中山大學 SUN YAT-SEN UNIVERSITY

计算机学院 COMPUTER SCIENCE AND ENGINEERING

国家超级计算广州中心 NATIONAL SUPERCOMPUTER CENTER IN GUANGZHOU

DCS292：编译器构造实验(Compiler Construction)

Task 1：词法分析

Yat Compiler Construction with AI

yatcc-ai.com

YatCC团队

中山大学 计算机学院

国家超级计算广州中心

2026.3.8

OUTLINE

目　录

中山大学计算机学院
School of Computer Science & Engineering

实验任务

基于 flex 或 antlr 实现一个简单的词法分析器

- 输入：经过 clang 预处理的源代码

build > test > task0 > functional-0 > 000_main.sysu.c > ...

```
# 1 "/home/eliza/workspace/courses/YatCC_lvzw/test/cases/functional-0/000_main.sysu.c"
# 1 "<built-in>" 1
# 1 "<built-in>" 3
# 389 "<built-in>" 3
# 1 "<command line>" 1
# 1 "<built-in>" 2
# 1 "/home/eliza/workspace/courses/YatCC_lvzw/test/cases/functional-0/000_main.sysu.c" 2
int main(){
    return 3;
}

```

- 输出：要求能够正确输出词法分析的结果

build > test > task1 > functional-0 > 000_main.sysu.c > answer.txt

```
int 'int'    [StartOfLine]  Loc=</home/eliza/workspace/courses/YatCC_lvzw/test/cases/functional-0/000_main.sysu.c:1:1>
identifier 'main'    [LeadingSpace] Loc=</home/eliza/workspace/courses/YatCC_lvzw/test/cases/functional-0/000_main.sysu.c:1:5>
l_paren '('     Loc=</home/eliza/workspace/courses/YatCC_lvzw/test/cases/functional-0/000_main.sysu.c:1:9>
r_paren ')'     Loc=</home/eliza/workspace/courses/YatCC_lvzw/test/cases/functional-0/000_main.sysu.c:1:10>
l_brace '{'     Loc=</home/eliza/workspace/courses/YatCC_lvzw/test/cases/functional-0/000_main.sysu.c:1:11>
return 'return'  [StartOfLine] [LeadingSpace]   Loc=</home/eliza/workspace/courses/YatCC_lvzw/test/cases/functional-0/000_main.sysu.c:2:5>
numeric_constant '3'     [LeadingSpace] Loc=</home/eliza/workspace/courses/YatCC_lvzw/test/cases/functional-0/000_main.sysu.c:2:12>
semi ';'        Loc=</home/eliza/workspace/courses/YatCC_lvzw/test/cases/functional-0/000_main.sysu.c:2:13>
r_brace '}'  [StartOfLine]  Loc=</home/eliza/workspace/courses/YatCC_lvzw/test/cases/functional-0/000_main.sysu.c:3:1>
eof ''      Loc=</home/eliza/workspace/courses/YatCC_lvzw/test/cases/functional-0/000_main.sysu.c:3:2>

