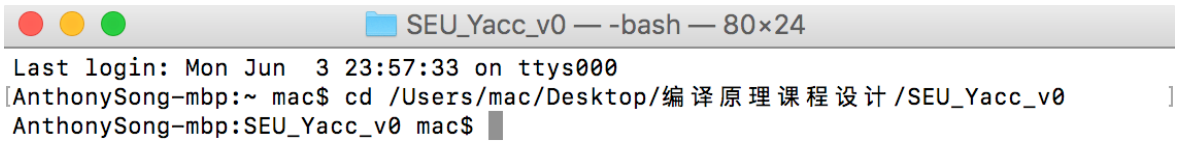
SEU_Yacc的执行步骤如下：

1. 打开终端，切换当前工作目录。



2. 输入 `cmake.` `make all`，编译项目。

```
AnthonySong-mbp:SEU_Yacc_v0 mac$ make all
[ 50%] Building CXX object CMakeFiles/SeuYacc.dir/SEU_Yacc.cpp.o
[100%] Linking CXX executable SeuYacc
[100%] Built target SeuYacc
AnthonySong-mbp:SEU_Yacc_v0 mac$
```

3. 输入 `./Seu_Yacc yacc.y`，根据 `yacc.y` 文件，产生语法分析器**SEU_Yacc**的C++代码文件 `yyparse.cpp`。

```
AnthonySong-mbp:SEU_Yacc_v0 mac$ ./SeuYacc yacc.y

------------Parse yacc.y.........done

------------calculate First set.........done

------------Construct LR(1) DFA.........done

------------Construct LALR DFA.........done

------------Construct LALR Parsing Table........done

------------Generate Code for SEU_Yacc........done
```

4. 输入 `g++ -std=c++11 yyparse.cpp -o Parser`，编译生成语法分析器**SEU_Yacc**的可执行文件。

```
AnthonySong-mbp:SEU_Yacc_v0 mac$ g++ -std=c++11 yyparse.cpp -o Parser
AnthonySong-mbp:SEU_Yacc_v0 mac$
```

5. 输入 `./Parser Lex_Tokens.txt`，对测试代码文件经过**SEU_Lex**得到的Token序列进行语法分析，测试**SEU_Yacc**的表现。终端可以打印出AST的结构，并且利用graphviz生成可视化AST的PDF。

```
AST:
+- translation_unit
   +- translation_unit
   |  +- translation_unit
   |  |  +- translation_unit
   |  |  |  +- translation_unit
   |  |  |  |  +- translation_unit
   |  |  |  |  |  +- translation_unit
   |  |  |  |  |  |  +- external_declaration
   |  |  |  |  |  |     +- declaration
   |  |  |  |  |  |        +- declaration_specifiers
   |  |  |  |  |  |        |  +- type_specifier
   |  |  |  |  |  |        |     +- INT   int
   |  |  |  |  |  |        +- init_declarator_list
   |  |  |  |  |  |        |  +- init_declarator_list
   |  |  |  |  |  |        |  |  +- init_declarator_list
   |  |  |  |  |  |        |  |  |  +- init_declarator_list
   |  |  |  |  |  |        |  |  |  |  +- init_declarator
   |  |  |  |  |  |        |  |  |  |     +- declarator
   |  |  |  |  |  |        |  |  |  |     |  +- direct_declarator
   |  |  |  |  |  |        |  |  |  |     |     +- IDENTIFIER   t
   |  |  |  |  |  |        |  |  |  |     +- =   =
   |  |  |  |  |  |        |  |  |  |     +- initializer
   |  |  |  |  |  |        |  |  |  |        +- assignment_expression
   |  |  |  |  |  |        |  |  |  |           +- conditional_expression
   |  |  |  |  |  |        |  |  |  |              +- logical_or_expression
   |  |  |  |  |  |        |  |  |  |                 +- logical_and_expression
   |  |  |  |  |  |        |  |  |  |                    +- inclusive_or_expression
   |  |  |  |  |  |        |  |  |  |                       +- exclusive_or_expression
   |  |  |  |  |  |        |  |  |  |                          +- and_expression
   |  |  |  |  |  |        |  |  |  |                             +- equality_expression
   |  |  |  |  |  |        |  |  |  |                                +- relational_expression
   |  |  |  |  |  |        |  |  |  |                                   +- shift_expression
   |  |  |  |  |  |        |  |  |  |                                      +- additive_expression
   |  |  |  |  |  |        |  |  |  |                                         +- multiplicative_expression
   |  |  |  |  |  |        |  |  |  |                                            +- cast_expression
   |  |  |  |  |  |        |  |  |  |                                               +- unary_expression
   |  |  |  |  |  |        |  |  |  |                                                  +- postfix_expression
   |  |  |  |  |  |        |  |  |  |                                                     +- primary_expression
   |  |  |  |  |  |        |  |  |  |                                                        +- CONSTANT   0
   |  |  |  |  |  |        |  |  |  +- ,   ,
   |  |  |  |  |  |        |  |  |  +- init_declarator
   |  |  |  |  |  |        |  |  |     +- declarator
   |  |  |  |  |  |        |  |  |     |  +- direct_declarator
   |  |  |  |  |  |        |  |  |     |     +- IDENTIFIER   j
   |  |  |  |  |  |        |  |  |     +- =   =
   |  |  |  |  |  |        |  |  |     +- initializer
   |  |  |  |  |  |        |  |  |        +- assignment_expression
   |  |  |  |  |  |        |  |  |           +- conditional_expression
   |  |  |  |  |  |        |  |  |              +- logical_or_expression
   |  |  |  |  |  |        |  |  |                 +- logical_and_expression
   |  |  |  |  |  |        |  |  |                    +- inclusive_or_expression
   |  |  |  |  |  |        |  |  |                       +- exclusive_or_expression
   |  |  |  |  |  |        |  |  |                          +- and_expression
   |  |  |  |  |  |        |  |  |                             +- equality_expression
   |  |  |  |  |  |        |  |  |                                +- relational_expression
```