中山大學

SUN YAT-SEN UNIVERSITY

编译原理
Project 1

(2022 - 2023 学年春季学期)

课程名称	编译原理
学院	计算机学院
授课教师	张献伟
学生姓名	唐喆
学号	20337111
专业	计算机科学与技术
Email	tangzh_33@163.com
完成时间	2023 年 03 月 10 日

School of Computer Science and Engineering, Sun Yat-sen University

编译原理 Project 1

20337111 唐喆

2023 年 03 月 10 日

目录

1 实验简介

本次实验需在SYsU-lang实现的代码框架下，基于 Flex 提供的接口完善编译器的词法分析器 (sysu-lexer)。sysu-lexer 的输入是一个预处理过的文件里面的字符序列，输出是与 clang -cc1 -dump-tokens 2>&1 相当的内容。输出的每一行格式如下：

- （一个）有效 token。
- （一个）由单引号包裹的匹配串。
- （若干）识别上一个有效 token 到这个有效 token 间遇到的无效 token。
- （一个）有效 token 的位置。
- 例子：

```
int 'int'	[StartOfLine] Loc=<tester/functional/000_main.sysu.c:1:1>
```

本报告主要记录了本次实验中我遇到的几个问题，同时完善了 Wiki 中提供的 token 表，最后展示我的实验的结果。

2 要点记录

参照 Wiki 上提供的完善文档，本次实验的完成思路主要分为以下三步：

- 学习 Flex 的基本语法，了解 Flex 的接口；
- 模仿已提供的代码，编写正则表达式，完善 token 表；
- 归纳冗余字符模式，参照Clang 文档，添加冗余字符的输出。

整体上来说，完成实验一的难度不会太高。本部分主要记录我遇到的几个小问题以及对应的解决方法，同时将完善后的 token 表展示如下。

2.1 正则表达式书写

本部分中主要记录了正则表达式书写可能会遇到的一些问题：

1. 仅匹配空格：使用 *[<Space>]* 即可（Flex 解析引擎下），其中 <Space> 指的就是键盘上输入的空格；

2. 匹配空白字符：使用 *[\f \r \t \v]* 即可（Flex 解析引擎下），注意最后有一个空格，以匹配空格；

3. 规则非常复杂时，使用圆括号”()” 进行分隔，如对于 16 进制的浮点表达，我们可以使用：*0x([0-9a-fA-F]+?[0-9a-fA-F]*|[0-9a-fA-F]*?[0-9a-fA-F]+)([pP][+-]?[0-9]+)?*；使用圆括号可以分隔不同子串，在这里我们分割了不同形式的 16 进制浮点数表达子串；

4. 匹配字符串：我们希望贪婪地匹配尽可能长的字符串，又不希望在字符串连续声明，且逗号分隔的前提下将两个字符串错误匹配为一个，这时我们可以使用排他性定义：*"[^,\n]*"*。

5. 最后，在正式编译前可以使用在线引擎（如引擎参考 1等）先测试正则表达式是否书写正确，节省时间。

以上即遇到的问题以及对应的解决方法。

2.2 冗余字符格式化输出

参照Clang 文档，总共会有 12 种情况。在这里我使用了一个 int 来记录各类 Flag，从低位开始编码：

```
enum TokenFlags {
  StartOfLine = 0x01 , LeadingSpace = 0x02 , DisableExpand = 0x04 , NeedsCleaning
      = 0x08 ,
  LeadingEmptyMacro = 0x10 , HasUDSuffix = 0x20 , HasUCN = 0x40 , IgnoredComma =
      0x80 ,
  StringifiedInMacro = 0x100 , CommaAfterElided = 0x200 , IsEditorPlaceholder =
      0x400 , IsReinjected = 0x800
};
```