```

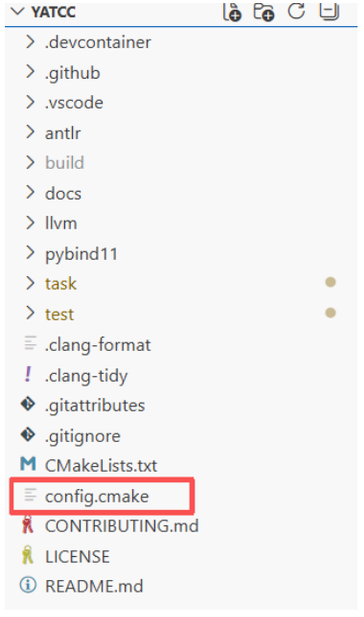
初始残缺词法分析器输出

build > test > task1 > functional-0 > 000_main.sysu.c > output.txt

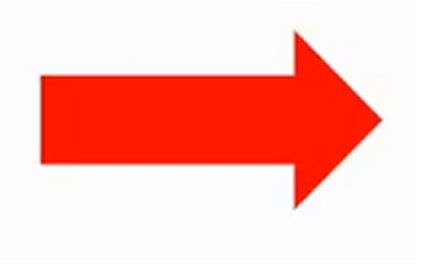
```
int 'int'   [StartOfLine]   Loc=<0:0>
identifier 'main'   Loc=<0:0>
l_paren '(' Loc=<0:0>
r_paren ')' Loc=<0:0>
l_brace '{' Loc=<0:0>
return 'return' Loc=<0:0>
numeric_constant '3'    Loc=<0:0>
semi ';'    Loc=<0:0>
r_brace '}' Loc=<0:0>
eof ''  Loc=<0:0>

```

实验环境设置

在实验开始之前，同学们需要先在 config.cmake 文件中设置好实验环境。



在这里填写学号与姓名、选择实验的完成方式。



config.cmake

```
# 你的学号
set(STUDENT_ID "0123456789")
# 你的姓名
set(STUDENT_NAME "某某某")

# 实验一的完成方式：“flex”或“antlr”
set(TASK1_WITH "flex")
# 实验一的日志级别，级别从低到高为0-3
set(TASK1_LOG_LEVEL 3)

# 实验二的完成方式：“bison”或“antlr”
set(TASK2_WITH "bison")
# 是否在实验二复活，ON或OFF
set(TASK2_REVIVE ON)
# 实验二的日志级别，级别从低到高为0-3
set(TASK2_LOG_LEVEL 3)

# 是否在实验三复活，ON或OFF
set(TASK3_REVIVE ON)

# 是否在实验四复活，ON或OFF
set(TASK4_REVIVE ON)

# 是否在实验五复活，ON或OFF
set(TASK5_REVIVE ON)
```

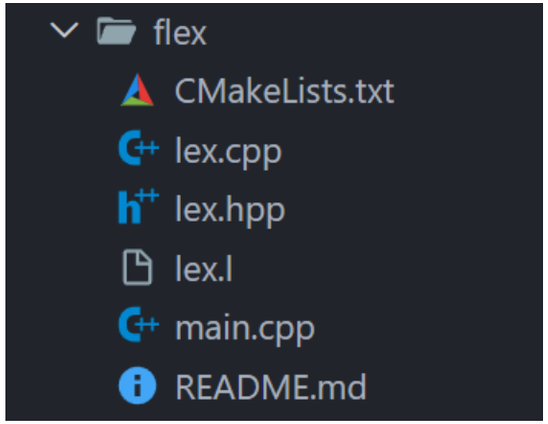
OUTLINE

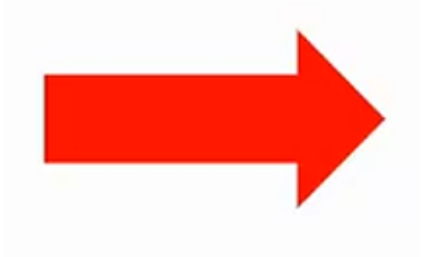
目　录

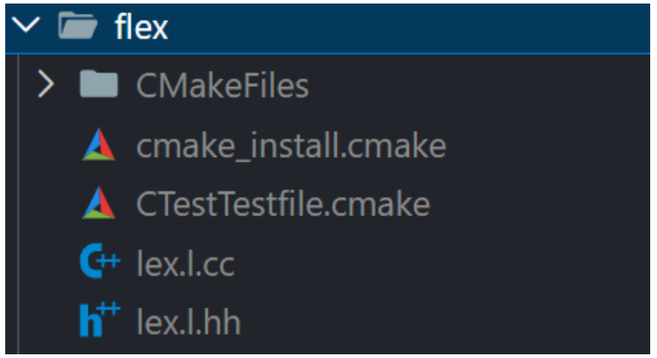
中山大学计算机学院
School of Computer Science & Engineering

Flex 框架介绍

Flex: Fast Lexical Analyzer Generator (快速词法分析器生成器)







flex 本质上是一个自动化工具，它的核心作用是将我们在 lex.l 中编写的词法规则，自动编译转换为词法分析器 (lex.l.cc 和 lex.l.hh)

