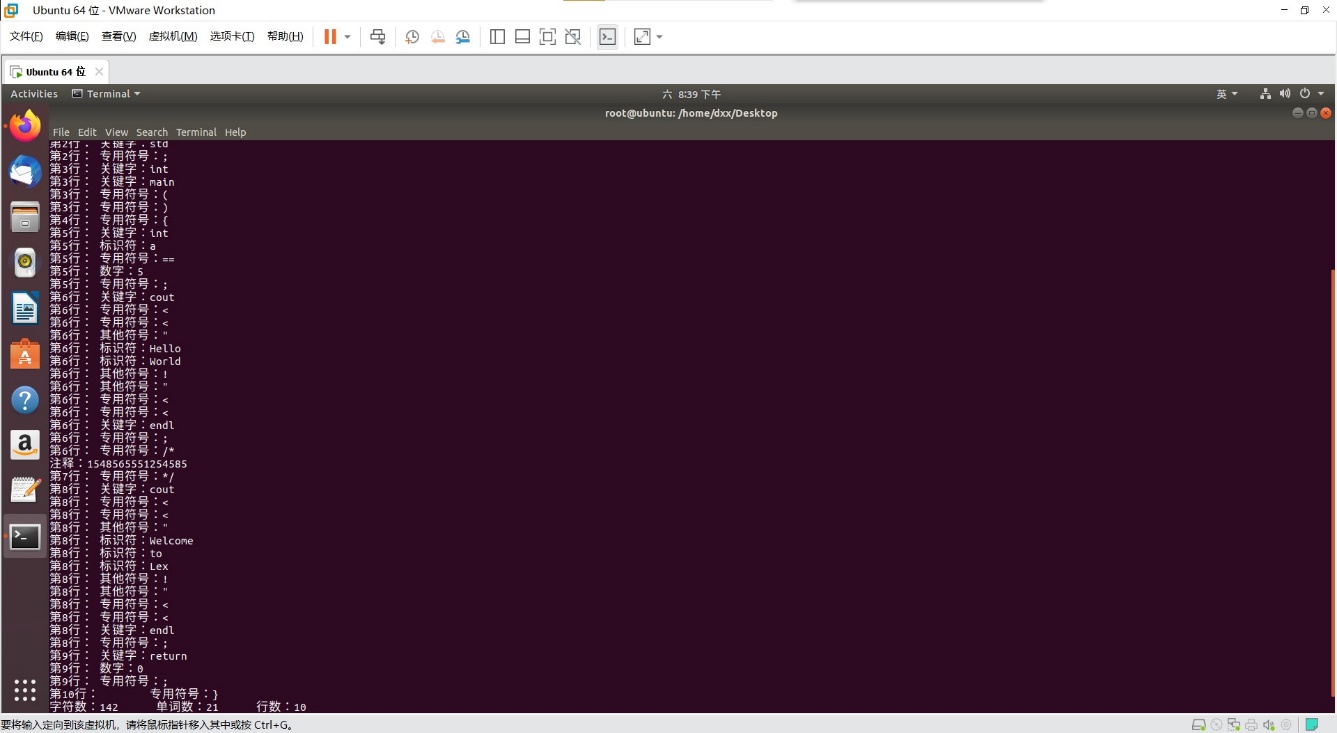


图28 结果显示

可以在桌面上看到生成的lex.yy.c文件、parser文件以及result.txt结果输出文件。打开result.txt文件可以看到输出结果。

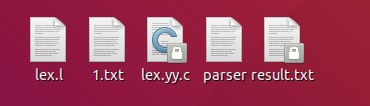


图29 生成文件

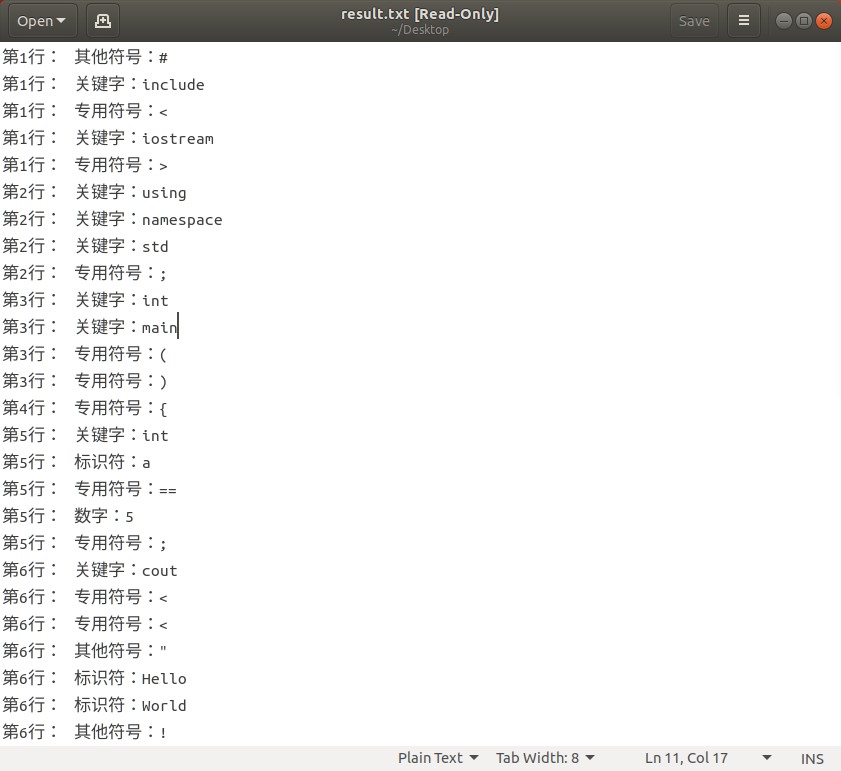


图30 结果输出文件1

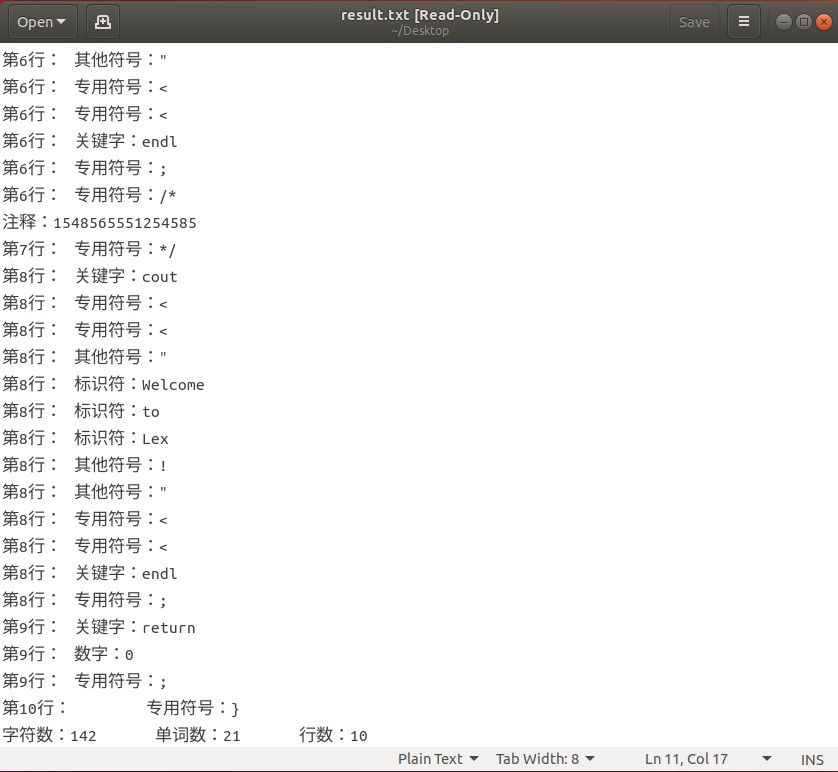


图31 结果输出文件2

## 5.4 分析程序输出结果

结果分析同Windows环境下的lex.l执行结果分析一致，在此不再赘述。

# 6 实验总结

在本次实验中，遇到了Windows环境下的Visual Studio 2019的编译器对于此次的lex.yy.c文件编译出错的问题，其在Linux环境下编译完全通过，但在Windows环境下出现代码错误与字符格式问题。经过一系列查找解决方案后，最终抛弃了Visual Studio 2019的自带编译器，而选择Cygwin编译器，它是把基于Linux的GNU编译器变相移植到Windows系统上，从而解决了编译出错的问题。另一方面问题是对于注释的处理和文件的输出，经过查阅一些资料和参考模板后，对注释增加了辅助函数，进行了注释输出处理，以及完成了词法分析结果输出到result.txt文件里的要求。

相较于实验一的程序代码，本次实验的代码量更大一些，也进一步细化了词法分析的各个符号和标点，进一步增强了词法分析器的功能，当然有些地方仍旧不完善，比如无法区分输入字符串和标识符。

通过本次实验进一步学习了Lex的语法规则，并利用Flex设计了一个简单的词法分析程序，完成了对C语言子集的词法分析并统计代码中的字符数，单词数和行数，其中也完成对注释的处理，并且学会了文本输出的方式。可以说，此次实验是受益匪浅，为今后的实验与学习打下了基础。