四号程序员

Keep It Simple And Stupid

使用Antlr的C接口(3.4版)

昨天听同事介绍了Antlr 4，它是一个非常强大的词法、语法分析器辅助生成工具，比之前用Flex + Bison强太多倍。

遗憾的是，当前的v4只支持Java，暂不支持C、C++，于是降级了一把，尝试了一下在3.4版上使用Antlr C。

要说明的是，这不是一篇原创文章，我参考了两篇文章，如下：

《用ANTLR3实现规则解析—-1-安装》 http://blog.csdn.net/wfp458113181wfp/article/details/9148577

《ANTLR Example in C》 http://contrapunctus.net/blog/2012/antlr-c

并针对实际情况，对一些步骤做出了修改、补充，于是有了本文。

一、编译安装 antlr c library

```
wget http://www.antlr3.org/download/C/libantlr3c-3.4.tar.gz
tar -xzvf ./libantlr3c-3.4.tar.gz
./configure --enable-64bit
make
sudo make install

```

二、下载 antlr 3.4 jar包

尽管我们安装了c library，但是从.g文件，到各种.h .c文件的过程，还是要依赖antlr java的。

特别注意：必须要用3.4的jar包，我试了3.5.1，果断不行……

```
wget http://www.antlr3.org/download/antlr-3.4-complete.jar
```

三、编写语法文件(.g)

后续的语法，驱动程序，都是直接照搬开头提到的两篇参考文献，仅做了必要的修改。

```
grammar ExprCppTree;
```

```
2
3  options {
4      language = C;
5      output = AST;
6      ASTLabelType=pANTLR3_BASE_TREE;
7  }
8
9  @header {
10     #include <assert.h>
11 }
12
13 // The suffix '^' means make it a root.
14 // The suffix '!' means ignore it.
15
16 expr: multExpr ((PLUS^ | MINUS^) multExpr)*
17     ;
18
19 PLUS: '+';
20 MINUS: '-';
21
22 multExpr
23     : atom (TIMES^ atom)*
24     ;
25
26 TIMES: '*';
27
28 atom: INT
29     | ID
30     | '('! expr ')'!
31     ;
32
33 stmt: expr NEWLINE -> expr  // tree rewrite syntax
34     | ID ASSIGN expr NEWLINE -> ^(ASSIGN ID expr) // tree notation
       | NEWLINE ->   // ignore
35     ;
36
37 ASSIGN: '=';
38
39 prog
40     : (stmt {pANTLR3_STRING s = $stmt.tree->toStringTree($stmt.tree);
41              assert(s->chars);
                 printf(" tree \%s\n", s->chars);
42             }
43       )+
44     ;
45
46 ID: ('a'..'z'|'A'..'Z')+ ;
47 INT: '~'? '0'..'9'+ ;
48 NEWLINE: '\r'? '\n' ;
49 WS : (' '|'\t')+ {$channel = HIDDEN;};
50
51
```

四、生成c中间文件 (Antlr Target C)

```
1 java -jar ../antlr-3.4-complete.jar ./ExprCppTree.g
2 # 看一下文件，应该有这些
3 ExprCppTree.g  ExprCppTreeLexer.c  ExprCppTreeLexer.h  ExprCppTreeParser.c  ExprCppTree-
  Parser.h  ExprCppTree.tokens
```

五、编写驱动文件

这里同样照搬的，main.cpp

```cpp
#include "ExprCppTreeLexer.h"
#include "ExprCppTreeParser.h"
#include <cassert>
#include <map>
#include <string>
#include <iostream>

using std::map;
using std::string;
using std::cout;

class ExprTreeEvaluator {
    map<string,int> memory;
public:
    int run(pANTLR3_BASE_TREE);
};

pANTLR3_BASE_TREE getChild(pANTLR3_BASE_TREE, unsigned);
const char* getText(pANTLR3_BASE_TREE tree);

int main(int argc, char* argv[])
{
  pANTLR3_INPUT_STREAM input;
  pExprCppTreeLexer lex;
  pANTLR3_COMMON_TOKEN_STREAM tokens;
  pExprCppTreeParser parser;

  assert(argc > 1);
  input = antlr3FileStreamNew((pANTLR3_UINT8)argv[1],ANTLR3_ENC_8BIT);
  lex = ExprCppTreeLexerNew(input);

  tokens = antlr3CommonTokenStreamSourceNew(ANTLR3_SIZE_HINT,
                                            TOKENSOURCE(lex));
  parser = ExprCppTreeParserNew(tokens);

  ExprCppTreeParser_prog_return r = parser->prog(parser);

  pANTLR3_BASE_TREE tree = r.tree;

  ExprTreeEvaluator eval;
  int rr = eval.run(tree);
  cout << "Evaluator result: " << rr << '\n';

  parser->free(parser);
  tokens->free(tokens);
  lex->free(lex);
  input->close(input);

  return 0;
}

int ExprTreeEvaluator::run(pANTLR3_BASE_TREE tree)
{
    pANTLR3_COMMON_TOKEN tok = tree->getToken(tree);
    if(tok) {
        switch(tok->type) {
        case INT: {
            const char* s = getText(tree);
            if(s[0] == '~') {
                return -atoi(s+1);
            }
            else {
                return atoi(s);
            }
        }
        case ID: {
            string var(getText(tree));
            return memory[var];
        }
```

```
64          case PLUS:
65              return run(getChild(tree,0)) + run(getChild(tree,1));
66          case MINUS:
67              return run(getChild(tree,0)) - run(getChild(tree,1));
68          case TIMES:
69              return run(getChild(tree,0)) * run(getChild(tree,1));
70          case ASSIGN: {
71              string var(getText(getChild(tree,0)));
                int val = run(getChild(tree,1));
72              memory[var] = val;
73              return val;
            }
74          default:
75              cout << "Unhandled token: #" << tok->type << '\n';
                return -1;
76          }
77      }
78      else {
79          int k = tree->getChildCount(tree);
80          int r = 0;
81          for(int i = 0; i < k; i++) {
82              r = run(getChild(tree, i));
83          }
            return r;
84      }
85  }
86
87  pANTLR3_BASE_TREE getChild(pANTLR3_BASE_TREE tree, unsigned i)
88  {
89      assert(i < tree->getChildCount(tree));
90      return (pANTLR3_BASE_TREE) tree->getChild(tree, i);
91  }
92
93  const char* getText(pANTLR3_BASE_TREE tree)
94  {
95      return (const char*) tree->getText(tree)->chars;
96  }
97
98
99
100
101
102
103
104
105
106
```

六、编译，测试

生成的可执行文件是test

```
1  # 此处，我直接链接的静态.a库
2  g++ -g -Wall *.cpp *.c ../lantlr3c/lib/libantlr3c.a -o test -I. -I ../lantlr3c/include/
```

测试数据为：

```
1  cat ./data
2  1+2*(3+4)
```

测试结果：

```
./test ./data
 tree (+ 1 (* 2 (+ 3 4)))
Evaluator result: 15
```

您可能也喜欢如下文章：

1. C++使用strtok实现分割字符串。
2. 让flex中支持中文scanner
3. 求最长的数字字符串
4. 试用ICTCLAS分词系统
5. 数据结构重读 – 括号匹配

This entry was posted in Linux and tagged Antlr, C, Target, 词法分析, 语法分析 on 2013-11-09 [https://www.-coder4.com/archives/4016] .