词法分析程序测试方案

1. 测试目标

能够实现语法分析程序的基本功能；能够以树形式输出词法分析的结果和错误信息，错误信息包括错误所在行数和不匹配token；能够输出语法分析输入的文件信息和程序执行时间时间。

1. 测试环境

操作系统：macOS Mojave

编辑器：vs code

代码类型：C++

1. 程序说明

代码放在Source Code，分别是SyntaxAnalysis.h和main.cpp两个文件。SyntaxAnalysis.h实现了与语法分析有关的函数，main.cpp实现了整个SyntaxOut.txt输出的大体框架。

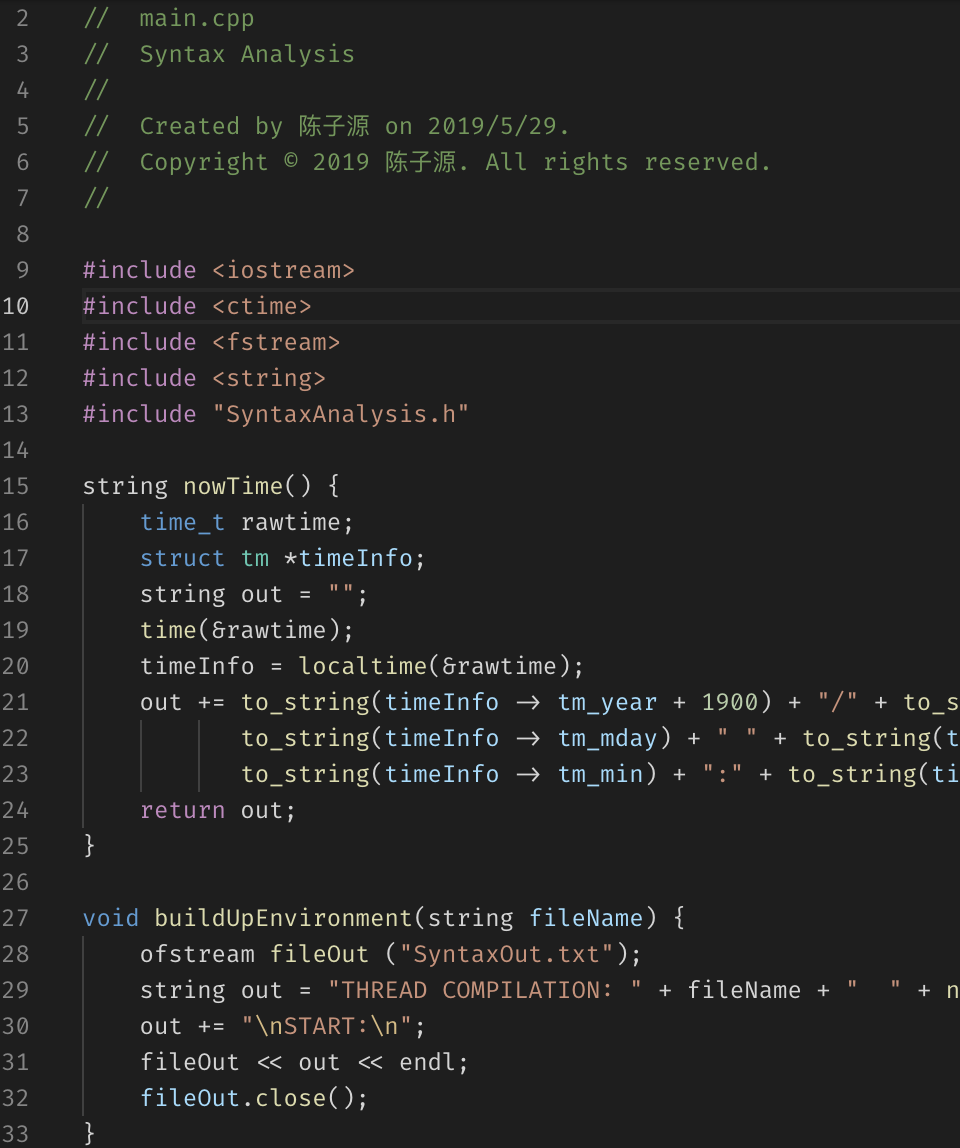
代码的编译和运行通过命令行终端实现，先是进入到Source Code文件夹，然后运行以下两个命令