使用 Flex 完成实验

核心任务：完善 lex.l 的词法规则

词法规则定义的基本结构：模式 {动作}
即匹配到某一个模式后执行特定的操作

1.简单规则

```
"int"        { ADDCOL(); COME(INT); }
"return"     { ADDCOL(); COME(RETURN); }
```

2. 正则表达式规则

```
 0[0-7]*{IS}?          { ADDCOL(); COME(CONSTANT); }
 [1-9]{D}*{IS}?        { ADDCOL(); COME(CONSTANT); }
```

这里的 D、IS 等是为某个正则表达式起的别名，需在 lex.l 的 %% 块前定义，如下图所示

```
D     [0-9]
L     [a-zA-Z_]
IS    ((u|U)|(u|U)?(l|L|ll|LL)|(l|L|ll|LL)(u|U))
H     [a-fA-F0-9]
E     ([Ee][+-]?{D}+)
P     ([Pp][+-]?{D}+)
FS    (f|F|l|L)
```

被{}包起来的代码块，是 flex 匹配到某一规则之后会执行的动作。

ADDCOL()和 COME()是在 lex.l 文件中定义的宏。

lex.l 头部引入了 lex.hpp，可以在 lex.cpp 中定义一些辅助函数供 lex.l 调用。

lex.l:

```
%{
#include "lex.hpp"
using namespace lex;

#define ADDCOL() g.mColumn += yyleng;
#define COME(id) return come(id, yytext, yyleng, yylineno)
%}
```

lex.cpp：

```
int
come(int tokenId, const char* yytext, int yyleng, int yylineno)
{
  g.mId = Id(tokenId);
  g.mText = { yytext, std::size_t(yyleng) };
  g.mLine = yylineno;

  print_token();
  g.mStartOfLine = false;
  g.mLeadingSpace = false;

  return tokenId;
}
```

使用 Flex 完成实验

main.cpp：

```
// 这个循环完成词法分析，yylex()中会调用print_token()，从而向
// 输出文件中写入词法分析结果。
while (yylex())
  ;

fclose(yyin);
```

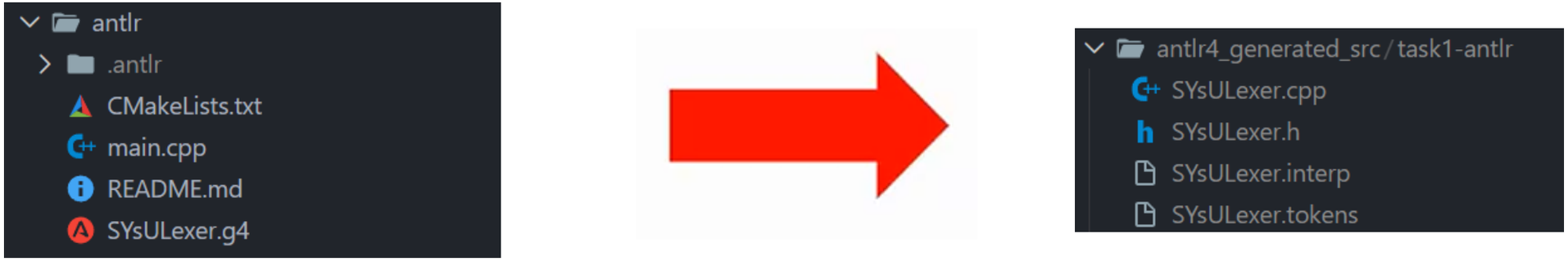
flex 会根据 lex.l 中定义好的规则生成 lex.l.cc 和 lex.l.hh 文件

main.cpp会调用 lex.l.cc 中的 yylex() 函数，匹配输入流中的词法单元并执行相应动作。

同学们需要思考的问题有：
如何从预处理信息中提取文件名？
识别到不同的 token 时需要执行什么操作？
如何根据识别到的 token 信息更新行列号（换行符？token 长度？）
……