对应的输出函数如下：

```
std::string yytokenFlag[] = {
  "StartOfLine", "LeadingSpace", "DisableExpand", "NeedsCleaning",
  "LeadingEmptyMacro", "HasUDSuffix", "HasUCN", "IgnoredComma",
  "StringifiedInMacro", "CommaAfterElided", "IsEditorPlaceholder", " IsReinjected"
};
void yyGenerateTokenFlags(std::string & yy_token_flag, int yycolumn, int yyleng,
    int yy_space, int& yy_first) {
  int yy_tag = 0, yy_curr_loc = yycolumn - yyleng;
  if(yy_space > 0 && yy_curr_loc - yy_space == 1)
    yy_tag |= LeadingSpace;
  if(yy_first == 0)
  {
    yy_tag |= StartOfLine;
    yy_first = 1;
  }
```

```
  for (int i = 0; i < 12; ++i)
    if (yy_tag & (1 << i))
      yy_token_flag += " [" + yytokenFlag[i] + "]";
}
```

2.3 完善后的分词表

最后，在提供的词表基础上，我继续完善了部分未提及的 token，同时将其整理如下：

Token	Name	Comment
\	comma	
;	semi	
[	l_square	
]	r_square	
{	l_brace	
}	r_brace	
(	l_paren	
)	r_paren	
=	equal	
+	plus	
-	minus	
!	exclaim	
*	star	
/	slash	
%	percent	
<	less	
>	greater	
<=	lessequal	
>=	greaterequal	
==	equalequal	
!=	exclaimequal	
&&	ampamp	
||	pipepipe	
int	int	
float	float	
char	char	
void	void	
const	const	
if	if	
else	else	
do	do	
while	while	
break	break	
continue	continue	
return	return	
intconst	numeric_constant	Include the floating point expression in either decimal or hexadecimal form.(e or p)
string	string_literal	Streams begin and end with "
ident	identifier	
<<EOF>>	eof	

图 1: 整理后的 Tokens 表

3 结果展示

根据 Wiki 上提供的指令，我简单整合了一个编译-测试脚本：

```
# 编译安装
# `${CMAKE_C_COMPILER}` 仅用于编译 `.sysu.c`
# 非 SYsU 语言的代码都将直接/间接使用 `${CMAKE_CXX_COMPILER}` 编译（后缀为 `.cc`）
rm -rf $HOME/sysu
cmake -G Ninja \
  -DCMAKE_C_COMPILER=clang \
  -DCMAKE_CXX_COMPILER=clang++ \
  -DCMAKE_INSTALL_PREFIX=$HOME/sysu \
  -DCMAKE_MODULE_PATH=$(llvm-config --cmakedir) \
  -DCPACK_SOURCE_IGNORE_FILES=".git/;tester/third_party/" \
  -B $HOME/sysu/build
cmake --build $HOME/sysu/build
cmake --build $HOME/sysu/build -t install

set -x
for i in {1..3}; do
  # 测试
export PATH=$HOME/sysu/bin:$PATH \
  CPATH=$HOME/sysu/include:$CPATH \
  LIBRARY_PATH=$HOME/sysu/lib:$LIBRARY_PATH \
  LD_LIBRARY_PATH=$HOME/sysu/lib:$LD_LIBRARY_PATH &&
  sysu-compiler --unittest=lexer-${i} "**/*.sysu.c"
done
```

三个测试用例截图如下。由于三个测试用例输出类似，我使用”history | tail -n 3” 来展示对应的测试项目结果。可以看到，我们顺利通过了测例 ***Lexer 1 ∼ 3***，具体如下：

