第31卷 第14期 计 算 机 工 程 2005年7月
Vol.31 № 14 Computer Engineering July 2005
·软件技术与数据库· 文章编号：1000—3428(2005)14—0102—03 文献标识码：A 中图分类号：TP311

C/C++代码自动生成脚本语言接口的实现

官尚元[1,2]，张芝萍[2]，徐立锋[2]，缪 敬[2]
（1. 哈尔滨工程大学计算机科学与技术学院，哈尔滨 150001；2. 中兴通讯公司成都研究所，成都 610041）

摘 要：对于开发灵活的科学软件来说，脚本语言是一个强大的工具。然而开发人员经常遇到一个问题：如何将编译过的C/C++代码集成到一个解释器。为了解决上述问题，设计了一个可扩展的编译器——接口产生器（IG）。IG主要任务是把编译过的C/C++代码集成到脚本语言解释器中。因此，该文的主要目的就是解决上述相关问题。

关键词：科学软件；脚本语言；Python；接口编译器

Implementation of Automatic Generating Scripting Language Interfaces with C/C++ Code

GUAN Shangyuan[1,2], ZHANG Zhiping[2], XU Lifeng[2], MIAO Jing[2]
(1.Computer Science and Technology Institute, Harbin Engineering University, Harbin 150001;
2.Institute of Chengdu, ZTE Corporation, Chendu 610041)

【Abstract】 Scripting languages are powerful tool for the construction of flexible scientific software. However, a common problem faced by the developers of a scripted scientific application is that of integrating compiled C/C++ code with an interpreter. To solve this problem, an extensible compiler, interface generator (IG), is designed to automate the task of integrating compiled code with scripting language interpreters. Therefore, the primary goal of this paper is to cover these topics.

【Key words】 Scientific software; Scripting languages; Python; Interface compiler

1 国内外现状

解决脚本语言和C/C++代码集成的主要办法是开发一个可扩展的接口编译器，将C/C++代码转换成指定的脚本语言接口。这样，很多问题就迎刃而解。比较出名的工具有：

（1）SWIG（simplified wrapper and interface generator）：它是由一个具有特殊用途的C++编译器发展而来的[3]。最早出现在1995年Los Alamos国家实验室，用来为SPaSM短程分子动态学代码生成脚本语言接口。

（2）F2PY （Fortran to Python interface generator）：主要是在Fortan和Python之间提供一个连接。它实际上是Python的一个扩展工具，根据指示文件产生Python C/API模块[4]。它不支持C/C++。

我们自己设计了一个IG（Interface Generator）——可扩展的接口编译器。它支持Tcl、Python、Perl和C、C++等多种语言，同时也高效地支持了C/C++高级编程风格。目前，IG用于自动测试工具中并取得了良好的效果。随着IG的不断优化，其用途必将越来越广。

2 可扩展的编译器（IG）

IG是一个简化脚本语言和科学软件集成的接口编译器，但不是一个完整的C/C++编译器。其基本原理是：直接把C/C++头文件编译成脚本语言接口。

2.1 IG的实现

2.1.1 实现

IG的前端是由一个扩展的C++预处理器和分析器组成，但它们的实现方式与传统的不一样。原因在于它们的输入文件同时混合了特殊的IG指示(前面以%开头)和C++声明。IG主要关心的是接口，因此不会去分析函数体。从内部来说，IG构造了一个完整的分析树并提供了一个横向的API。那些API与XML-DOM规范[5]中所描述的相类似。

在代码生成期间，一系列分层次的处理函数来处理分析树节点。处理函数会决定是直接生成包代码还是转换节点并交给其它处理函数来处理。利用C++类虚拟方法选择机制来定义每个目标语言的行为。从最小要求来看，一个语言模型仅仅需要实现产生函数和变量的处理函数。例如：

```
class MinimalLanguage:
public Language {
    public:
        void main(int argc, char * argv[]);
        int top(Node * n);
        int functionWrapper(Node * n);
        int constantWrapper(Node * n);
        int nativeWrapper(Node * n);      };
```

