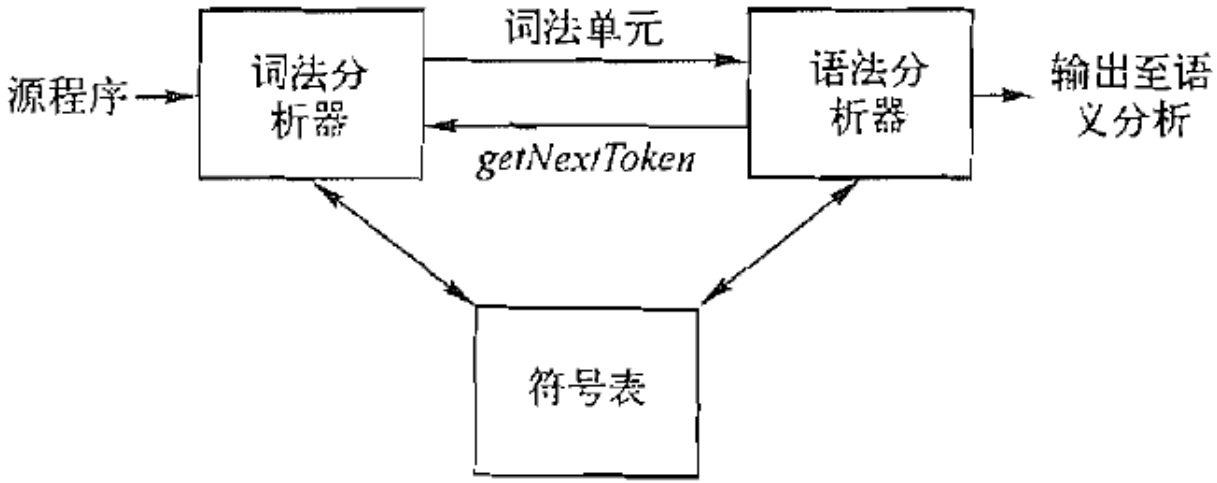
词法分析
(1. 词法分析器生成器 ANTLR v4)

魏恒峰

hfwei@nju.edu.cn

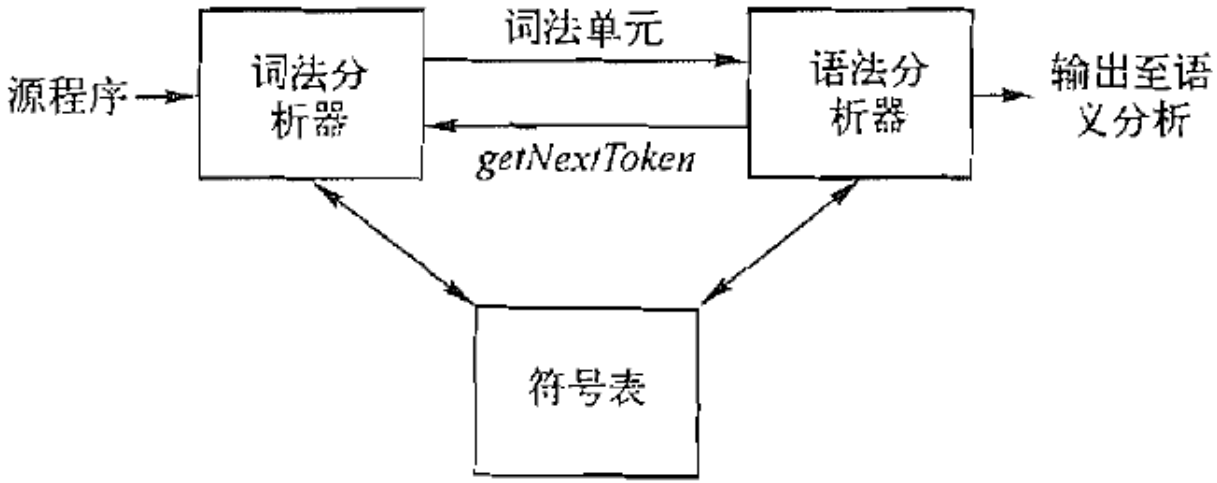
2024 年 03 月 06 日 (周三)

输入： 程序文本/字符串 s (CharStream)



输出： 词法单元流 (TokenStream)

输入： 程序文本/字符串 s (CharStream) + **词法单元 (token) 的规约**



输出： 词法单元流 (TokenStream)

