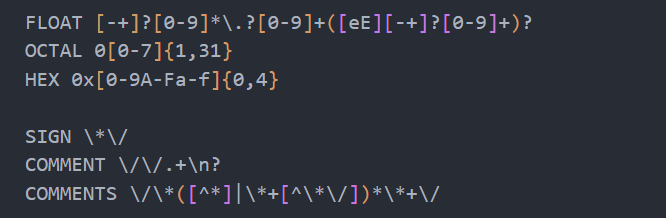
1. 主要功能

本次实验中，我使用flex和bison编写了一个可以针对C--语言进行词法分析和语法分析的程序parser，并且通过了实验指导书提供的十个测试用例。

1. **词法分析**

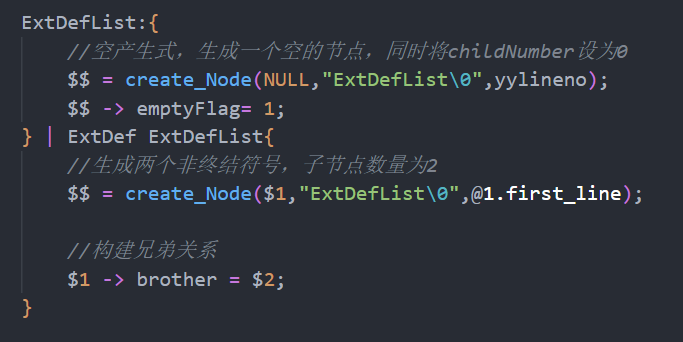
通过正则表达式来进行单词的识别，完成带指数的小数，八进制和十六进制数，以及注释的识别。其中针对注释不做处理，识别后直接跳过即可。

同时在识别的时候针对每一个词语都进行语法树节点的生成，遇到ID，INT（包括Octal和Hex）和FLOAT等需要存储值的节点，生成节点的同时还会将其对应的值从字符单词转化成对应的数据。例如针对013这个八进制数，要将其转化为13存到节点当中。如果没能检测到该单词，那么就用.来进行匹配并输出报错信息

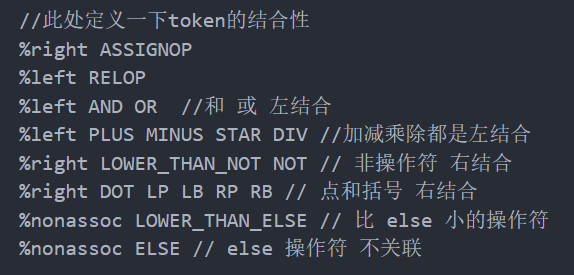


1. **语法分析**

利用bison，根据指导书提供的文法写好了token后，开始按照产生式来进行表示式的书写，此时将其产生式右部作为产生式左部节点的子节点，如果有多个子节点，子节点之间也会以兄弟节点指针进行连接。



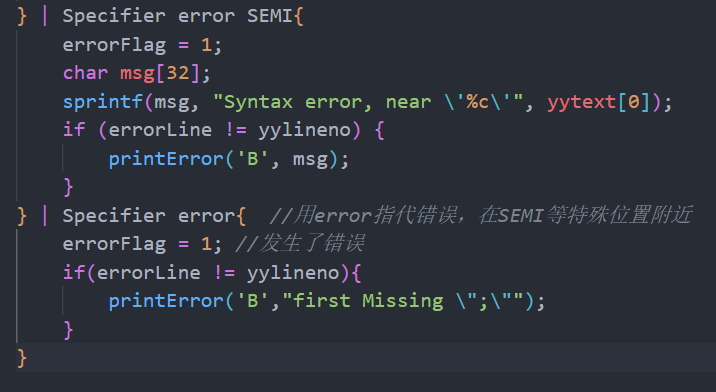
需要注意的是，根据实验指导书的提示，需要对不同的token进行结合性的判断以消除bison的二义性。



1. **错误提示**

在这里为了简化错误的打印，实现了一个printError()的函数来打印A型或B型错误，其中按照实验指导书的要求来打印具体错误，并且开启了yylineno来跟踪行号，一方面可以跟踪如果行号发生变化，意味着出现了新的错误。