为了能对类和结构体打包，一个语言模型通常会实现更多的处理函数。下面是IG实现Tcl语言模型：

```
class TCL8 : public Language {
    public:
        virtual void main(int argc, char *argv[]);
        virtual int top(Node *n);
        virtual int functionWrapper(Node *n);
        virtual int variableWrapper(Node *n);
```

基金项目：国家“863”计划基金资助项目（2002AA1Z2306）
作者简介：官尚元（1979—），男，硕士，主研方向：嵌入式软件开发环境；张芝萍、徐立锋、缪 敬，高工
定稿日期：2004-05-17 **E-mail**：guan.shangyuan@zte.com.cn

```
        virtual int constantWrapper(Node *n);
        virtual int nativeWrapper(Node *n);
        virtual int memberfunctionHandler(Node *n);
        virtual int membervariableHandler(Node *n);
        virtual int constructorHandler(Node *n);
        virtual int destructorHandler(Node *n);
        virtual int destructorHandler(Node *n);
        virtual int validIdentifier(String *s);
    char *  usage_string(char *iname, SwigType *,
ParmList *l);
}
```

其余的语言模型实现方式都是类似的。

2.1.2 IG 组织结构

图 1 显示了 IG 的组织结构。预处理器的功能是实现条件编译和宏扩展，生成 IG 和 C 编译器都可接受的接口/头文件。IG 的核心部分是一个 YACC（Yet Another Compiler-Compiler）[6]分析器。分析器的输入是一个包含 IG 指示和声明的文件。为了生成代码，分析器调用大约 10 多个的函数来实现如下的一些功能：写一个包装函数，连接一个变量，对一个 C++成员函数打包等。每个目标语言都是以一个 C++类来实现的。当然，这个类也包含能产生 C 代码的函数。文档系统实现的方式也是类似的。能产生的文档有：ASCII，LaTeX 或 HTML。IG 的输出是一个 C 文件和一个文档文件。C 文件也可以集成到解释器中；文档文件仅供大家参考。

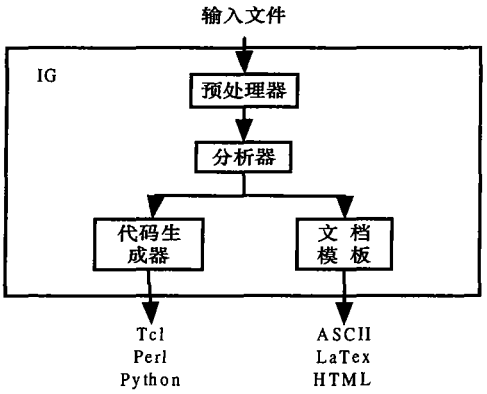

图 1 IG 的组织结构

2.2 关键技术

该小节主要解决类型、类、对象和一些 C++高级编程风格的转换问题。

2.2.1 类型体系

实现 IG 真正的问题在于复杂的类型体系。对于类型，需要正确地处理程序中的 typedef，名字空间和其它类型相关的声明。类型是语言之间数据编组的焦点；同时也是许多 C++问题（包括：内存管理，重载，模板和继承）的基础。

IG 的默认方法是：导出所有的非基本的类型并把它们看成是非透明的类型标签指针。这样就可以避免上述相关问题；同时在脚本解释器中可以操作任何类型的对象。IG 使用一个运行类型体系来确认：只有可以接受的类型指针才传递到包中。下面以指针为例来分析 IG 处理类型的方法。

IG 把指针映射成包括值和类型的字符串。例如，一个"Vector *"指针在 Python 中类似：_100f8e2_p_Vector。

在 Python 中，指针是非透明的对象。但能够在不同的 C 函数之间传递并操作。

总的说来，指针在 Python 中和 C 中的工作方式是一样的。二者根本区别在于：Python 中不能废弃指针。换言之，不能够查看对象内部并进行操作。这样做的好处在于，避免了许多数据表示相关的难题，并且适合多目标语言。

