1. 词法分析

词法分析模块可以对语法文本进行扫描，如果发现不符合定义的词素会根据格式报告相关错误，符合定义的词素会执行相应的动作创建语法书叶子节点并返回 词汇单位。描述各种词的词汇规则

1. 标识符
   {letter}({letter}|{digit})*
2. 关键词
   int float struct return if else while
3. 运算符号
   + - * / && ||
   RELOP >|<|>=|<=|==|!=
4. 边界符号
   SEMI ;
   COMMA ,
   ASSIGNOP =
5. 括号
   ( ) [ ] { }
6. 注释
   comment1 (\/\/[^\n]*)
   comment2 (\/*((([^*])|(*[^\/]))*)*\/)
   comment {comment1}|{comment2}

如果没有匹配的字符 显示为无效字符并打印错误信息“.”。用于不匹配的字符被解析为无效字符的符号。 设置变量 haserror=1，表示出错。

```
{
    fprintf(stderr,"Error type A at Line %d: Mysterious characters \"%s\".\n",yylineno,yytext);
    err = 1;
}
```

2. 语法分析

语法分析模块基于基本语法规则。 并增加左右组合的处理过程 优先 和错误恢复它可以根据指定的格式检查语法并报告错误。 如果语法正确 到达根节点时执行该规则。 打印语法映射的操作，语法单位受到左右组合和优先级的限制。

```
23   %right ASSIGNOP
24   %left OR
25   %left AND
26   %left RELOP
27   %left PLUS MINUS
28   %left STAR DIV
29   %right NOT
30   %left LP COMMA RP LB RB DOT
31   %nonassoc LOWER_THAN_ELSE
32   %nonassoc ELSE
```

语法单元规范的处理流程，以 ExtDefList->ExtDef ExtDefList 为例进行说明。 当可以按照这种语法进行规范时，要执行的动作是依次创建子节点的数据列表，并根据子节点的数据列表调用节点创建函数。 创建一个父节点并将返回的指针分配给左侧的语法单元。 函数的参数依次代表语法单元名称、第一次出现位置、子节点个数、子节点数据数组。

```
37  ExtDefList : ExtDef ExtDefList {struct Ast* t[2]={$1,$2};$$=newNode("ExtDefList",@1.first_line,2,t);}
38             | {$$=newNode0("ExtDefList",0);}
39             ;
```

以下函数用于向树添加节点、打印地图以及在语法不匹配时打印错误。

```
void yyerror(const char* msg)
{
    //ERR = 1;
    //fprintf(stderr,"Error type B at Line %d:%s\n",yylineno,msg);
}
void yyError(char* msg)
{
    ERR = 1;
    fprintf(stderr,"Error type B at Line %d:%s.\n",yylineno,msg);
}
```

3.实现及结果分析

1. Makefile 用于编译一个解析器程序，测试输入文件的词法和句法分析
   a) 词法的第一个命令弹性词法.l
   b) 语法的第二条命令野牛 -d -v 语法.h
   c) 解析器程序的第三条命令 gcc main.c syntax.tab.c -lfl -o 解析器
2. 只需调用以下命令即可调用所有命令制作
3. 可以使用以下任何命令使用测试文件测试解析器
   a) 使用解析器 ./parser/input1.c
   b) 使用 make，对于 input1.c，对于 input3.c 测试文件制作 t1

```
┌──(kali㉿kali)-[~/Desktop/Code]
└─$ ls
input1.c  input3.c   lex.yy.c  run          syntax.tab.h  tree.h
input2.c  lexical.l  main.c    syntax.tab.c  syntax.y

┌──(kali㉿kali)-[~/Desktop/Code]
└─$ flex lexical.l

┌──(kali㉿kali)-[~/Desktop/Code]
└─$ bison -d -v syntax.y

┌──(kali㉿kali)-[~/Desktop/Code]
└─$ gcc main.c syntax.tab.c -lfl -o parser

┌──(kali㉿kali)-[~/Desktop/Code]
└─$ ./parser input1.c
Error type A at Line 4: Mysterious characters "~".

┌──(kali㉿kali)-[~/Desktop/Code]
└─$
```