词法分析器的三种设计方法 (由易到难)

词法分析器的三种设计方法 (由易到难)

词法分析器生成器

词法分析器的三种设计方法 (由易到难)

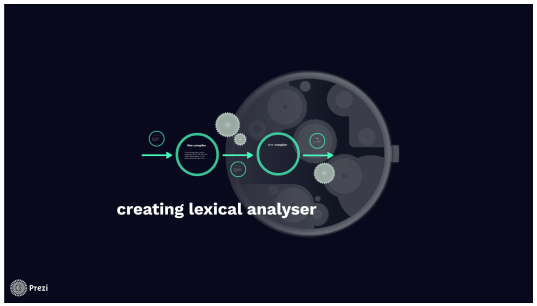
词法分析器生成器

手写词法分析器

词法分析器的三种设计方法 (由易到难)

词法分析器生成器

手写词法分析器



自动化词法分析器

词法分析器的三种设计方法 (由易到难)

词法分析器生成器

手写词法分析器

自动化词法分析器

很多生产环境下的编译器 (如 gcc) 仍选择**手写词法分析器**

gcc / gcc / c-family / c-lex.cc

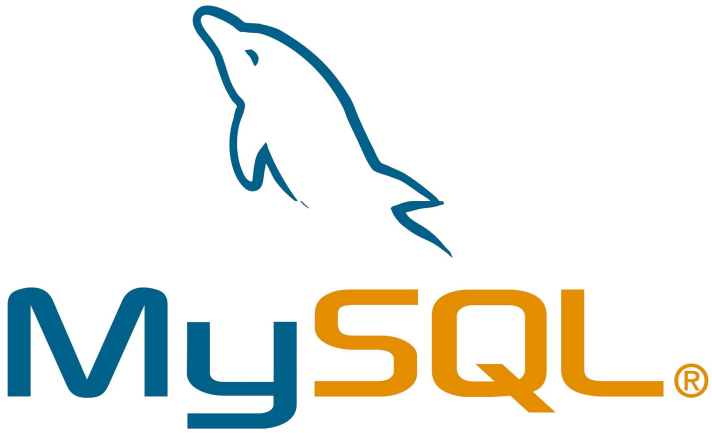
jakubjelinek c: Handle scoped attributes in __has*attrib