其实，Python 中操作 C++对象也是采用类似指针的方法。结构体和联合体的实现机制与 3.2.2 节中的类实现相似。

2.2.2 类和对象

对于面向对象编程来说，IG 把过程和变量映射到目标语言等价对象中去。例如，一个 C 函数映射成脚本语言中的一个命令，一个全局变量映射成脚本语言中的变量。对于 C++来说，IG 能够创建一个真实反映目标语言中类接口的包。例如，定义一个类如下：

```
class Complex{
  double rpart, ipart;
public:
   Complex(double r = 0, double i = 0): rpart(r), ipart(i) {} ;
   double real();
   double imag();
   ...      };
```

为把类打包，首先把类压缩成过程包。如：

```
Complex * new_Complex(double r = 0,
  double i = 0) {
     return new Complex(r,i);      }
void delete_Complex(Complex * self) {
     delete self;      }
double Complex_real(Complex * self) {
     return self->real();}
...
```

接下来，这些过程用来构建用目标语言写的代理类。如，用 Python：

```
class Complex:
    def_init (self, * args):
       self.this = new Complex(* args)
       self.thisown = 1
    def_del (self):
       if self.thisown:
          delete Complex(self.this)
    def_real(self):
    return Complex real(self.this)
    ...
```

在代理类中，对于下层的 C++对象保存在一个特殊的属性中（.this）。属性值包括指针值和类型表示符。例如，一个 Complex *对象的属性值：

```
_100f8ea0_p_Complex
```

在包中的类型信息用于运行类型检查。类型检查遵循 C++相同的规则，如：继承，域和 typedef。此外，IG 完全支持多重继承。

2.2.3 特殊的指示

对于复杂的应用程序，IG 可能需要一些帮助才能产生包。IG 提供了很多特殊的指示用来控制包代码产生。这些指示可以放在输入文件的任何地方，也可以与 C++声明混合。

假设想把 C++重载操作符与 Python 操作符联系起来。一种方法如下：

```
%rename(_add_) Complex::operator+ (const Complex &) const;
%rename(_sub_) Complex::operator- (const Complex &) const;
%rename(_neg_) Complex::operator- () const;
  ...
```

```
class Complex {
    ...
    Complex operator+ (const Complex &c) const; //_add_
    Complex operator- (const Complex &c) const; //_sub_
    Complex operator- () const;                 //_neg_     };
```

特殊的指示还可以解决如其它一些问题：C++中的模板和目标语言并不支持的 C++风格。例如，Python 中的赋值语法完全不同于 C++中的。于是，可以忽略赋值操作符：

```
%ignore operator=; // Ignore
assignment everywhere
```

2.2.4 自定义

利用一些特殊的指示可以修改 IG 代码生成器。最常用的指示如类型映射。类型映射改变 IG 中转换包中函数数据的方式并集成到类型体系中。

其余自定义是与所有声明有关。例如，如果想在一个指定的类方法中获得一个 C++异常并转换成一个 Python 异常，那么可以按如下方法：

```
%exceptObject::getitem{
    try{$action} catch(IndexError) {
     PyErr_SetString(PyExc IndexError, "bad index");
return NULL;          }
  }
  class Object{
      virtual Item * getitem(int index);     };
```

如果 Object 是一个基类，异常处理代码会被粘贴到衍生类中出现 getitem 的地方。既然只需定义一次且衍生类自动继承，所以这种方法非常有利于用抽象类定义接口。

常见的自定义指示有：%rename，用于解决名字冲突；%ignore，忽略无用的 C++风格；%exception；%extend，简单的使用新属性来扩展已经存在的类和结构；%feature 等。其具体应用参考 IG 用户手册。

