编译原理实验 1 实验报告

161220143 吴御洲

一、概述

实现了对 c—程序的词法分析和语法分析，对于存在错误的程序可以报出错误类型和所在位置；对于正确的程序可以打印出语法树。

此外实现了选作 1.1，即十六进制和八进制数的识别。

二、文件结构

lexical.l 文件中实现了正则表达式匹配词法单元，维护了 bison 程序编写过程中语法单元位置的维护，同时规定了对对应词法单元执行的操作；

syntax.y 文件中规定了词法单元语法单元属性值的类型、运算符的结合性和优先级，规定了产生式并根据产生式去生成相应的语法树；

syntaxtree.h 中设计了语法树节点的结构，定义了生成节点、生成树和打印树的函数；

syntaxtree.c 中具体实现了.h 文件中 3 个函数；

main.c 使用了书上的 main 函数。

三、具体实现

lexical.l syntax.y 文件中的内容基本参考书后附录 A

节点包括 name, line_no, type, FirstChild, NextSib 和一个 union，分别为节点对应语法单元名，初现时所在行号，类型(1 为词法单元，-1 为产生空串的，0 为其它)，第一个子节点的地址，自己下一个兄弟节点的地址；union 里有 int double char*三种，分别存储 INT, FLOAT, ID/TYPE 的词法单元的内容。

因为有不同类型，所以函数接收的参数数目可能不一样，故使用 va_list 来处理。

新建节点的函数在 lexical.l 里面调用，可能是 2 个参数或 3 个参数，前两个一定是词法单元名和所在行，第三个只有 INT FLOAT ID/TYPE 有，即实际值。所以在函数中分别赋值，两个指针赋 NULL，type 设为 1，之后考察第一个参数是哪一种，去调用 va_arg，根据数据类型去获取第三个参数，并赋给 union 里的变量。

新建树的函数在 syntax.y 里面调用，至少 4 个参数，分别是语法单元名字，所在行，值，后续输入的个数，先读入这四个，并进行相应的赋值，type 赋 0，如果后续输入个数为 0，说明这个可以产生空串，type 修改为-1，值这个变量为词法单元才有用的，这里为了结构的一致性，直接给 int 类型的赋 0；之后同样使用 va_list，根据传入的第四个参数，循环读入后续，并根据 FirstChild 是否为 NULL 来将子节点依次加入。