Code Blame 1785 lines (1612 loc) · 49.1 KB

gcc / libcpp / lex.cc

jakubjelinek c: Handle scoped attributes in __has*at

Code Blame 5717 lines (5073 loc) · 158 KB



mysql-server / sql / sql_lex.cc

roylyseng and dahlerlend Bug#35889990: Setting

Code Blame 5266 lines (4560 loc) · 170 KB

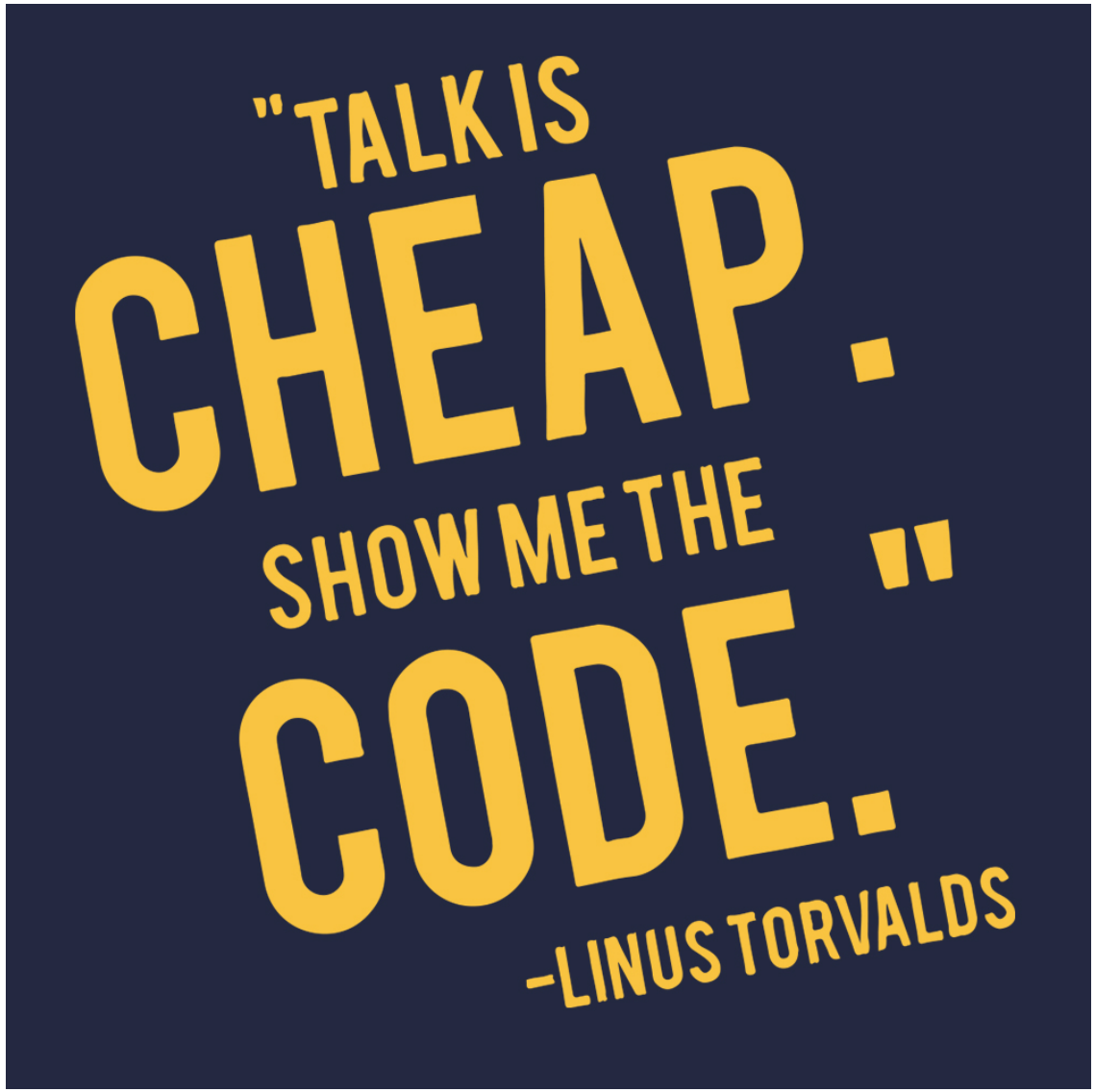
词法分析器生成器

输入： 词法单元的规约

SimpleExpr.g4

输出： 词法分析器

- SimpleExprLexer.java


"TALK IS
CHEAP.
SHOW ME THE
CODE."
-LINUS TORVALDS

命令行式使用 ANTLR v4

Quick Start

To try ANTLR immediately, jump to the *new* ANTLR Lab!

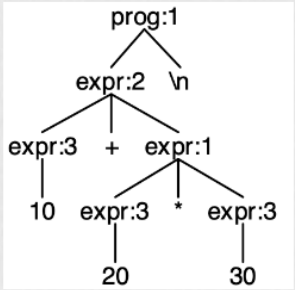
To install locally, use antlr4-tools, which installs Java and ANTLR if needed and creates `antlr4` and `antlr4-parse` executables:

```
$ pip install antlr4-tools
```

(Windows must add . . \LocalCache\local-packages\Python310\Scripts to the PATH). See the Getting Started doc. Paste the following grammar into file `Expr.g4` and, from that directory, run the `antlr4-parse` command. Hit control-D on Unix (or control-Z on Windows) to indicate end-of-input. A window showing the parse tree will appear.