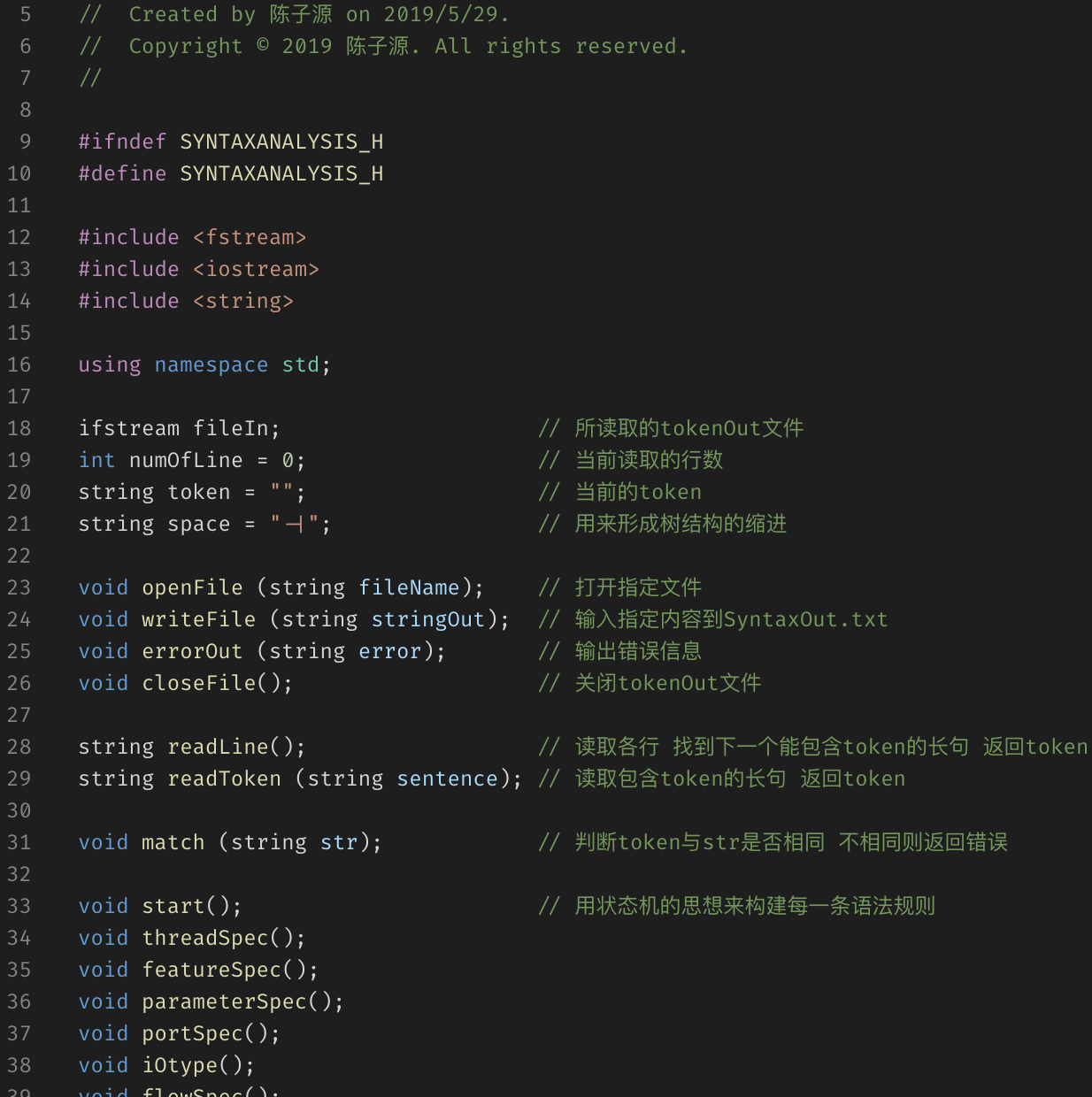
分别是使用g++编译器编译生成Unix可执行文件SyntaxAnalysis，并且**运行该可执行文件，后面需要附上所要执行的文件名称作为main函数的参数，才可对所选文件进行词法分析，如果没有参数，则程序会运行错误。**

