Flex 和 Bison 生成 C++ 代码方法与应用详解

廖琼章
(广西现代职业技术学院,广西 河池 547000)

【摘 要】 flex 和 bison 是 Linux 操作系统中两个重要的语言工具。对 flex 和 bison 生成的代码进行了分析与研究,通过一个实例讲解如何使用 flex 和 bison 产生一个语法分析器的 C++ 代码。

【关键词】 词法分析;语法分析;flex;bison
【中图分类号】TP314 【文献标识码】 A 【文章编号】 1003-2673(2011)06-59-03

1 Flex 和 bison 简单介绍

Flex 和 bison 都是 GNU 项目产品,源代码公开,免费使用。Flex 是词法生成器,即单词识别器,用于从输入流中提取所需要的词素单元。用户只需要把相应语言所用的单词用正则表达式用一个文件描述出来,Flex 能方便快捷的把它转换成对应 C 语言源程序。每一种语言都有一定的语法规则,这些语法规则使用通常使用 BNF 格式描述,把这些描述生成一个后缀名为.yy 的文件。Bison 是根据.yy 文件中的语法规则,生成一个 LALR(l)语法分析器。因此,语法规则描述必须符合 LALR(l)能识别的规则,一方面可以减少冲突,另一方面是因为 bison 对 LALR(l)语法规则产生的代码进行了优化。

2 为什么要用 C++

Flex 和 bison 默认生成的是 C 语言源程序,这一点是为了与 lex 和 yacc 保持兼容。Flex 和 bison 也可以生成 C++ 代码。C++ 代码相对 C 语言源程序来说,维护相对简单一些,特别是对一些比较复杂工程项目使用 C++ 更加容易实现和维护。由 LALR(l)工作原理可知,当终结符数量比较大时,分析表占用的空间非常大,会影响到编译程序的运行效率。正是因为这一点,yacc 限定了终结符的数量不能超过 126 个。Bison 虽然没有对终结符的个数进行限制,但它的工作原理和 yacc 一样。考查下列一组 FX2 系列 PLC 的功能指令:

MOV	D100	D200
DMOV	D100	D200
MOVP	D100	D200
DMOVP	D100	D200

这四条都是传送指令,只是执行方式不同:第一行是 16 位传送,第二行是 32 位传送,第三行是 16 位上升沿传送,第四行是 32 位上升沿传送。如果用 4 个终结符来描述区分他们是可行,但是,FX2 系列 PLC 有 246 条功能指令,按平均一条指令有 3 种形式算,需要 738 个终结符,还不算操作数和逻辑指令。这么多的终结符生成的分析表,将占用了大量的存储空间,导致算法效率变低。如果我们按指令所带的操作数个数来分类,可以把这 246 种指令分成 4 类:一是无操作数指令,二是只带一个操作数的指令,三是带 2 个操作数的指令,四是带 3 个操作数的指令。每一种指令又最多有 4 种不同的操作方式,这样用 16 个终结符来描述就够了,从而大大减少终结符的使用。我们可以使用 C++ 定义一个通用的基类,形式如下:

```
class basefunc{
.....;
Public: virtual execute();
.....;
};
```

其他的指令都从 basefunc 继承如传送类指令:

```
Class movfunc:public basefunc
{…….;
Public :virtual execute();
….
};
Class dmovfunc:public basefunc
{
……
Public:virtual execute();
……
};
```

当词法分析程序识别出传送类指令时只需要 basefunc* p = new movfunc 或者是 basefunc* p=new dmovfunc; 把终结符 FUNC2 或者是 DFUNC2 返回给 bison 的同时也把 p 作为终结符的语义值返回。在语法分析模块代码中充分应用 C++ 的多态性,使用 p->execute()来执行相应的动作。使用这种方法有利于对编译器的设计模块化,可以简单协调多人共同协作完成,维护也更加方便。

3 Flex 生成 C++ 代码方法及生成的类的使用

要想使用 Flex 生成的 C++ 代码,先要对 FlexLexer.h 头文件进行分析,了解它的工作原理。通过研究分析 FlexLexer.h 可以知道 Flex 定义了一个抽象基类 FlexLexer,并定义了它的继承类 yyFlexLexer。由 #define yyFlexLexer yyFlexLexer 这一行,可知我们可以通过对 yyFlexLexer 重新定义,让他生成另一个类,这样我们就可以在一个项目中对不同的语言生成不同的单词识别类。要实现这一点我们只要在.ll 文件中加入一个% option prefix="Exa1"即可。有了这一行后,Flex 生成代码的开

