

### 编译原理实验一

**词法分析与语法分析**

学 院 计算机科学与技术

学　　 号

班　　 级

学 生

日 期 2022年3月26日

## 编写文件列表

lexical.l : Flex源代码文件，用于编译生成C语言的词法分析器。

syntax.y : Bison代码文件，用于编译生成C语言的文法分析器。

main.h : main.c头文件。

main.c : 主程序入口，根据输入文件进行词法分析和文法分析，并提供构建和遍历输出语法分析树、错误提示等功能的工具函数。

## 程序实现功能

#### 词法分析

按照实验指导教程编写Flex文件lexical.l，引用syntax.tab.h，在规则部分除注释每一条action返回相应的词法单元，编译生成lex.yy.c用于词法分析。

#### 注释

按照实验指导教程，单行注释“//”在被匹配后，丢弃在同一行的其后面的所有输入，即读取到‘\n’。

多行注释“/\*”“\*/”，由于指导书说明C--不支持嵌套的多行注释，则我们可以使用词法分析进行识别处理。编写正则表达式 "/\*"([^\\*]|(\\*)\*[^\\*/])\*(\\*)\*"\*/" 并在action中不做任何事来丢弃。

#### 文法分析

按照实验指导教程及其后面的C--附录文法描述编写Bison文件syntax.y，借助Flex词法分析函数，进行文法分析。引用main.h，定义终结符或非终结符的属性值为分析树节点node，并在语法分析过程中调用main.c辅助函数insertNode构造分析树，在规约到Progarm后分析完毕，调用preorderTraversal先序遍历分析树输出分析树。

#### 构建语法分析树

在main.c中编写构建和遍历输出语法分析树、错误提示等功能的工具函数。

构造分析树节点结构体，因为分析树为多叉树，成员变量用于保存节点的token，行号，兄弟节点和最左儿子节点，因为终结符由其类型确定且不能预先确定，使用union终结符的属性，节省空间。

编写函数，自底向上构建分析树：每当进行规约，为产生式右部的非终结符创建新的分析树节点，将相关信息记录到成员变量中，将产生式右边的节点按照顺序作为产生式左边节点的子节点向上构建分析树。其中八进制、十六进制常数需要进行进制转换，再记录。

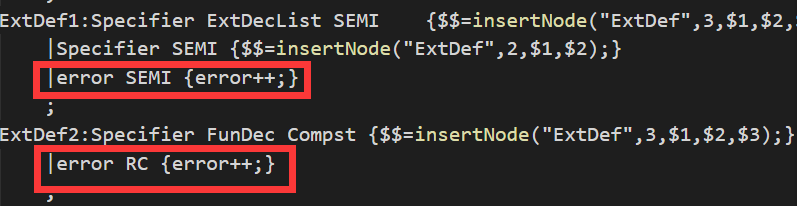
遍历分析树时，按照多叉树遍历方法，先序遍历分析树。终结符和非终结符打印各自需要打印的信息。

#### 2.5 错误恢复

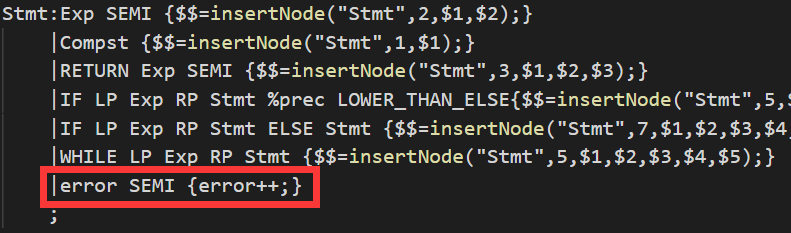
在分析错误时，要求能忽略错误继续分析。

参照龙书中的“Yacc错误恢复”，观察C--文法示意图，在可能出错的位置分别加入虚构词法单元error。

根据ExtDef结尾的词法单元不同，将ExtDef分为ExtDef1和ExtDef2，分别使用不同的error产生式，跳“；”和“}”后。如图。



Stmt递归分析其只能由“；”结尾，故添加下图中的error产生式。

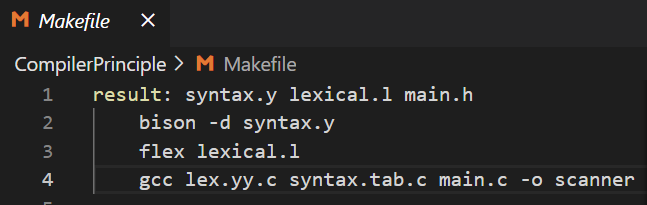


对错误数使用全局变量error计数，只有error=0时才在分析结束时打印分析树。

## 编译和使用方法

#### 3.1编译

使用Makefile进行编译，进入目录后在终端输入make，按照Makefile文件内进行编译和链接后，生成可执行文件scanner。



#### 3.2使用方法

./scanner [待分析文件名]

例： ./scanner input1.cmm