可以在命令行看到有输出，并且在文件夹有tokenOut.txt的生成，具体输出和内容会在测试内容中进行展示。

main.cpp的部分截图如下



LexicalAnalysis.h的部分截图如下



可以看待代码的编写过程满足C++编程规范，以及写有相应的注释

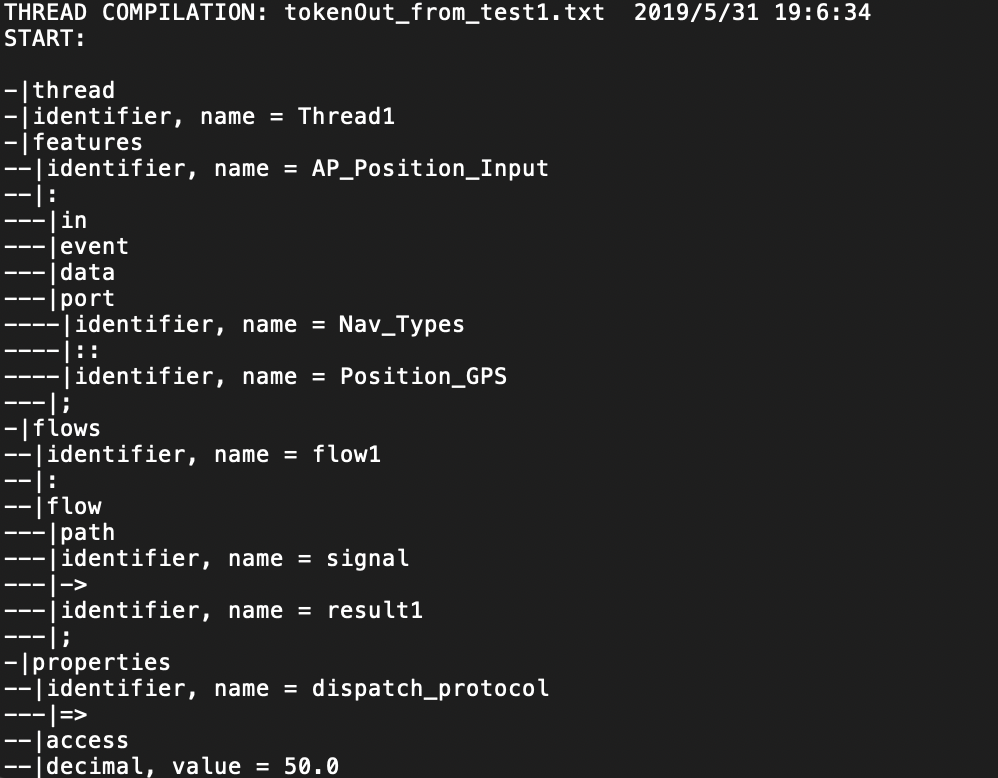
1. 测试内容
2. **实验所提供的测试文件**
3. 使用测试文件test1.txt生成的tokenOut\_from\_test1.txt

命令行所运行代码和输出如下



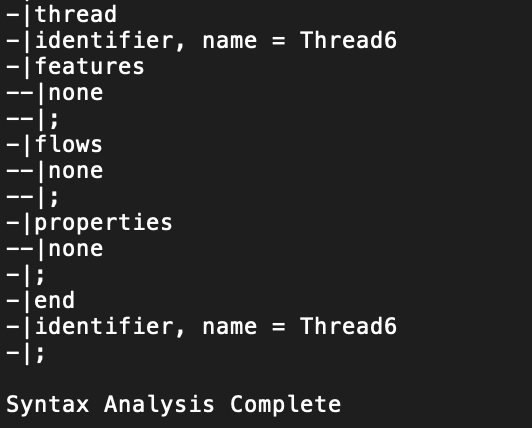
输出内容为所检测到的整个语法分析过程没有遇到错误。

接下来打开所生成的SyntaxOut.txt文件，**具体文件会在“语法分析程序实验提交/附件/测试文件和测试结果/test1”给出**，以下是部分截图。



在文件开头的一二行中，可以看到这是面向线程的语法分析程序，并且有标注所执行的文件名称、词法分析的时间和START的文件开始读取标志，意思是在这之后就是词法分析的结果。