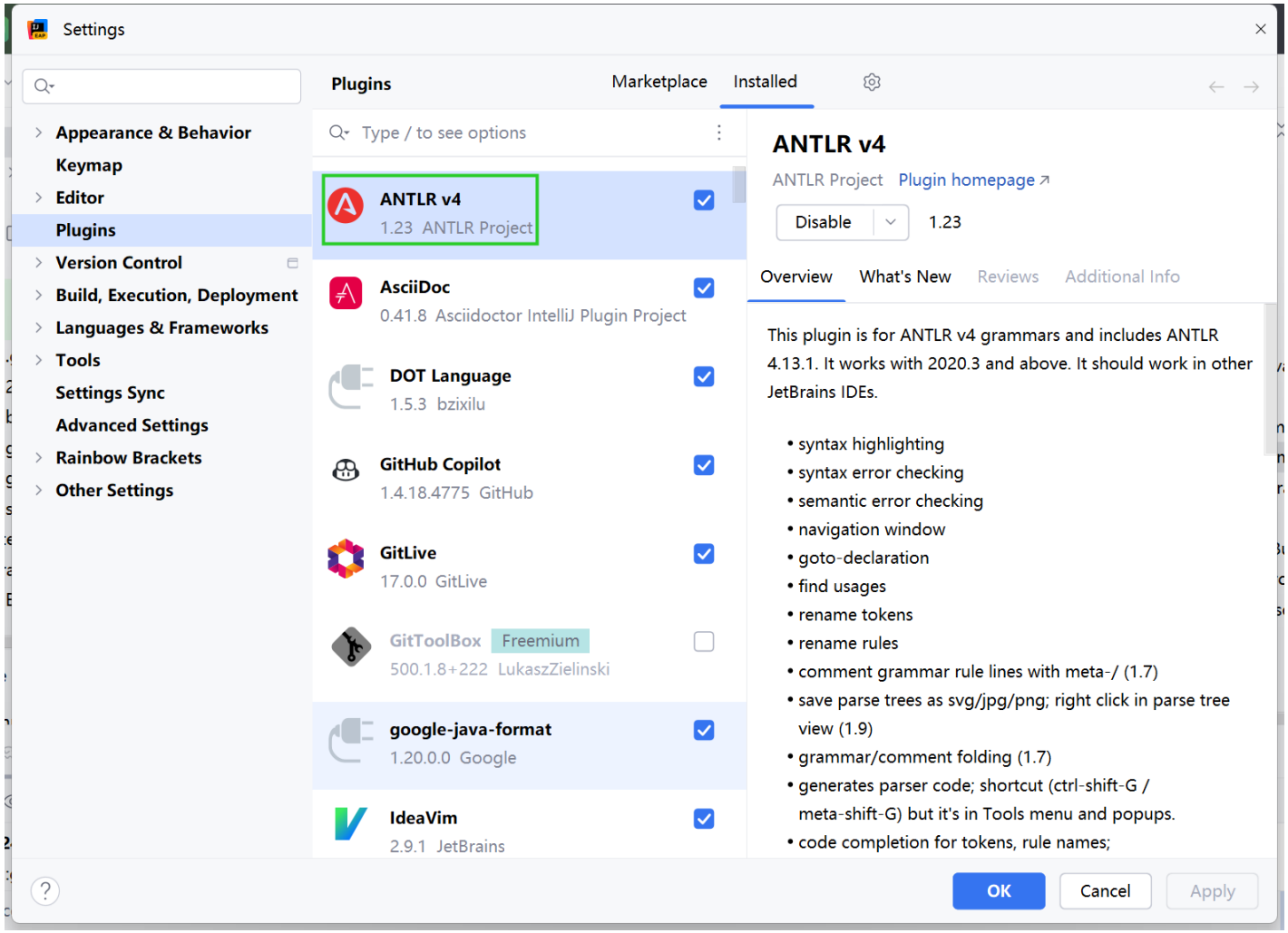
```
grammar Expr;
prog:   (expr NEWLINE)* ;
expr:   expr ('*'|'/') expr
    |   expr ('+'|'-') expr
    |   INT
    |   '(' expr ')'
    ;
NEWLINE : [\r\n]+ ;
INT     : [0-9]+ ;
```

```
$ antlr4-parse Expr.g4 prog -gui
10+20*30
^D
$ antlr4 Expr.g4  # gen code
$ ls ExprParser.java
ExprParser.java
```



