**电子科技大学 信息与软件工程 学院**

**标 准 实 验 报 告**

**（实验）课程名称编译技术**

**电子科技大学教务处制表**

**电 子 科 技 大 学**

**实 验 报 告**

**组队排序1 学生姓名：成棋伟 学 号：2022130102013**

**指导教师：周尔强**

**实验时间：2024.10.11 实验地点：信软楼西304**

**一、实验室名称：信软 303/304**

**二、实验项目名称：词法分析器**

**三、实验学时：4学时**

**四、实验内容及步骤：**

①实验概述：

编写xxx.l文件，借助Flex(Fast Lexical Analyzar)在特定指令下处理生成lex.yy.c的C语言源程序，即词法分析器。经过编译后，该源代码能够生成可执行程序，识别分析特定C源程序，并按照规则输出。

例如对于源代码

int main(){

return 3;}

词法分析器的输出为：

<261, KEYWORD>

<260, main>

<285, SYMBOL>

<286, SYMBOL>

<287, SYMBOL>

<269, KEYWORD>

<258, 3>

<292, SYMBOL>

<288, SYMBOL>

②实验背景：

**Flex输入文件的基本格式：**

%{

Declarations (optional)

%}

Definitions (optional)

%%

Rules

%%

User functions (optional)

其中第一部分为C语言声明，第二部分为相关定义，第三部分为词法分析器的规则（识别到特定pattern对应的action），第四部分为用户自定义的C语言函数。

**常用的正则表达式：**

[0-9] 字符集0|1|2|3|4|5|6|7|8|9

例如：[0-9A-Za-z]，[aeiou0-9]

^ 字符集的补集，需要是字符集中第一个字符

例如：[^0-9] 匹配任何非数字字符

. 匹配“换行符”之外的任何字符

x? x 可重复0或1次

x+ x 需重复至少1次

x{n,m} x 需重复最少n次,最多m次

^x ^ 表示每行的最开始（虚拟的位置）

即匹配x，且x为该行第一个字符

x$ $ 表示每行的最末尾（虚拟的位置）

即匹配x，且x为该行最后一个字符

"x" 表示x本身，即使x为特殊符号

"x\*" 表示匹配x和\*号两个字符

③实验环境

Virtualization: vmware

Operating System: Ubuntu 23.10

Kernel: Linux 6.5.0-35-generic

Architecture: x86-64

Firmware Version: 6.00

测试平台：icoding

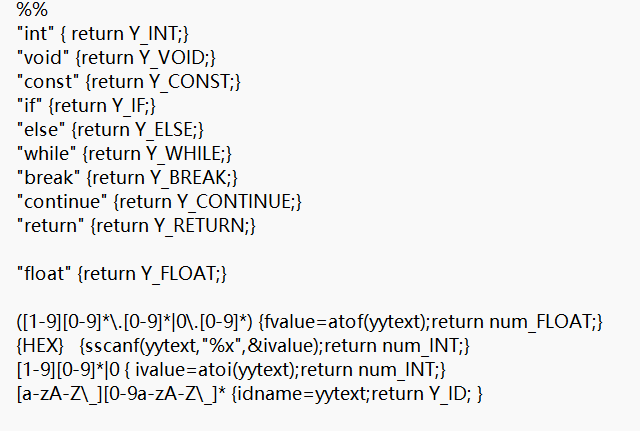
④实验步骤

**代码重构**

为了方便编译调试，我们可以将提供的头文件token.h里面的枚举定义放入同一目标文件auto\_flex.l中。

**正则表达式规则实现**

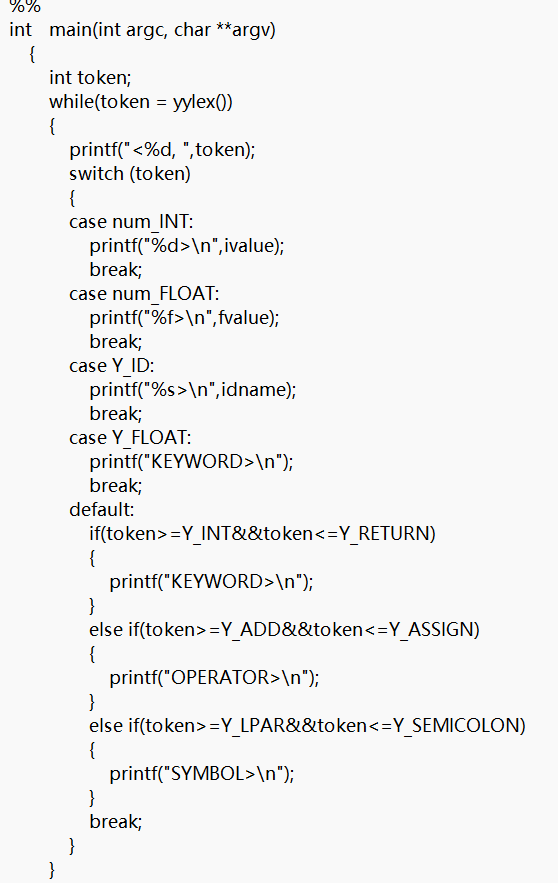
* + 定义一系列的正则表达式规则，用于匹配不同类型的词法单元，如关键字、标识符、数值、操作符和分隔符。
  + 关键字如int、void、if等直接返回对应的枚举值。
  + 浮点数和整数通过正则表达式匹配，使用atof()和atoi()函数将其转换为浮点数和整数。
  + 十六进制数通过正则表达式匹配，并使用sscanf()函数转换为整数。
  + 标识符通过正则表达式匹配，并存储其名称。
  + 操作符和分隔符如+、-、\*、/等被直接返回对应的枚举值。
  + 匹配注释//并将其忽略，同时空白字符（空格、制表符）和换行符也被忽略。