出于对所提供的语法分析程序输出模板阅读时所发生的缩进识别困难的问题，在此语法程序中用符号进行了更好的缩进识别。可以看到，在语法分析结果中，用“-”符号来表示树结构的缩进，用“|”符号来表示，自该符号起，后面皆是树节点的内容。



在这个小截图里面，有一个完整的thread树的内容，可以看到已经实现了对树结构的搭建、identifier信息的保留、保留字的识别和符号的识别，实现了实验所要求的语法分析基本功能。

在SyntaxOut.txt文件的最后，能够看到象征着语法分析结束并且没有错误的的“Syntax Analysis Complete”句样。

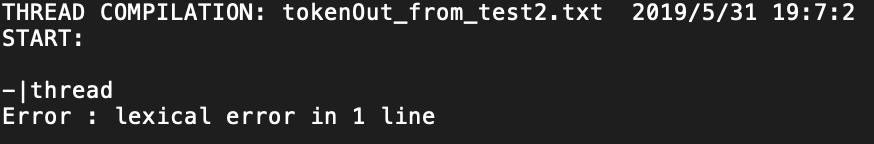
1. 使用测试文件test2.txt生成的tokenOut\_from\_test2.txt

命令行所运行代码和输出如下



可以看到，输出内容显示，在tokenOut\_from\_test2.txt文件中检测到词法分析的错误。错误内容是，在词法分析test2.txt的时候，在第1行就出现了词法分析的错误。

在语法分析程序中设定的逻辑与词法分析中的不同，词法分析会分析完所有的文本，输出全部错误。而语法分析程序则是遇到错误则马上报错并退出程序。



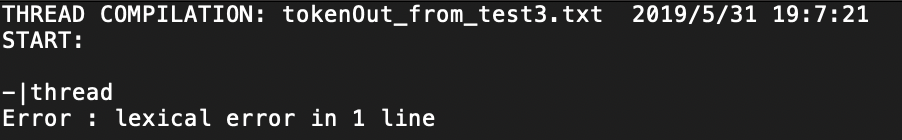
可以看到，在文件一开始，依然是执行文件名称及程序运行的时间。在下面的内容中，我们可以看到第一个被执行的树节点，但是在识别下一个token的时候，遇到了词法分析的错误，因而停止了程序的执行。

1. 使用测试文件test3.txt生成的tokenOut\_from\_test3.txt

命令行所运行代码和输出如下



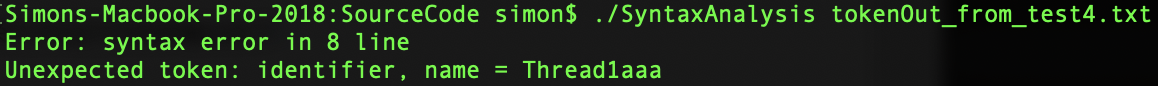
输出显示，在tokenOut\_from\_test3.txt文件中检测到词法分析的错误。



