第 29卷第 6期
2006年 12月

南 京 气 象 学 院 学 报
Journal of Nanjing Institute of Meteorology

Vol 29 No 6
Dec 2006

文章编号: 1000-2022(2006) 06-0859-05

脚本语言中对象技术的研究

陈金辉[1], 董飚[2]

(1. 南京信息工程大学 信息与通信系, 江苏 南京 210044;
2 南京工业职业技术学院 信息工程系, 江苏 南京 210016)

摘要: 以 Delphi中的对象模型为例, 利用封装技术, 在一个类 Pascal的脚本语言基础上构建了一个面向对象的脚本语言。研究了面向对象脚本语言中的对象实现技术, 并分析其工作机制。结合实例, 对中间代码的生成过程进行了说明。结果表明, 该对象技术具有简单性和有效性。

关键词: 脚本语言; 面向对象; 编译子系统; 执行子系统
中图分类号: TP314 文献标识码: A

A Research on Object Technology in Scripting Language

CHEN Jin-hui[1], DONG Biao[2]

(1. Department of Information and Communication, NUIST, Nanjing 210044, China;
2 Department of Information and Engineering, NIIT, Nanjing 210016, China)

Abstract On the basis of a Pascal-like source language and a Delphi object model, a object-oriented scripting language is designed by using wrapping object technology. The paper studies an object technology in scripting languages, and analyses its mechanism. A case study in which a practical code is generated for the scripting object is analyzed. Results show that the object technology is simple and efficient.

Key words: scripting language; object-oriented; compiler subsystem; executor subsystem

0 引言

脚本语言在编程语言和软件工程中通常被专家忽视。取而代之, 他们更注重 C++和 Java等面向对象程序设计语言。其实, 脚本语言和 C++、Java等互为补充。随着机器速度提高、组件技术及因特网的发展, 脚本语言将得到广泛的应用[1]。

本文研究脚本语言中的对象处理技术, 以 Delphi语言中的对象模型[2]为例, 研究如何在一个类 Pascal的脚本语言[3]中引入对象处理机制。所选用的类 Pascal脚本语言的运行环境由两部分组成: 编译器和解释器, 编译器对源代码进行编译生成中间代码, 解释器执行中间代码。本文所建立的面向对象脚本语言系统包括编译子系统、执行子系统两部分, 这两个子系统分别是类 Pascal运行环境中的编译器和解释器的扩充, 提供了对 Delphi对象模型的支持。

主动面向对象脚本语言在协同系统、主动推理系统等方面有广泛的应用前景。目前, 在主动语言领域有一些主动面向对象数据库系统, 如 HiPAC[4]、POSTGRES[5]等以及 AT&TBELL实验室

收稿日期: 2005-09-29; 改回日期: 2006-09-11
基金项目: 南京信息工程大学科研基金资助项目 (Y640)
作者简介: 陈金辉 (1970-), 女, 新疆乌鲁木齐人, 硕士, 讲师, 研究方向: 系统分析与集成, cjh@ nuist edu cn.

的 R++主动面向对象编译系统[6], 但是, 关于主动面向对象脚本语言系统的特点与实现方法尚缺乏系统全面的研究。虽然 XOTcl作为一种有代表性的面向对象的脚本语言, 可方便地实现多种设计模式[7], 但缺少对主动机制的支持。本文对脚本语言中 Delphi对象模型处理时, 充分考虑到对主动机制的支持[8-9]。

1 编译子系统中的对象处理技术

1.1 对象支撑环境

图 1为对象支撑环境, 是编译子系统提供给编译器的使用环境, 由对象编译接口、编译时类库导入器和编译时类库构成。

对象支撑环境在初始化阶段完成 3项工作:

(1)由编译时类库导入器自动装入目标语言和特定领域的类库信息, 建立基本的编译时类库;

(2)用户通过编译时类注册器把自定义的类导入编译时类库;

(3)在对象编译接口中为每个导入类建立该类的类类型信息和类助手实例, 并建立类类型与类助手实例间的关联。

在编译阶段, 当编译器发现当前的语法成分是对象操作时进入对象支撑环境, 在对象编译接口中的类类型表中查找该变量的类类型信息是否已存在? 若不存在, 提供错误报警。若存在, 则找到相关联的类助手, 由类助手通过编译时类库导入器访问编译时类库, 为编译器处理对象提供必要的帮助信息, 包括两个方面: 从编译时类库的类项中获取方法等的调用参数信息, 指导编译器生成压栈指令; 在类项中创建或查找所调用的外部过程号及在外部过程导入说明表中填写导入说明。