【作者简介】廖琼章(1975-),男,广西河池人,讲师,研究方向:代码生成自动化和编译技术。

始处就生成了 #define yyFlexLexer Exa1FlexLexer 这样的预处理语句。因此 class yyFlexLexer:public FlexLexer 实际上就是 class Exa1FlexLexer:public FlexLexer，生成了我们想要的新类。然而这个类还是比较复杂，我们还可以再对其进一步封装，让它使用起来更加简单。由于 yylex()是虚函数，Flex 默认生成的是 Exa1FlexLexer::yylex();这对于我们的编译程序来说还远远不够，因为单词除了语义之外还语义值。比如："D100"这一个单词，除了返回"DREG"这个信息之外，我们还想知道它到底是哪个数据寄存器，必须把它的地址（编号）100 也一起返回。因此必须要写我们自己的带参数的 yylex()。使用 #define YYDECL int Exa1Scanner::yylex(int* lval)来对 YYDECL 重新定义。这样就使得原来的 Exa1FlexLexr::yylex()变成没有定义，因此我们在.ll 文件中需要补充对它的定义。

4 bison 生成 C++ 代码的研究

Bison 根据.yy 文件中的语法规则转换成相应的 C 代码或者是 C++ 代码，在此只对生成的 C++ 代码进行研究。要使用 bison 生成的 C++ 类，就必需先对 bison 生成的相关文件(.tab.h 和.tab.cc)进行分析和研究。通过分析，我们了解到 Bison 默认生成了一个 Parser 类，同时另外产生了 2 个附加的类，一个 class position;，另一个是 class stack;。Position 类主要提供了一个精确定位单词的位置的操作，提供更多的出错信息，方便用户在运行编译程序时或者是作语法分析遇到语法错误时输出准确的出错位置。Stack 类提供了一个堆栈操作类，服务于 Parser 类，这是由 LALR（1）的工作原理决定的，用户不必对这 2 个类进行改写。通过对.tab.h 和.tab.cc 这 2 个文件的研究分析，我们发现：在 int Parser::parse()中调用的 yylex 并没有定义，也没有预先声明。这使用得我们有机会对所调用的 yylex 重新定义为我们想要的形式。因此，我们可以根据实际需要把它定义为函数或者宏。

5 Flex 和 bison 联合应用举例详解

为了节省编幅，就举一个简单的例子说明 Flex 和 bison 联合应用的过程和步骤。

第一步生成 Exa1.ll 文件，由它生成词法分析程序代码文件 lex.Exa1.cc。Exa1.ll 的内容并注释如下如下：

```
%{
#include "Exa1Scanner.h"
%}
INTEGER [0-9]+
FLOAT ([0-9]+)?\.[0-9]+
END  [eE][Nn][Dd]
/* c++ 表示生成 C++ 代码，prefix="Exa1" 表示生成的类名的前缀，noyywrap 表示生成一个默认的返回 1 的 yywrap() */
%option c++ noyywrap prefix="Exa1"
%%
{FLOAT}        {return TOKENS::FLOAT;}
{INTEGER}      {return TOKENS::INTEGER;}
{END}          {return TOKENS::END;}
.
%%
/* 由于我们在 Exa1Scanner.h 中对 YYDECL 重新定义：#define YY_DECL int Exa1::Exa1Scanner::yylex()，因此在此处必须对 Exa1FlexLexer::yylex()给予定义 */
int Exa1FlexLexer::yylex()
{
 std::cout<<" can not use Exa1FlexLexer::yylex()"<<std::endl;
return 0;
}
```

第二步是建立语法分析文件 Exa1.yy，由它产生语法分析程序代码文件 Exa1.tab.cc。Exa1.yy 的内容并注释如下如下：

```
%require "2.3"   // 所需要的 bison 的版本
%skeleton "lalr1.cc"  // 所生成的语法分析程序的框架
%defines  // 同时生成头文件
%define namespace "Exa1" // 使用的命名空间名字为 Exa1
%define parser_class_name "Exa1Parser"  // 生成的 C++ 类名为 Exa1Parser
%parse- param     {Exa1::Exa1Scanner &scanner} //Exa1Parser 类的一个数据成员，通常使用引用
%code requires
{
namespace Exa1 {class Exa1Scanner;}
}
%token INTEGER FLOAT END
%start prgs
%{
#include "Exa1Scanner.h"
//undef the yylex so we can use our Exa1::Exa1Scanner::yylex now
// 重新定义 Exa1Parser::parse()中使用的 yylex
#undef yylex
#define yylex scanner.yylex
%}

%%
prgs:prg
 |prgs prg
 |prgs END {std::cout<<"program end!!\n";return 0;}
 ;
prg:FLOAT {std::cout<<"Float"<<std::endl;}
 |INTEGER {std::cout<<"Integer\n";}
 ;
%%
// 必需自己定义一个错误处理函数
 void Exa1::Exa1Parser::error (const Exa1Parser::location_type& loc,const std::string& msg)
```

```
{
std::cerr<<"Syntax error!!!\n"<<std::endl;
}
```