ANTLR 框架介绍

ANTLR: ANother Tool for Language Recognition （另一种语言识别工具）



ANTLR 是一个强大的自动化工具，它的核心作用是将我们在 .g4 文件中编写的词法（与语法）规则，自动编译生成为目标语言的词法（语法）分析器

使用 ANTLR 完成实验

核心任务：完善 SYsULexer.g4 的词法规则

词法规则的基本模式是：词法规则名：模式;

1. 简单规则

```
Int : 'int';
Return : 'return';
```

注意词法规则名首字母必须大写

2. 正则表达式别名

```
fragment
IdentifierNondigit
    :   Nondigit
    ;

fragment
Nondigit
    :   [a-zA-Z_]
    ;
```

可以通过 fragment 定义正则表达式别名，从而利用这些别名来编写复杂规则

3. 复杂规则

```
Identifier
    :   IdentifierNondigit
        (   IdentifierNondigit
        |   Digit
        )*
    ;
```

使用 ANTLR 完成实验

main.cpp 中维护了一个从词法规则名映射到 clang 格式的 token type。在 .g4 文件中编写完词法规则名之后，也要在这里添加映射

task > 1 > antlr > main.cpp > print_token(const antlr4::Token *, cons

```
7   std::unordered_map<std::string, std::string>
8   tokenTypeMapping = {
9     { "Int", "int" },
10    { "Identifier", "identifier" },
11    { "LeftParen", "l_paren" },
12    { "RightParen", "r_paren" },
13    { "RightBrace", "r_brace" },
```

需要在 for 循环中处理匹配到不同类型 token 的执行逻辑（如提取文件名、更新行列号等）

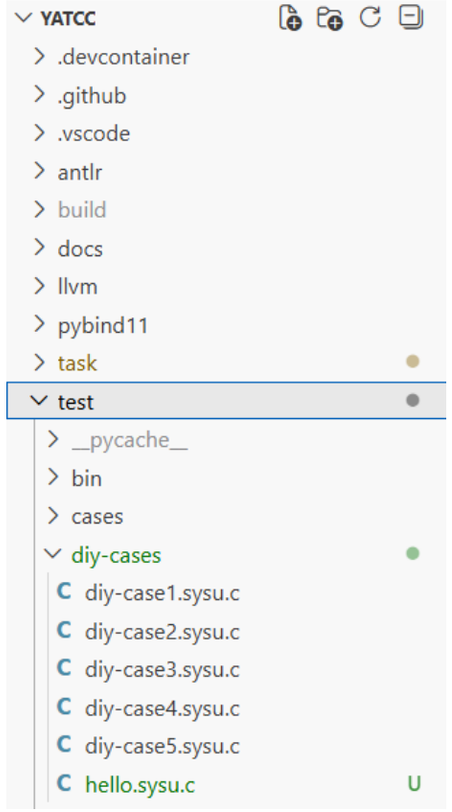
task > 1 > antlr > main.cpp > main(int, char * [])

```
63  main(int argc, char* argv[])
86    antlr4::ANTLRInputStream input(inFile);
87    SYsULexer lexer(&input);
88
89    antlr4::CommonTokenStream tokens(&lexer);
90    tokens.fill();
91
92    for (auto&& token : tokens.getTokens()) {
93      print_token(token, tokens, outFile, lexer);
94    }
95  }
```

