编译原理实验1报告

Note:使用`make parser`即可正常编译。

（一）实验概述

（1.1）词法分析

这一部分主要是正则表达式的编写。
比较复杂的的正则表达式如下：

- 浮点数

> 浮点数的定义需要仔细阅读讲义，我分为两种情况：
> （1）`\d+\.\d+`：讲义要求：小数点前后必须有数字出现。比如.9、9.都是不合法的浮点数
> （2）科学计数法的浮点数：
> 根据讲义，我进一步分为两种情况：
> - [ ] 01.23E12：小数点前后必须均有数字出现
> - [ ] 43.e-4或.5E03：小数点前或者后必须要有数字出现
> 之所以分类讨论是因为类似.E11等在其它语言可能合法的浮点数在本实验均不合法

- 各种进制的数字，这里要注意十进制不允许有多余的前导零出现，各种进制识别需要注意避免覆盖。
- 多行注释。 这里我借鉴了Bison的手册:

```
"/*"   {BEGIN(COMMENT);}
<COMMENT>"*/" {BEGIN(INITIAL);}
<COMMENT>([^*]|\n)+|.
<COMMENT><<EOF>> {
    printf("Error type A at Line %d: EOF in comment.\n", yylineno);
    lexical_error = 1;
    BEGIN(INITIAL);
}
```

- 边角细节：比如使用`atoi、strol`函数将字符串转数字时需要注意为`unsigned int`

（1.2）语法分析

除了按照讲义进行规则的撰写，我觉得比较复杂的是错误处理和恢复。

错误信息打印

使用bison内置的错误信息打印方式：

```
%define parse.error verbose
```

添加错误匹配规则

拿全局ExtDef举例，**其正确的语法包括全局变量定义、结构体定义、函数定义。**其错误恢复主要考虑了以下情况：

- **;、}等同步符号**
  比如：

```
ExtDef : ...
                error SEMI {}
```

- **分号缺失**
  比如：

```
ExtDef: ...
        | Specifier ExtDecList error {
        $$ = NULL;
        Warn("Global Definition Likely Missing ';'"); // 全局变量定义缺失分号
        yyerrok;
        };
        | Specifier error {
        $$ = NULL;
        Warn("Struct Definition Likely Missing ';'"); // 结构体定义缺失分号
        yyerrok;
        };
```

其它的非终结符也是采用了类似的原理进行分析。此处不再赘述。

（二）调试理论

（1）日志打印
通过宏定义，我可以很方便的开启或者关闭日志调试（对应Code目录下的logger.h文件）部分关键内容如下：

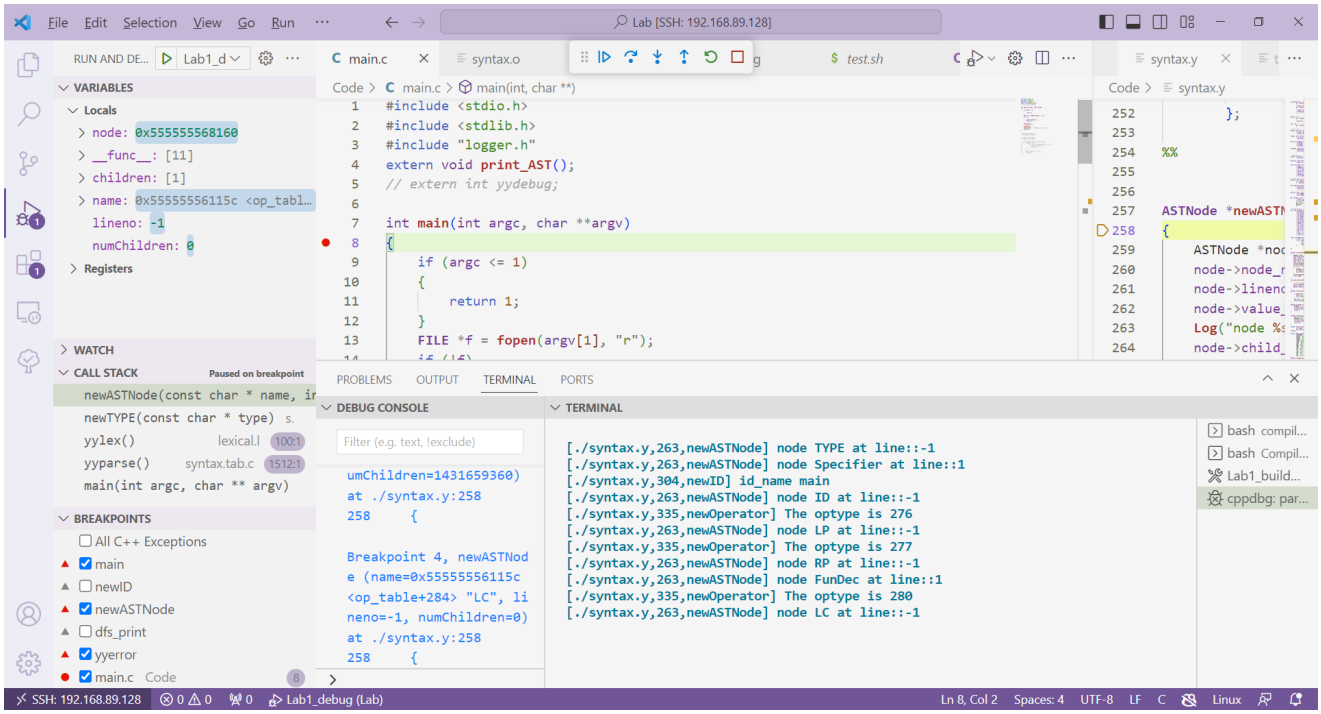
```
#define LOG_ON
#ifdef LOG_ON

#define Log(format, ...) \
  printf("\33[1;36m[%s,%d,%s] " format "\33[0m\n", \
      __FILE__, __LINE__, __func__, ## __VA_ARGS__)

#define Panic(format, ...) \
  do { \
```

```
    printf("\33[1;31m[%s,%d,%s] " format "\33[0m\n", \
        __FILE__, __LINE__, __func__, ## __VA_ARGS__);\
    assert(0);\
    }while (0)
```

（2）单步调试环境配置

使用了Vscode里的单步调试环境，在编译的时候添加`-g`选项。通过变量、断点、调用栈查看可以方便地跟踪程序执行流程。（对应代码的.vscode中的配置文件）

下面的图片右下角的蓝色日志即为logger.h调试宏开启之后的效果。可以非常清楚地看到词法和语法分析的过程。



（3）回归测试脚本(Code目录下的test.sh脚本)

我主要是使用了`diff`指令：

```
    if diff ../Test/myanswer$i.txt ../Test/answer$i.txt >/dev/null; then
        echo -e "\033[1;32mPassed!\033[0m"
    else
        echo -e "\033[1;31mFailed!\033[0m"
        echo "Differences for test $i:"
        diff ../Test/myanswer$i.txt ../Test/answer$i.txt
    fi
```

(4)开启Bison内部的debug宏查看状态机匹配过程（因为讲义有写，此处略去）