**主函数部分**

我们可以调用标识token,结合switch()函数对识别目标进行分类讨论。

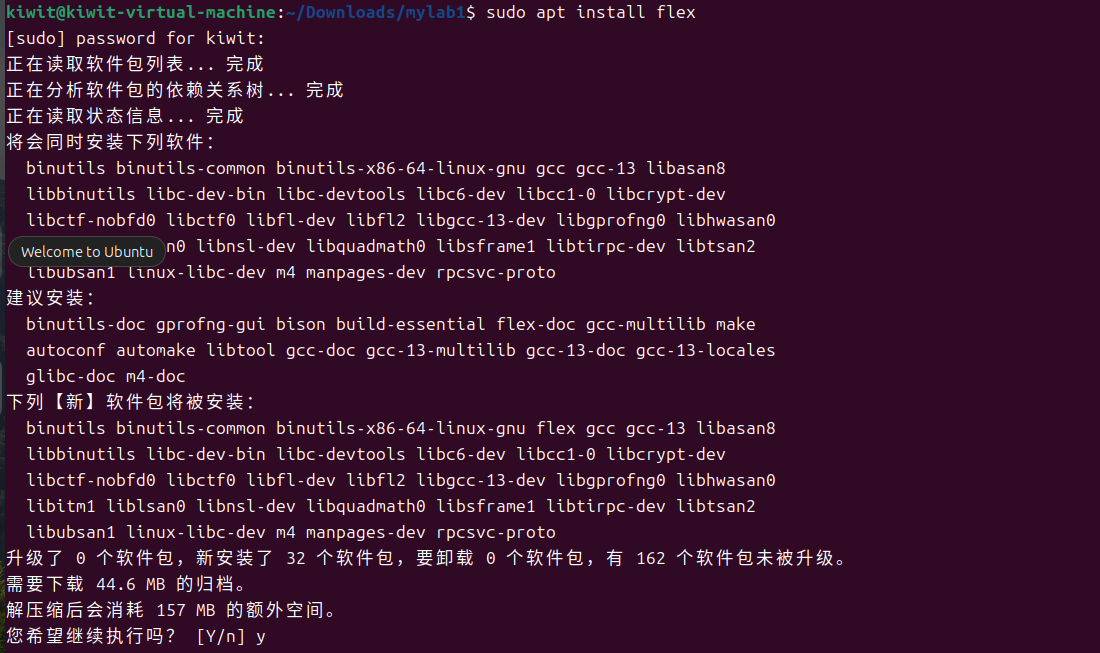
对于每个识别的词法单元，根据其类型打印不同的信息。比如：整数和浮点数打印其值，标识符打印其名称，关键字、操作符和分隔符打印相应的类别。