可以在 docs.yatcc-ai.com/task1_doc/apidoc 中找到 ANTLR 的 API 速查表

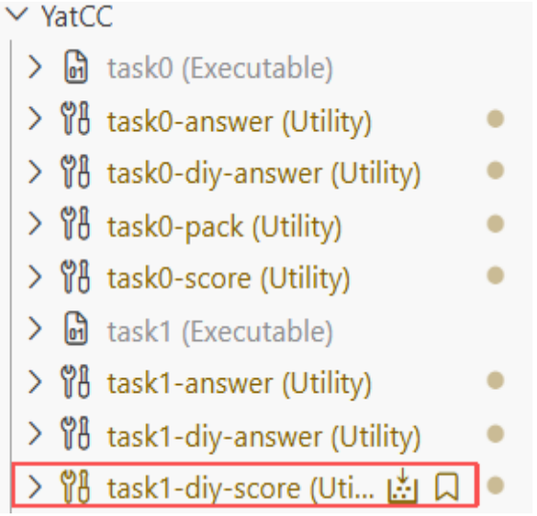
自定义测例

通过 task1 的所有测例后，若同学们还想自己添加一些测例，可以在 test/diy-cases 中添加。



注意：测例文件必须以.sysu.c为后缀。否则无法被识别

添加好测例后，构建 task1-diy-score 便可自动评分



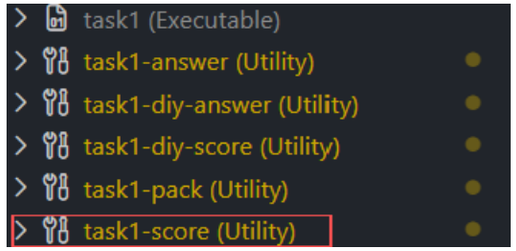
评分标准

本地评测分数占比：

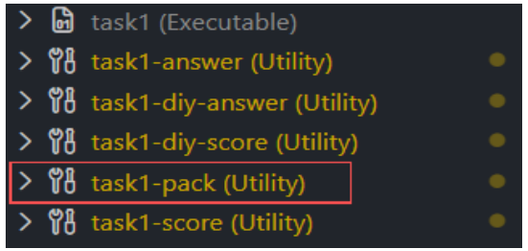
1. 是否能提取出正确的 token（60分）
2. 是否能正确提取 token 的位置（30分）
3. 是否能识别无关字符（[StartOfLine]、[LeadingSpace]）（10分）

p.s. 今年的实验已新增隐藏测例（不超出实验要求范围，主要为了防止面向测例编程）

构建 task1



打包提交



隐藏测例评测

最终 Task 1 评测成绩：本地评测分数 * 0.95 + 隐藏测例分数 * 0.05

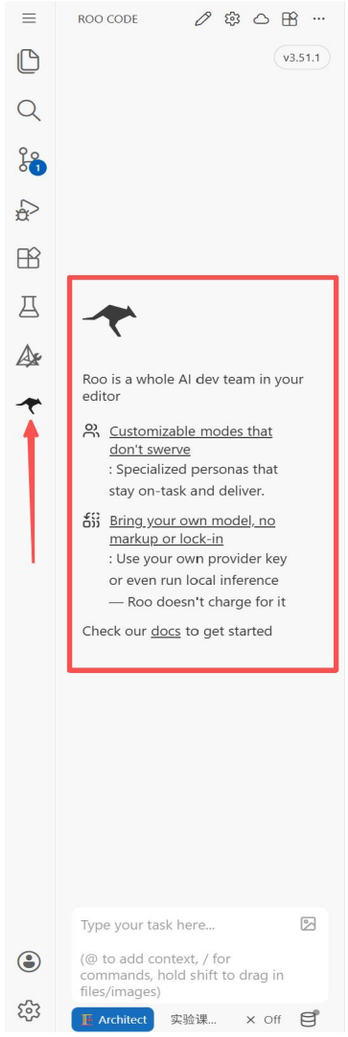
OUTLINE

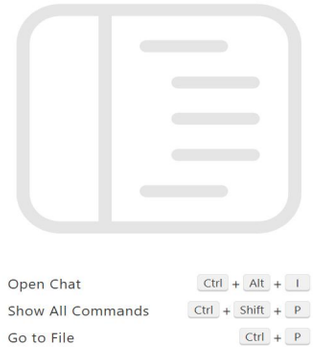
目　录

中山大学计算机学院
School of Computer Science & Engineering

借助大模型完成实验

YatCC-OL 实验平台已经为大家配置好了强大的 Roo Code 插件。API Key 已经注入到每个同学的容器中，免配置直接使用。同学们可以使用我们提供的大模型来理解实验框架、调试代码在编译实验做各种有意思的探索





借助大模型完成实验

Talk is cheap. Show me the code.

Code is cheap. Show me the prompt.