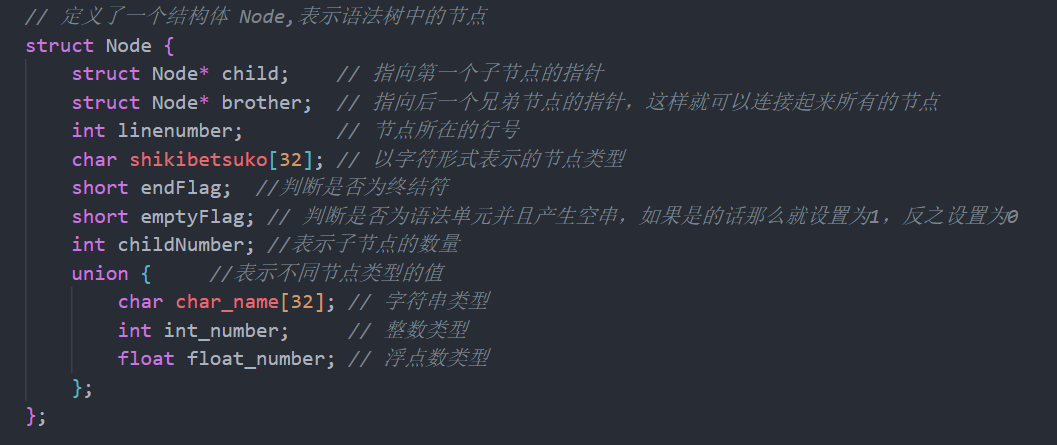
根据实验指导书的内容，可以在生成式的右部使用error来进行提示。同时，error的填入位置也需要考量，我主要将其放在了SEMI，COMMA，LP和RP之间等特殊的位置，方便对程序进行更有效的错误提示。



1. **打印语法树**

该功能中比较麻烦的是定义一个良好的数据结构来存储分析树的节点。语法分析树是一个典型的不固定的多子节点树，所以定义的时候我采用父子-兄弟树，并且其中有标识其语法成分和具体数据，以及特殊标志的字段。

之后打印只需要按照实验指导书的要求，利用depth来规定缩进的深度即可，同时声明一个head指针作为入口。

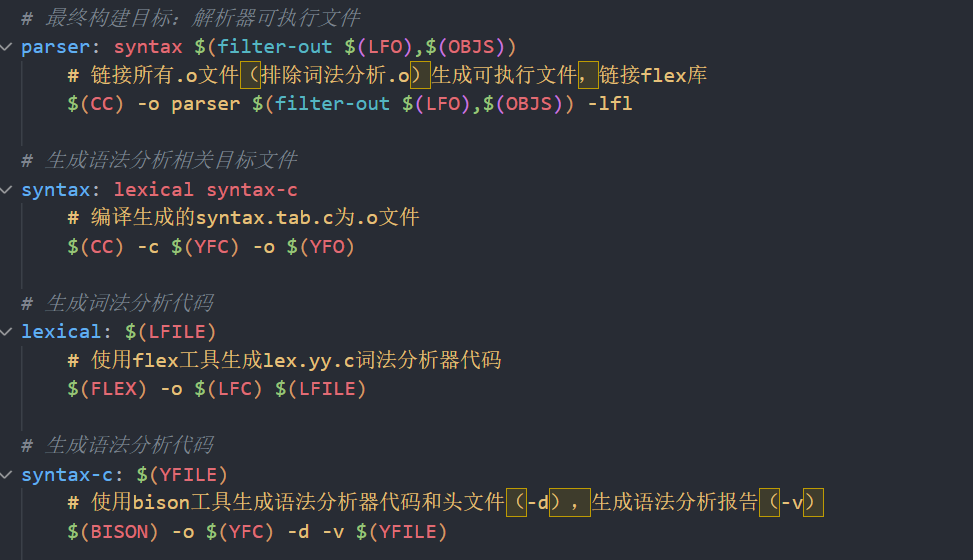


1. 编译过程

使用了makefile文件进行编译，借助了一下AI大模型进行更快的编写。makefile文件比手动编译和用sh脚本都要更加方便一些，可读性也更高一些。

主要编译的流程为

1. 先利用flex进行词法分析程序lexical.l的编译，生成lex.yy.c文件
2. 使用bison对语法分析的程序进行编译，主要生成syntax.tab.c和syntax.tab.h文件
3. 将上一步编译出的syntax.tab.c文件编译为syntax.tab.o文件
4. 将.c文件都编译为.o文件之后进行连接，生成parser文件，即可读入其他程序文件进行编译



此外为了方便测试和文件管理，makefile文件中还规定了两个伪目标test和clean，前者会用生成好的parser文件对test文件夹下的测试用例进行编译，后者会清除掉编译过程中生成的其他文件，只保留parser文件以使得目录更加简洁。

Makefile文件写好之后，只需要进行（1）make （2）make test （3）make clean三种命令即可完成编译，比较方便。