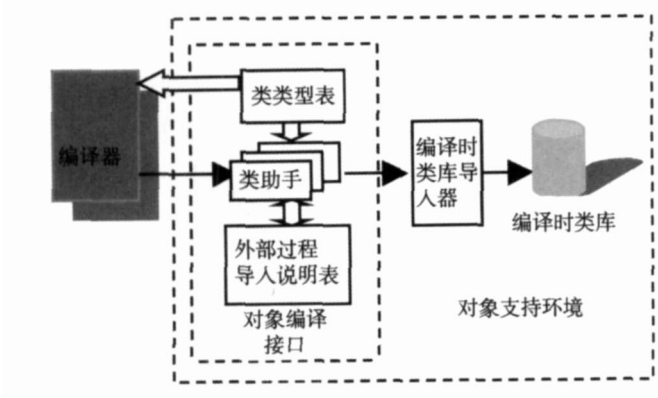

图 1 对象支撑环境

Fig. 1 Object supporting environment

1.2 生成外部过程调用的中间代码

在本文所构建的面向对象脚本语言系统中, 类中的方法、属性和类方法及对对象赋空值、比较和强制类型转换等操作均作为脚本语言中的外部过程, 即通过外部过程的概念引入对象处理机制。每个外部过程均有一个相应的导入说明 importdecl, 通过类助手, 能在外部过程导入说明表中生成或查找到一个外部过程记录。当执行子系统调用外部过程时, 按外部过程的导入说明, 从当前栈顶取出实在参数, 再根据外部过程地址调用该外部过程即可。

压栈指令是编译器生成的。对于应用程序中方法等的调用, 编译器通过类助手, 查找到对应的类项, 再根据类项中的信息生成实参压栈指令。对于应用程序中的对象赋空值等操作, 编译器可根据导入说明, 生成压栈指令。

1.3 外部过程导入说明表

类助手为编译器建立外部过程的同时, 为该外部过程生成导入说明 importdecl, importdecl: := "class"+类名+'|'+类方法名+'|'+cc+params, 其中: 当执行子系统装入该过程时, "class"指示其类型, 该项值不变; cc表示调用约定; params值是由 1和 0组成的字符串, 由编译时类项中的 procdecl决定(当类方法有结果时相应为取 1, 无结果取 0, 为变参时相应为取 1, 其他参数取 0)。

1.4 类助手

类助手为编译器生成中间过程, 有两个作用:

(1)支持方法的调用;

(2)支持赋空值、比较和强制类型转换等操作。

当编译器处理到类或对象时, 通过类助手, 创建或查找一个外部过程, 由该外部过程处理该类或对象。每个类类型实例对应一个类助手。如果应用程序是第一次调用对象的方法、属性和类的类方法, 则在类项中填写相应的过程号, 否则在类项中查找相应的值。

2 执行子系统的外部过程执行技术

2.1 外部过程支撑环境

外部过程支撑环境如图 2所示, 由外部过程执行接口、运行时类库导入器和运行时类库构成, 是执行子系统提供给解释器的使用环境。

外部过程支撑环境在初始化阶段, 建立运行时类库和外部过程记录, 具体包括 3项工作:

(1)由运行时类库导入器自动装入目标语言和特定领域的类的运行时信息, 建立基本的运行时类

库, 支持外部过程的执行;

(2)用户调用运行时类库导入器把自定义的类导入运行时类库;

(3)由外部过程执行接口中的外部过程安装器安装外部过程, 为外部过程生成外部过程记录。

在执行阶段, 当解释器执行到外部过程时进入外部过程支撑环境, 调用外部过程执行器, 外部过程执行器根据相应的外部过程记录通过运行时类库导入器访问运行时类库, 执行外部过程。

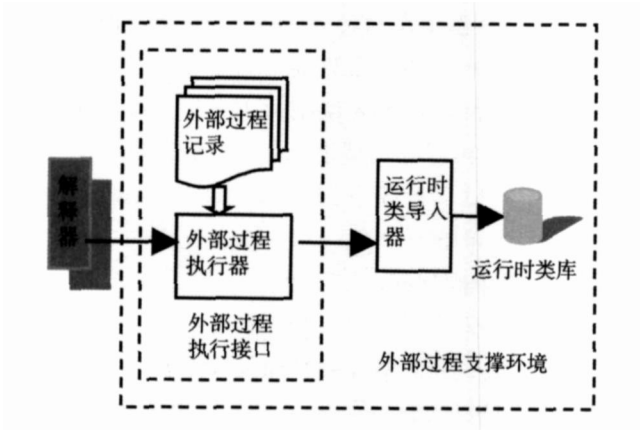

图 2　外部过程支撑环境

Fig. 2　External procedure supporting environment

2.2　外部过程支撑环境的形式描述

外部过程支撑环境 EE: : = < T_1, T_2, CS_{run}, Stack, T_{other} > 。其中: T_1为外部过程记录表; T_2为类类型表; CS_{run}为运行时类库; Stack 为执行子系统的数据栈; T_{other}为执行子系统中其他辅助表。