第三步写自己的 Exa1Scanner.h **文件。该文件内容如下：**

```
#pragma once
#include <iostream>
#include "Exa1.tab.hh"
typedef Exa1::Exa1Parser::token TOKENS;
 //#define YY_DECL TOKENS Exa1::Exa1Scanner::
yylex
(Exa1::Exa1Parser::semantic_type* lval)
#define YY_DECL int Exa1::Exa1Scanner::yylex()
#ifndef yyFlexLexer
#define yyFlexLexer Exa1FlexLexer
#include<FlexLexer.h>
#endif
namespace Exa1{
class Exa1Scanner:public Exa1FlexLexer
{
public:
Exa1Scanner():Exa1FlexLexer(0,0){}
virtual ~Exa1Scanner(){}
virtual int yylex();
 virtual int yylex (Exa1::Exa1Parser::semantic_type* lval)
{yylval = lval;return yylex();}
private:
Exa1::Exa1Parser::semantic_type* yylval;
};
}
```

第四步写 Main.cc **联合** Flex **和** bison **的驱动程序，内容如下**:

```
#include "Exa1Scanner.h"
int main()
{
Exa1::Exa1Scanner sc;
Exa1::Exa1Parser parser(sc);
return parser.parse();
}
```

第五步 执行 flex Exa1.ll ;bison Exa1.yy;g++ *.cc –o test **就生成了一个** test **的可执行程序。以上过程在** Fedora 13 **系统上实践通过。把生成的相应** C++ **代码复制到** Windows XP SP3 **系统上，使用** VC++2008 **编译通过。**

参考文献

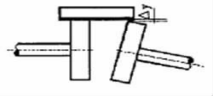
[1]吕映芝，张素琴，蒋维杜.编译原理[M].北京：清华大学出版社，1998.

（上接第 46 页）

振动。联轴器的螺栓不能任意互换。联轴器法兰盘上的联接螺栓是经过称重的，每一螺栓重量基本一致。此外，在拧紧联轴器的联接螺栓时，对称、逐步拧紧采用测力矩扳手使每一螺栓上的拧紧力矩基本一致，以免因各螺栓受力不匀而使联轴器在装配后产生歪斜。严控密封部位的装配。在装配完后检查联轴器的刚性可移件能否进行少量的移动，并进行检查有无卡滞现象。

由于联轴器在加工、装配等环节中，不可避免地存在误差，导致联轴器联接的两轴产生相对位移，破坏其同轴度。因此，安装后的联轴器在投入运行前要对其进行调试。

联轴器调试涉及的是联轴器所联两轴的同轴度，应控制在规定的范围内。校正工具采用比较简单直观的直尺，如下图所示。



用直尺测两轴位移

用直尺和塞尺测量联轴器的外圆和端面或轴间相对位移见上图。如不符合校正精度，须反复调试直至在两个相互垂直的平面内的偏移量都小于允许值为止。

3 注意事项

⑴ 在用直尺调试时，为提高调整精度，事先应将联轴器径向端面的表面清理干净，使之平整、光滑、无锈、无毛刺。

⑵ 为避免测量误差加大，用专用工具调试时，应作好记号。把法兰盘均分为 4～8 个测点，以便测到精确数据。

⑶ 对于大型设备如 P1102 的安装，调试是“先高低再水平”，因为高低(径向位移)由垫片控制的，螺栓紧固后才能测量出差值，如果先调水平 (轴向位移) ，再调高低 ，调高低后水平位置会移动。

⑷ 使用百分表进行测量前，应使百分表同步。大小指针都指“0”，这样有利于读数。否则会出现小指针指在刻度“2”和“3”的中间，而大指针位于“0”左右，读数误差较大。

参考文献

[1]齐福江.机械工程手册[M].北京：机械工业出版社，1997.
[2]中国工程建设标准化协会化工工程委员会.化工机器安装工程施工及验收规范》HGJ20203-2000，2000-12-21.