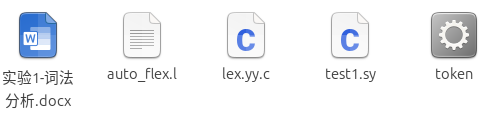
1. **实验运行结果及测试：**

**在LINUX环境测试**

命令行中输入”sudo apt install flex”安装配置flex相关环境



之后输入“flex auto\_flex.l”自动生成lex.yy.c文件



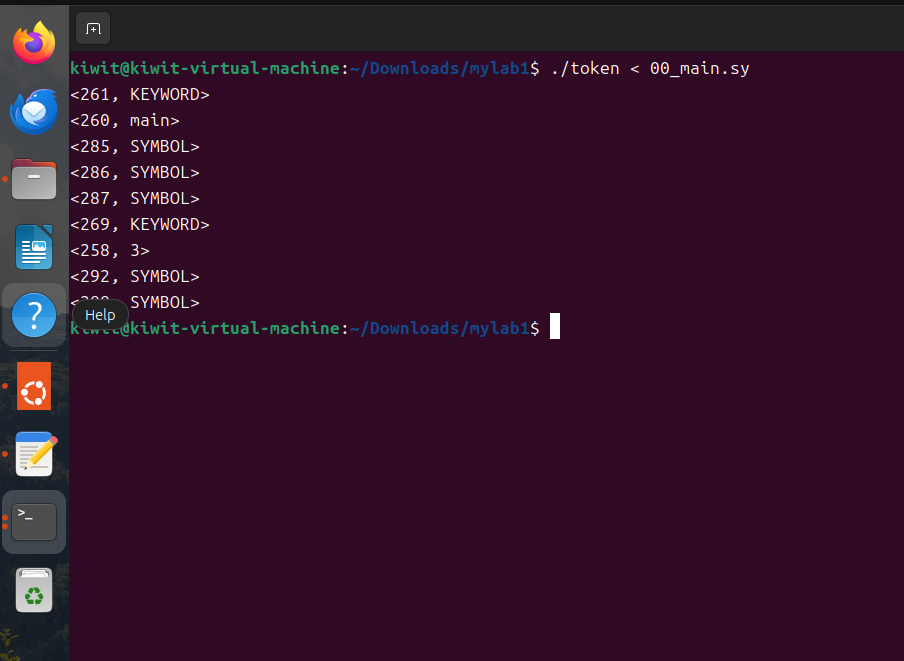
再然后使用GCC编译lex.yy.c文件，链接相关库，并生成可执行文件：

“[gcc](https://so.csdn.net/so/search?q=gcc&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/m0_73181729/article/details/_blank) -o token lex.yy.c -lfl”

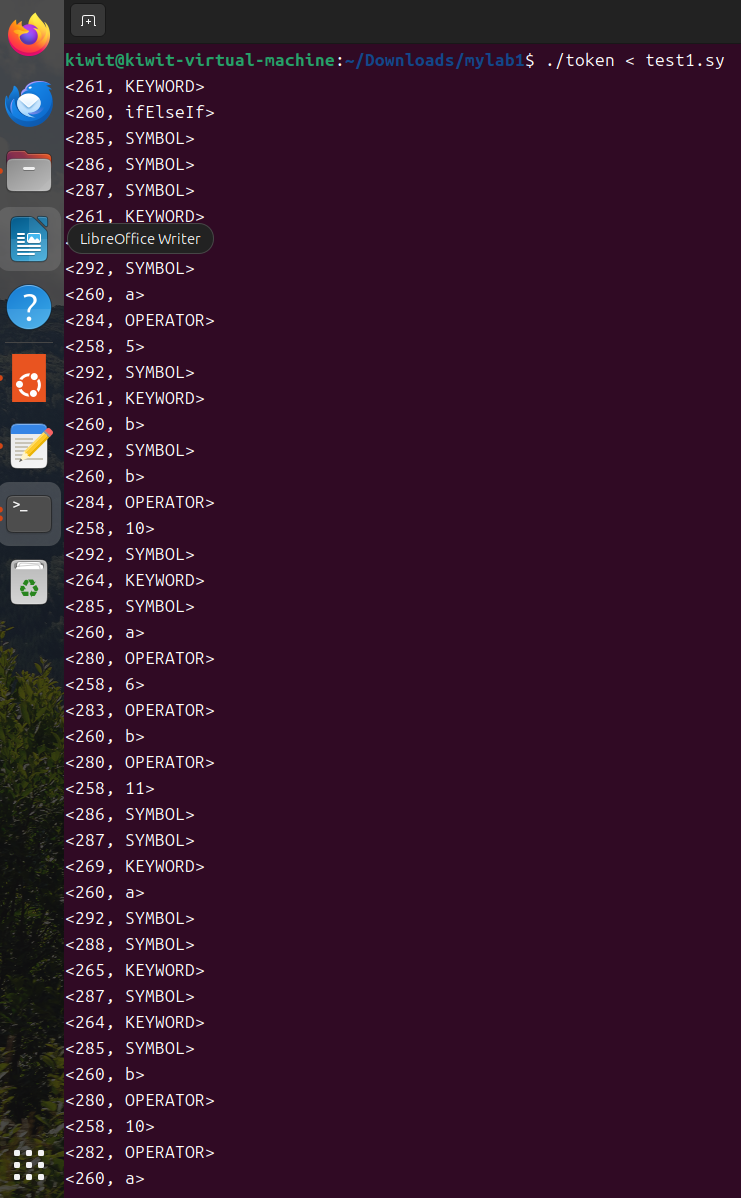
上图中可以看到token字样的文件就代表成功生成。

再之后我们在test\_cases文件夹中导入测试用例,

输入命令“./token < 00\_main.sy”重定向测试文件，从而进行词法分析



当然我们也可以自己调用其他测试用例，不再一一列举了。



1. **实验结论与总结：**

通过本次实验，我了解到了如何使用Flex框架模型来设置生成词法分析器；熟悉了linux环境下文件的配置、编译和运行；实验经历也加深了我对正则表达式的认识和了解，对词法分析器的实现细节也从课堂上单一的理论知识转换到了实践认知当中，拓宽了对整个编译器前端的理解。

**报告评分：**

**指导教师签字：**