打印树的函数有两个参数，第一个为根节点的地址，第二个为缩进数量（第一次调用时为 0），首先根据第二个数打印空格缩进，后面按要求打印，最后依次递归调用，打印所有

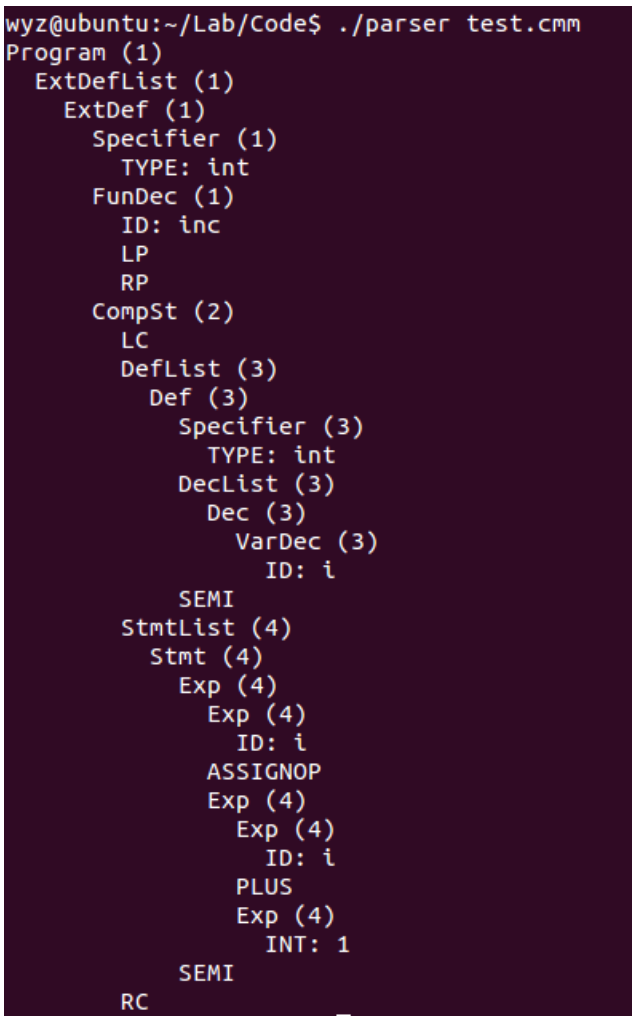
子节点。

四、如何运行

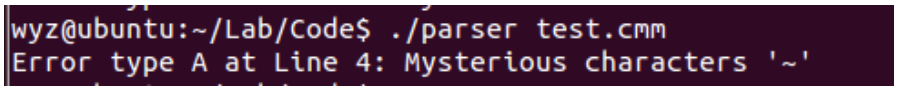
所有文件放在/Code 下，在/Code 输入 make parse 后生成可执行文件，之后将 xxx.cmm 测试文件也放在/Code 下，输入./parse xxx.cmm，即可显示结果。

五、效果展示（均为书上例子）

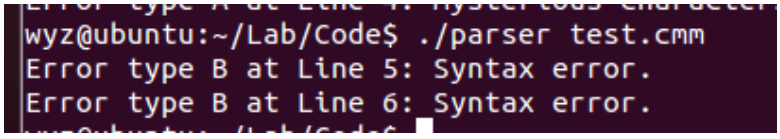
例 1.3



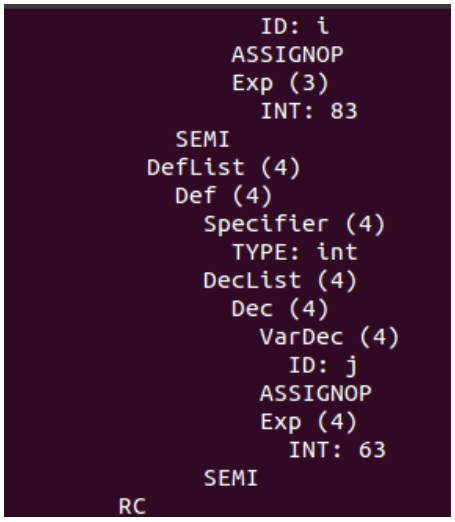
例 1.1



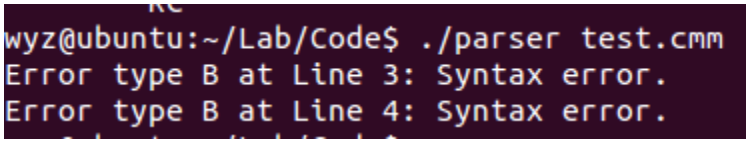
例 1.2



例 1.5（查看进制转换结果一致）



例 1.6



六、备注

在执行 make parser 时出现了

make: Warning: File `lexical.l' has modification time 5.2e+04 s in the future

flex -o ./lex.yy.c ./lexical.l

bison -o ./syntax.tab.c -d -v ./syntax.y

gcc -c ./syntax.tab.c -o ./syntax.tab.o

gcc -std=c99 -c -o main.o main.c

gcc -std=c99 -c -o syntaxtree.o syntaxtree.c

gcc -o parser ./syntax.tab.o ./main.o ./syntaxtree.o -lfl -ly

make: warning: Clock skew detected. Your build may be incomplete.

根据查到的结果可能是时间不同步，通过执行下面的命令解决了

find . -type f | xargs -n 5 touch

make clean

make parser

不知道在检查时会不会出现同样的问题