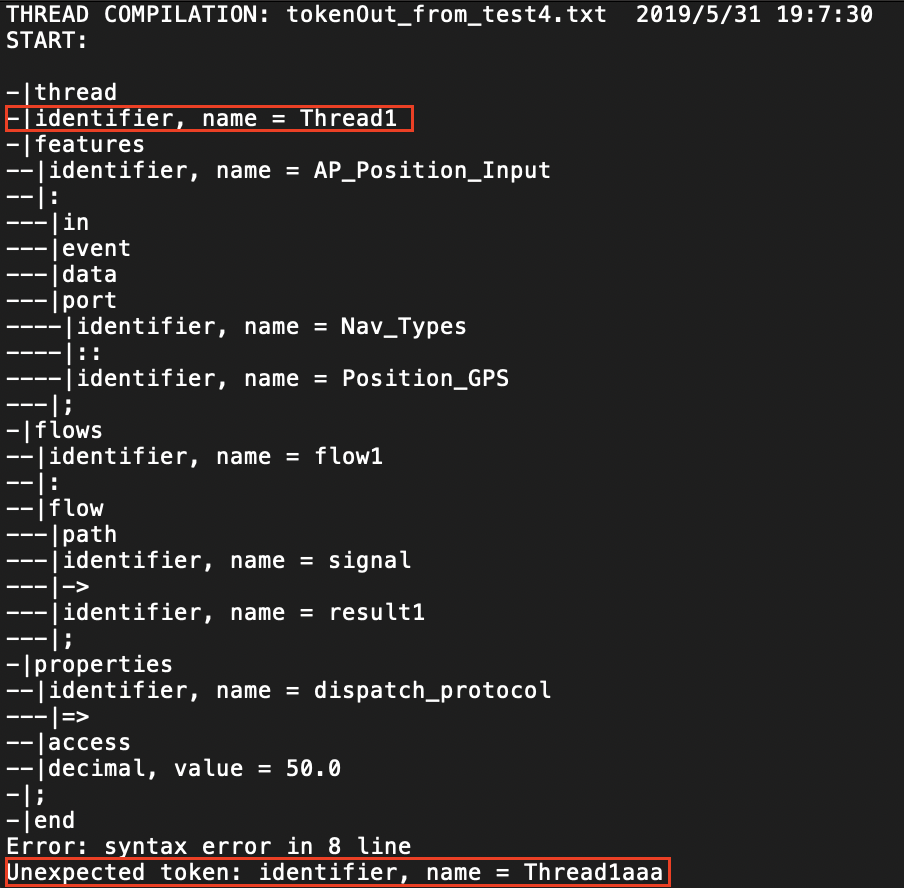
以上出现的错误与tokenOut\_from\_test2.txt中的错误一致，此处不作赘述。

1. 使用测试文件test4.txt生成的tokenOut\_from\_test4.txt

命令行所运行代码和输出如下



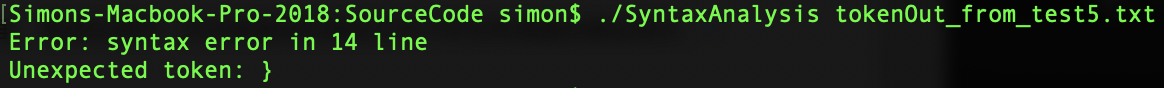
输出显示，在tokenOut\_from\_test4.txt文件中检测到语法分析的错误，错误内容出现在text4.txt也就是程序所提供的测试原文的第8行，错误内容是出现了不应该出现的token，“identifier, name = Thread1aaa”



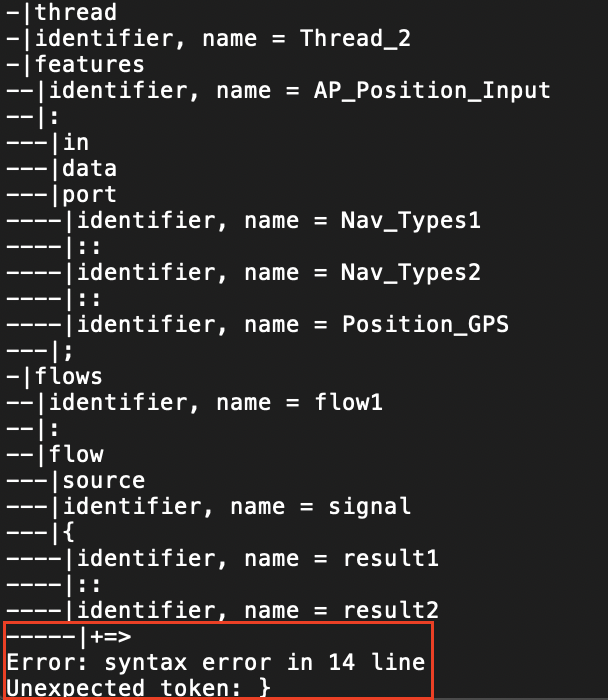
通过查看所输出的SyntaxOut可以看到，错误原因是该thread在结束的时候，end后面所接的identifier的命名与一开始的命名不一致，因此出现了语法错误