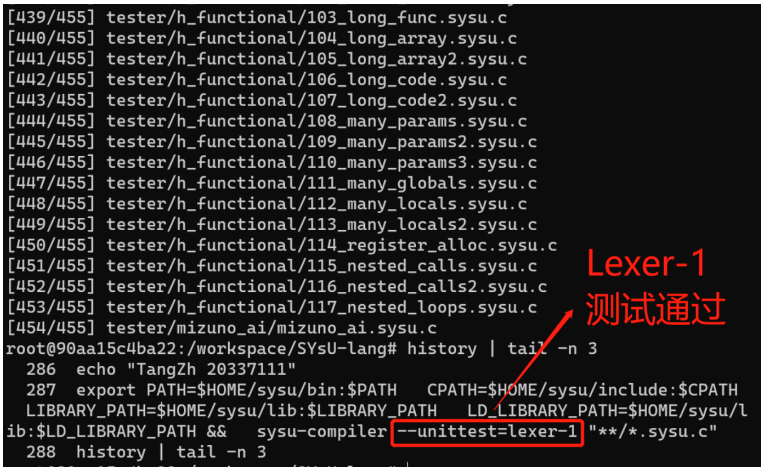

图 2: Lexer-1 测试截图

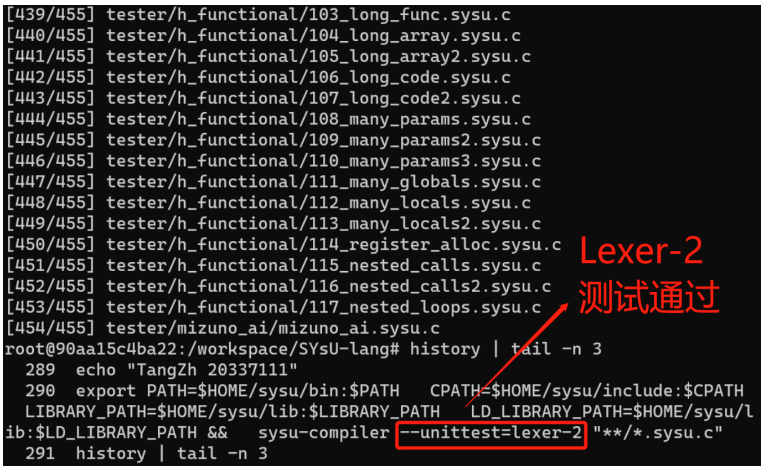

图 3: Lexer-2 测试截图

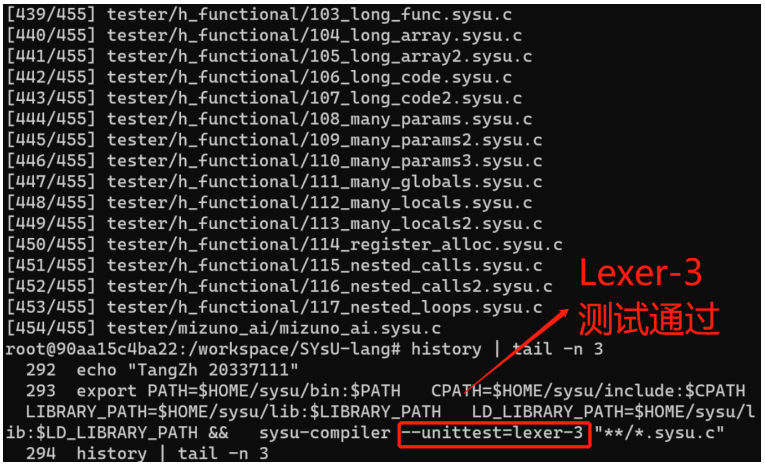

图 4: Lexer-3 测试截图

最后，我们也顺利通过了测试指令 *CTEST_OUTPUT_ON_FAILURE=1 cmake –build $HOME/sysu/build -t test*:

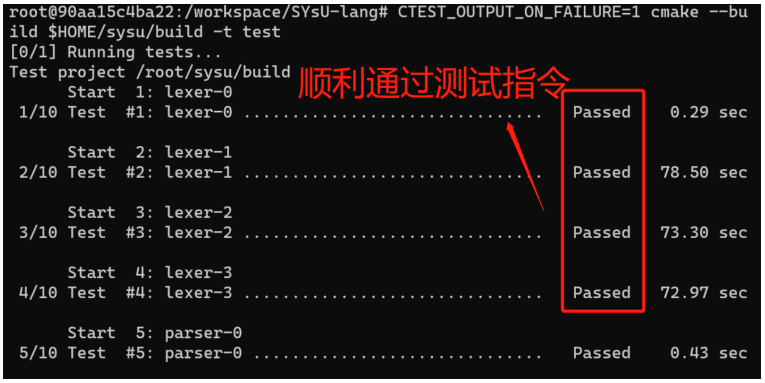

图 5: 各实验得分测试