外部过程执行器执行外部过程与 Delphi对象模型紧密相关。外部过程可被执行, 当且仅当获得以下的参数: < ∂_1, ∂_2, ∂_3 > 。其中: ∂_1是与输入数据有关的{EAX, ECX, EDX, CurrStack}的子集; ∂_2是过程指针; ∂_3是与结果存放位置有关的集合{AL, AX, EAX, EDX: EAX, ST(0)}中的某一元素。

定义如下 6个变换函数:

Δ_1(EE, $ExtPro_i$) = cc, 根据 EE, 获取 $ExtPro_i$的调用约定 cc。

Δ_2(EE, $ExtPro_i$) = Fself, 根据 EE, 获取 $ExtPro_i$类或对象的 Fself。

Δ_3(EE, $ExtPro_i$) = n, 根据 EE, 获取存放 $ExtPro_i$结果的参数。

Δ_4(EE, $ExtPro_i$) = PList, 根据 EE, 使 $ExtPro_i$参数 PList满足构造器和析构器的要求。

Δ_5(EE, $ExtPro_i$) = ptr, 根据 EE, 获取 $ExtPro_i$的实际调用指针。

Δ_6(EE, $ExtPro_i$) = ParList, 根据 EE, 获得 $ExtPro_i$的参数列表 ParList, 使 ParList满足参数传递的要求。

其中: EE为当前外部过程支撑环境, $ExtPro_i$为当前正被调用的外部过程。

下面证明 Δ_1是可实现的。

EE= < T_1, T_2, CS_{run}, Stack, T_{other} > , 而 T_1为外部过程记录表, 表中有外部过程的导入说明项, 导入说明 Importdecl由字符串组成, 其中包含 cc, 证毕。

对于其他变换, 根据 EE, 可逐条验证。

经过 Δ_6, Δ_4, Δ_2, Δ_1等变换, 可获得 ∂_1; 由 Δ_5可获得 ∂_2; 由 Δ_3可获得 n, 根据 n的类型, 可决定 ∂_3。因此, 经 Δ_6、Δ_5、Δ_4、Δ_3、Δ_2、Δ_1变换后, 可获取 < ∂_1, ∂_2, ∂_3 > 值。至此证明外部过程在本文设计的系统模型可以实现。

2.3　外部过程执行器

2.3.1　外部过程执行器结构

根据变换函数, 先进行 Δ_1、Δ_2、Δ_3、Δ_4、Δ_5变换, 变换的顺序和变换所对应的算法须根据外部过程的类型分别设计, 再进行 Δ_6变换, 获取 < ∂_1, ∂_2, ∂_3 > 值, 最后执行外部过程。据此而设计的外部过程执行器结构具有层次特性, 如图 3所示。依据不同的参数传递方式, 外部过程执行器为每类外部过程设计一通用的调用函数。设计、扩充调用函数, 是外部过程执行器执行代码能力的关键。

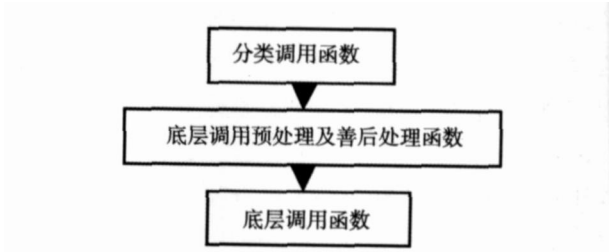

图 3　外部过程执行器结构

Fig. 3　Structure of external procedure executor

2.3.2　分类调用函数

为完成除 Δ_6外的变换, 把调用函数分为 6类: 构造函数、虚拟构造函数、公布的属性、事件属性、方法、属性, 分别对应 ClassCallProcConstructor、ClassCallProcVirtualConstructor、ClassCallProcProperty、ClassCallProcEventPropertyHelper、ClassCallProcMethod、ClassCallProcPropertyHelper, 通过分类调用函数, 可以获取不同类型的外部过程的过程指针 Ptr, 类或对象的 FSelf, 调用约定 cc, 参数列表 ParamList, 结果 n等参数。

2.3.3 底层调用函数

根据调用约定、参数传递、方法调用的不同,设计了 6个过程(RealCall_Register, RealCall_CDecl, RealCall_Other, RealFloatCall_Register, RealFloatCall_CDecl和 RealFloatCall_Other)组成底层外部过程调用函数,包含了除 SafeCall以外的所有情况。