https://www.antlr.org/

交互式使用 ANTLR v4



https://www.antlr.org/tools.html

编程式使用 ANTLR v4

https://docs.gradle.org/current/userguide/antlr_plugin.html

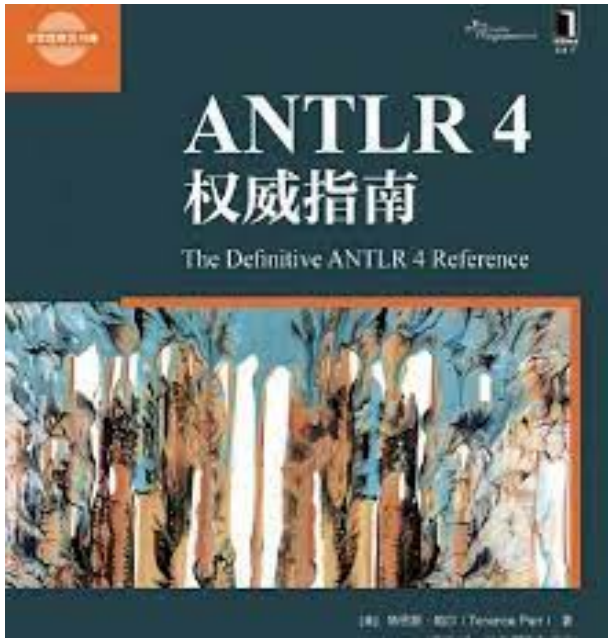
ANTLR v4 中的**冲突解决**规则

最前优先匹配: 关键字 *vs.* 标识符

`ML_COMMENT` *vs.* `DOC_COMMENT`

最长优先匹配: `1.23`, `>=`, `ifhappy`

非贪婪匹配: `()??`, `()*?`, `()+?`



5.5: 识别常见的语法结构

15.5: 词法规则

15.6: 通配符与非贪婪子规则

12: 掌握词法分析的“黑魔法”

以**编程的方式**使用 ANTLR 4 生成的 `xxxLexer.java`

以**编程的方式**使用 ANTLR 4 生成的 `xxxLexer.java`

```
@header {
package simpleexpr;
}
```

以**编程的方式**使用 ANTLR 4 生成的 xxxLexer.java

```
@header {
package simpleexpr;
}
```

```
CharStream input = CharStreams.fromStream(is);
SimpleExprLexer lexer = new SimpleExprLexer(input);

lexer.getAllTokens().forEach(System.out::println);
```

lexer grammar

Section 4.1 (1. 语法导入) of《ANTLR 4 权威指南》

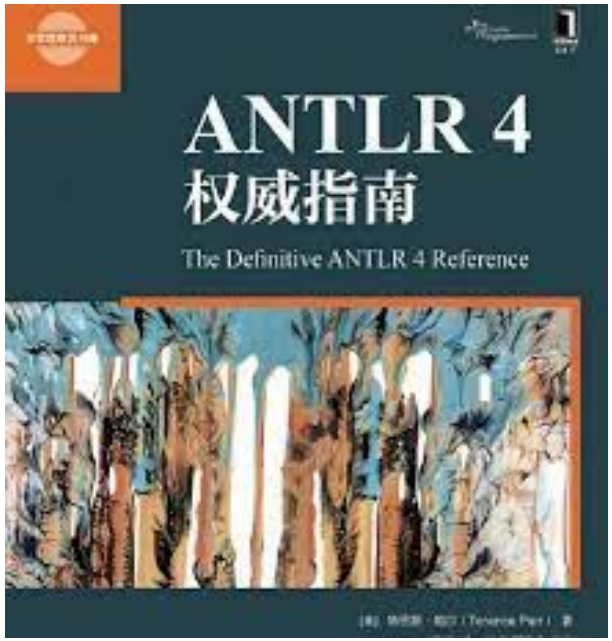
```
lexer grammar SimpleExprLexerRules;

// Comment out the following lines
// Otherwise, there will be duplicate package statements
// @header {
// package simpleexpr;
// }
```

```
grammar SimpleExpr;
import SimpleExprLexerRules;

@header {
package simpleexpr;
}
```

You can learn a lot from grammars-v4/c.



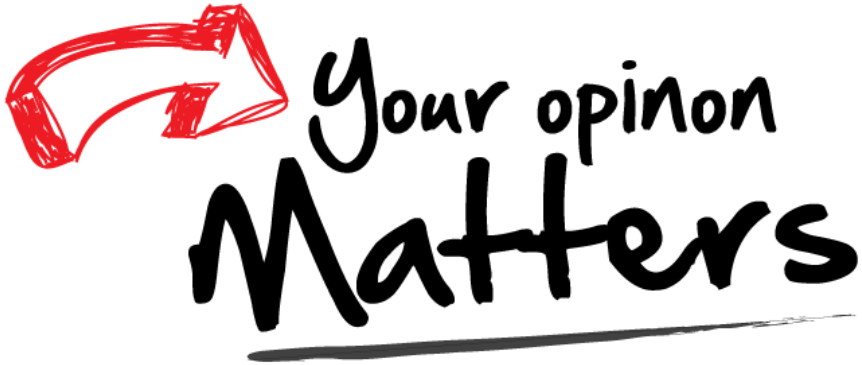
5.5: 识别常见的语法结构

15.5: 词法规则

15.6: 通配符与非贪婪子规则

12: 掌握词法分析的“黑魔法”


Thank
You!



Office 926

hfwei@nju.edu.cn