事实上，大量的 IG 指示只不过是宏的预处理，最后都扩展成%typemap 或%feature 指示。

3 总结及以后的工作

IG 是一个简化脚本脚本语言和科学软件集成的编译器。它特别适合现有软件并支持一系列的 C++语言特点。使用自动代码生成器如 IG，程序人员很容易在程序开发的各个阶段使用脚本语言。科学家利用脚本语言很容易构造集数据分析、可视化、数据库管理和网络等一体的软件。

设计 IG 的目的是用来支持 C/C++开发软件，它不适合其他语言。IG 不是一个完整的 C++编译器，也不支持某些先进的 C++特性，如嵌套类定义。此外，诸如重载操作符和模板也需要额外的指示。IG 以后的工作主要集中在生成代码的质量，支持更多的语言和增加可靠的特点。

参考文献

1 Aspect-oriented Software Development. http://www.aosd.net.2003-12-10

2 Beazley D M. Automated Scientific Software Scripting with SWIG. Future Generation Computer Systems, 2003: 599–609

3 Beazley D. IG: An Easy to Use Tool for Integrating Scripting Languages with C and C++. In: Proceedings of the Fourth USENIX Tcl/Tk Workshop, 1996: 129-139

4 Peterson P, Martins J, Alonso J. Fortran to Python Interface Generator with an Application to Aerospace Engineering. In: Proceedings of the Ninth International Python Conference,2000

5 Holzer S. Inside XML. New Riders Publishing, 2001

6 Yacc Theory.http://epaperpress.com/lexandyacc/.2003-12-20

(上接第 56 页)

6 小结

在 Web 信息爆炸的今天，随着 Web 上用户群体的迅猛发展，搜索引擎技术是一个具有极大潜力的研究方向。搜索引擎在传统信息检索的技术基础上采用了针对 Web 特点的各种技术以提高检索性能。本文对搜索引擎技术作了系统的归纳和介绍，分析了各部分的关键技术和相关研究情况，并对未来的发展方向作了展望。目前的搜索引擎系统在信息收集、分类、可视化、个性化以及检索结果的准确性等方面仍有一定的不足，如何利用自然语言处理、人工智能、数据挖掘等技术来提高搜索引擎的性能表现将是未来搜索引擎技术的重要发展趋势。

参考文献

1 Shkapenyuk V, Suel T. Design and Implementation of a High-performance Distributed Web Crawler. In Proceedings of the 18th International Conference on Data Engineering (ICDE'02), San Jose, CA, 2002: 357-368

2 Cho J, Garcia-Molina H, Page L. Efficient Crawling Through Url Ordering. In 7th Int. World Wide Web Conference, 1998

3 Chakrabarti S, van den Berg M, Dom B. Focused Crawling: A New Approach to Topic-specific Web Resource Discovery. In Proc. of the 8th Int. World Wide Web Conference (WWW8), 1999

4 Rennie J, McCallum A. Using Reinforcement Learning to Spider the Web Efficiently. In Proc. of the Int. Conf. on Machine Learning (ICML),1999

5 Spertus E. Parasite: Mining Structural Information on the Web. In : Proc. of the Sixth Int'l World Wide Web Conf. , 1997

6 Cho J, Garcia-Molina H. The Evolution of the Web and Implications for an Incremental Crawler. In Proc. of 26th Int. Conf. on Very Large Data Bases, 2000: 117-128

7 Henzinger M R, Heydon A, Mitzenmacher M, et al. on Near-uniform URL Sampling. In Proc. of the 9th Int. World Wide Web Conference, 2000

8 Raghavan S, Garcia-Molina H. Crawling the Hidden Web. In Proc. of 27th Int. Conf. on Very Large Data Bases, 2001

9 王丽坤，王 宏，陆玉昌. 文本挖掘及其关键技术与方法.计算机科学，2002， 29（12）:12-20

10 王继成，萧 嵘，孙正兴等. Web 信息检索研究进展.计算机研究与发展, 2001, 38(2): 187-194