2.3.4 底层调用预处理及善后处理函数

当调用函数获取参数后,就要对参数进行预处理,包括确定参数传递顺序、计算含有间接寻址的参数在内存中的直接地址、确定调用何种底层函数等。

当从底层函数返回时,做一些善后处理,把结果值赋给相应变量,如:把结果返回给调用函数。

3 用例分析

例:下面的程序在屏幕上显示一窗体。

```
var f:TForm;
begin f = TForm.CreateNew(f,0);
      f.Show;
      while f.Visible do
      Application.ProcessMessages;
      f.free;
end
```

3.1 中间代码及分析

①类类型列表

……

Type[15]: TApplication//类类型 TApplication

Type[16]: TForm //类类型 TForm

Type[17]: TComponent//类类型 TComponent

②变量列表

Var[0]: 15 TApplication//TApplication型变量

Var[1]: 16 TForm //TForm型变量

③内部过程列表

Proc[0] //过程 0

[0] PUSHTYPE 6//创建类型为 6的变量,并把该变量的地址压栈

[1] ASSIGN Base[1], [0] //把 0赋值给 Stack[1]

[2] PUSHTYPE 17

[3] ASSIGN Base[2], GlobalVar[1] //把全局变量 1的地址赋值给 Stack[2]

[4] PUSHTYPE 5

[5] ASSIGN Base[3], [16]

[6] PUSHVAR GlobalVar[1] //把全局变量 1的地址压栈

[7] CALL 1//调用过程 1

[8] POP//弹栈

……

[34] RET//过程返回

④外部过程导入说明表

Proc[1] class:TForm|CREATENEW|00010000

Proc[2] class:TForm|SHOW|0000

……

在编译器初始化阶段,建立编译时类库、类型信息,并为每个类类型建立类助手。当编译器扫描到 CREATENEW时,根据 TForm的类助手,创建外部过程。在外部过程导入说明表中登记该外部过程的过程号为 1,再以导入说明为模板,生成对外部过程 1的调用 CALL 1指令(导入说明为 class:TForm|CREATENEW|00010000)。

3.2 执行中间代码

在初始化阶段,生成外部过程记录。当执行第[7]条指令,调用过程 1。外部过程执行器根据外部过程记录 FSelf,Ptr, cc, ParamList和 n等参数,建立外部过程执行栈 ParamList,再调用分类调用函数 ClassCallProcVirtualConstructor,其工作如下:

①ClassCallProcVirtualConstructor分别把 FSelf,flag,参数 1赋值给 EAX、EDX、ECX寄存器,参数 2(资源指针 f)压入栈,选择 RealCall_Register作为底层调用函数。

②调用 RealCall_Register后,返回值在 EAX寄存器中,再由转换函数把 EAX中的结果值转换为指针类型后,赋值给 n,也就是使全局变量 f获得调用 TForm.CreateNew(f,0)后得到的对象引用值。

③清空外部过程执行器栈 ParamList,返回解释器。

4 结论

本文构建了面向对象脚本语言的编译子系统、执行子系统模型,提出了外部过程的概念,证明了外部过程的可执行性并予以实现,从而成功引入了对象处理机制,最后剖析了实例代码。此项研究成果已用于开发主动面向对象脚本语言。

参考文献:

[1] Ousterhout J. Additional information for scripting white paper[EB/OL]. (2005-1-20)[2005-08-10]. http://www.ajubasolutions.com/people/john_ousterhout/scriptextra.html/.

[2] Ray Lischner. DELPHI技术手册[M]. 肖雪莲,等译. 北京:中国

电力出版社, 2001.

[3] Remobjects Software Development Company. Pascal Script3.0 [CP/OL]. (2005-3-12) [2005-07-20]. http://www.remobjects.com/.

[4] Dayal U, Blaustein B, Buchmann A. The HiPAC Project: combining active databases and timing constraints[DB/OL]. (1988) [2005-07-20]. SIGMOD Record, 17(1): 51-70.

[5] Andrew Yu, Jolly Chen. The POSTGRES95 User Manual[EB/OL]. [2005-08-18]. http://www.kmc.eku.edu/tools/pg_manual/pg95user.html.

[6] AT&T BELL Laboratories. R++ User Manual[EB/OL]. [2005-08-20]. http://www-db.research.bell-labs.com/user/pfps/rpp/UseManual.ps.

[7] Gustaf N, Uwe Z. Filters as a language support for design patterns in object-oriented scripting languages[C]//Proceedings of the 5th Conference on Object-Oriented Technologies and Systems (COOTS'99). San Diego, 1999: 3-9.

[8] 董飚, 陈金辉, 张颖超. 主动面向对象系统中编译子系统的设计与实现[J]. 微计算机信息, 2005, 21(23): 198-200.

[9] 陈金辉, 董飚, 张颖超, 等. 主动面向对象系统中执行子系统的设计与实现[J]. 微计算机信息, 2006, 22(15): 289-291.