1. 使用测试文件test5.txt生成的tokenOut\_from\_test5.txt

命令行所运行代码和输出如下



输出显示，在tokenOut\_from\_test4.txt文件中检测到语法分析的错误，错误内容出现在text5.txt也就是程序所提供的测试原文的第14行，错误内容是出现了不应该出现的token，“}”



在语法分析的规则中，+=>属于splitter的内容，而splitter后面应该接constant或者access这两个关键字其中之一，不能直接接“}”符号。

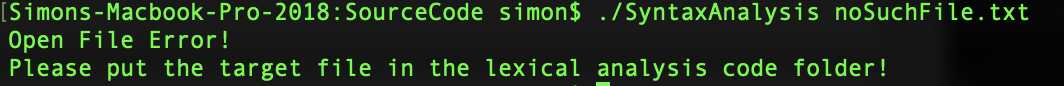
自此，实验所提供的五个测试文件已经执行完毕，并都实现了词法分析的功能，通过了测试。

1. **自身设计的测试项目**

在这里，笔者会提供一些实验提供的测试文件中所没有涵盖的测试项目。

测试一：不存在所要执行的文件

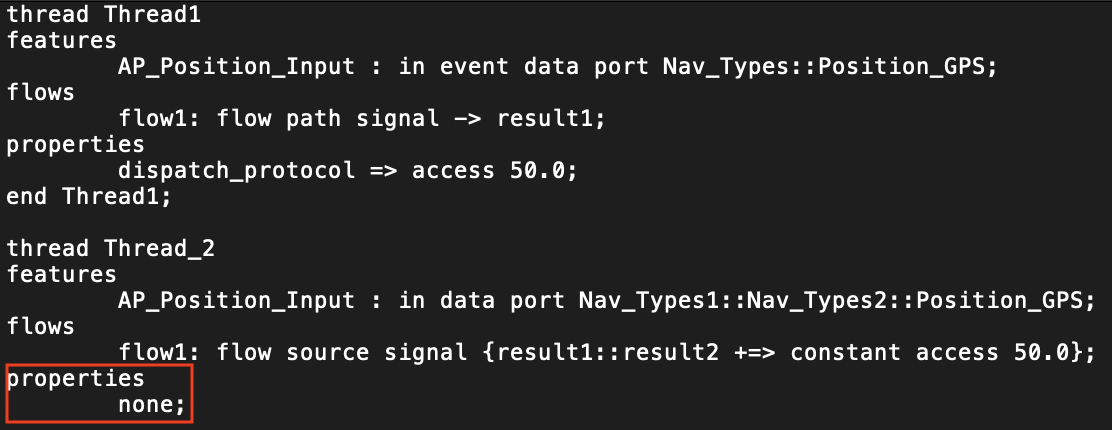
命令行所运行代码和输出如下



命令行直接输出报错，并给出解决方案，观察此时的Source Code文件夹，此时没有SyntaxOut.txt生成，测试通过。

测试二：额外的补充测试

制作test6.txt文件如下



可以看到，在文件的最后，是“none;”结尾的，不符合语法规则。

生成相应的tokenOut\_from\_test6.txt文件，并通过词法分析。



命令行所运行代码和输出如下



出现了一个新的错误类型，错误发生在test6.txt的第16行中，错误内容是在该行，本不应该是语法分析的终止位置，但是下面再也没有输入了，因此出现错误。

因此，该语法分析程序通过所有自制测试。

**六个测试文件、其所生成的tokenOut\_from\_testX.txt文件和其所生成的六个SyntaxOut.txt文件，会在“语法分析程序实验提交/附件/测试文件和测试结果/testX”给出，X有1、2、3、4、5和6，意味着不同的测试用例。**