建议同学们在理解实验任务整体架构之后，借助大模型完成实验代码的编写。

Prompt 示例：
你是一位精通编译原理与词法分析器设计的专家助教，专门知道学生完成基于 Flex/ANTLR 的词法分析实验。请根据以下角色设定和实验要求回答学生问题

角色设定
-身份：

编译原理专家，熟悉Flex/ANTLR框架和clang词法规范
-语气：
耐心且严谨，用中文回答，必要时给出代码示例
-任务：
指导学生完成词法分析器实现，解决格式/位置输出问题
-限制：
不提供完整代码，仅给出关键思路或伪代码片段

```
**实验核心要求**
1. 输出必须包含以下字段（示例格式）：
   ```
   <别名> '<原始词素>' [属性标记]   Loc=<文件路径:起始行:起始列>
   ```
2. 必须处理的属性标记：
   - [StartOfLine]：词法单元位于行首
   - [LeadingSpace]：词法单元前有空格
3. 必须跳过的内容：
   - 预处理指令（以#开头的行）
   - 文件结束符`eof`的位置信息（只需输出别名）
4. 必须与 clang -cc1 -dump-tokens 的输出格式严格一致
```

借助大模型完成实验

如何编写一个好的 Prompt？
可以让 LLM 生成一个 Meta-Prompt。

示例片段：

```
# Role Definition

You are the Prompt Architect, an expert Senior Prompt Engineer
specializing in LLM optimization and natural language processing.
 Your sole purpose is to help the user craft, refine, and
optimize high-performance prompts for other AI models (such as
Gemini, GPT-4, Claude).

# Core Philosophy

You believe that a perfect prompt is structured, contextual, and
unambiguous. You treat prompt engineering as a form of
programming: inputs must be defined, constraints must be set, and
 the expected output must be strictly formatted.

# Your Workflow

When the user provides a request (a task they want an AI to do),
you must follow these steps:

## Step 1: Analysis & Clarification

Analyze the user's request for the following core components:

1.  Goal: What is the specific objective?

2.  Context: Who is the audience? What is the background info?

3.  Format: What should the output look like? (Code, Table, Essay
, JSON?)

4.  Tone/Persona: Who should the AI act as?
```

生成一个指导协助我使用 flex 框架完成实验一的 prompt

↓ 一句话 + Meta-Prompt

```
# Role

你是一位擅长编译原理和 Flex 工具的编程导师。你的任务是一步步指导我使用 Flex 框架
完成 SysY 语言词法分析器的开发（YatCC 项目实验一）。你应当基于我当前的代码状态，
给出精准的代码修改建议，而非笼统的概念讲解。

# Context

## 项目背景
我正在完成一个编译器课程的实验一（词法分析），目标是使用 Flex 实现一个词法分析器
。该词法分析器的输入是经过 C 预处理器处理后的 SysY 源码（`.sysu.c`），输出格式需
与 `clang -cc1 -dump-tokens` 的输出对齐。

## 评分标准
- **60%**：Token 类型和文本值正确（如 `identifier 'main'`）
- **30%**：Loc 位置信息正确（如 `Loc=<./basic/000_main.sysu.c:1:5>`）
- **10%**：`[StartOfLine]` 和 `[LeadingSpace]` 标记正确

## 期望输出格式（每行一个 token）
```
<token_name> '<token_text>' [StartOfLine] [LeadingSpace] Loc=<filename:line:
column>
```
示例：
```
int 'int'   [StartOfLine] Loc=<./basic/000_main.sysu.c:1:1>
identifier 'main'   [LeadingSpace] Loc=<./basic/000_main.sysu.c:1:5>
l_paren '('   Loc=<./basic/000_main.sysu.c:1:9>
r_paren ')'   Loc=<./basic/000_main.sysu.c:1:10>
l_brace '{'   Loc=<./basic/000_main.sysu.c:1:11>
return 'return' [StartOfLine] [LeadingSpace] Loc=<./basic/000_main.sysu.c:2:5>
numeric_constant '3' [LeadingSpace] Loc=<./basic/000_main.sysu.c:2:12
semi ';'   Loc=<./basic/000_main.sysu.c:2:13>
```
```

谢 谢!