

北京科海培训中心

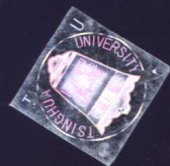
Addison-Wesley

基于C++ **CORBA** 高级编程

[美] Michi Henning Steve Vinoski 著

徐金梧 徐 科 吕志民 等译

清华大学出版社



Addison—Wesley

北京科海培训中心

基于 C++

CORBA 高级编程

[美] Michi Henning, Steve Vinoski 著

徐金梧 徐 科 吕志民 等译

清华大学出版社

(京)新登字 158 号

著作权合同登记号:01-1999-3342

内 容 提 要

CORBA 规范是目前最具生命力的跨平台技术,它独立于网络协议、编程语言和软硬件平台,支持异构的分布式计算和不同编程语言的对象重用。

全书共 22 章,系统地介绍了 CORBA 的基本体系和概念,IDL 语义和映射为 C++ 的规则、POA 和对象生命周期,CORBA 机理和 ORB,动态 CORBA 特性以及 CORBA 重要的服务程序。本书的独到之处在于它不仅介绍概念及资源,更重要的是讲述超越 API 的 CORBA 内部机制、各种设计方案及其优缺点,还有不少令你少走弯路的技巧和建议,此外提供实际开发细节的代码实例。

本书是一本使用 C++ 编写 CORBA 应用程序的实用指南,适用于大学教师和研究生作为教材或参考书,也可作为从事 CORBA 技术开发的软件工程师的参考书。

Advanced CORBA® Programming with C++

Copyright ©1999 by Addison Wesley Longman, Inc.

All rights reserved. No part of this book shall be reproduced, stored in a retrieval system, or transmitted by any means, electronic, mechanical, photocopying, recording, or otherwise, without written permission from the publisher.

本书中文简体字版由美国 Addison-Wesley 公司授权北京科海培训中心和清华大学出版社出版。未经出版者书面允许不得以任何方式复制或抄袭本书内容。

版权所有,盗版必究。

本书封面贴有清华大学出版社激光防伪标签,无标签者不得进入各书店。

书 名: 基于 C++ CORBA 高级编程

作 者: Michi Henning, Steve Vinoski

译 者: 徐金梧 徐科 吕志民等

出版者: 清华大学出版社(北京清华大学校内,邮编 100084)

<http://www.tup.tsinghua.edu.cn>

印刷者: 北京门头沟胶印厂

发 行: 新华书店总店北京科技发行所

开 本: 787×1092 1/16 印张: 47.25 字数: 1154 千字

版 次: 2000 年 7 月第 1 版 2000 年 11 月第 2 次印刷

印 数: 5001~7000

书 号: ISBN 7-302-03956-9/TP. 2316

定 价: 80.00 元

译者序

《基于 C++ CORBA 高级编程》是根据 Addison Wesley Longman 公司 1999 年出版的 *Advanced CORBA[®] Programming With C++* 一书翻译成中文的。原文作者是两位从事 CORBA 技术的资深专家。本书按照 OMG 最新公布的 CORBA 2.3 标准规范,系统深入地介绍了 CORBA 的基本体系和概念,IDL 语义和映射为 C++ 的规则,POA 和对象生命周期、CORBA 机理和 ORB、动态 CORBA 特性以及 CORBA 重要的服务程序。该书始终围绕着工程应用实例,深入浅出地展示了采用 CORBA 开发应用程序的基本步骤和各种实现方案。作者以其独到的眼光和深厚的功底向我们揭示了 CORBA 内在的多彩世界。它不仅是一本教材,还是一本工具书。无论你是初次接触 CORBA(即 CORBA),还是有丰富编程经验的程序员,这本书都会使你受益匪浅。

自 OMG 在 1991 年推出 CORBA 的第一个版本以来,它经历了几年的磨练,现已成为软件开发的主流,并被工业界广泛接受。随着电子商务的迅速崛起,CORBA 将越来越被行家看好。CORBA 独立于网络协议,独立于编程语言,独立于软硬件平台,它是目前最有生命力的跨平台技术。尤其是 OMG 于 1998 年公布了 CORBA 2.3 版,使 CORBA 技术日臻完美。目前世界上几家重要的系统软件公司以及像 HP 公司,Sun 微系统公司,Inprise 公司,HyperDe 公司以及 IONA 技术公司都纷纷推出各自的 CORBA 2.3 版开发平台。

全书共分 6 部分,由 22 个章节组成。每一部分的内容如下:

第 1 部分:CORBA 简介。这一部分包括 CORBA 的综述,并结合一个简单的 CORBA 应用程序分析 CORBA 的主要组成部分。阅读这部分内容后,你将了解 CORBA 的基本体系结构和概念、理解它的对象和请求发送模型及建立一个 CORBA 应用程序的基本步骤。

第 2 部分:CORBA 的核心。介绍用 C++ 编写 CORBA 的核心:接口定义语言(IDL),将 IDL 映射为 C++ 的规则,如何使用 POA 和如何支持对象生命周期操作。这一部分内容引入了一些工程实例,这些实例将贯穿全书。在以后章节中将利用这些实例来展开 CORBA 的各种特性和应用编程接口(API),并讨论如何运用到实际应用程序中。阅读这部分内容后,你就可以创建使用多种 CORBA 特性的复杂的 CORBA 应用程序。

第 3 部分:CORBA 机理。重点介绍 CORBA 网络协议和支持 CORBA 对象模型的机理,比如位置透明性和协议的独立性。阅读这部分内容后你应对 ORB 底层机制以及不同供应商所提供的设计选择是如何影响某个特定 ORB 的可扩展性、性能和灵活性等有个清晰的认识。第 2、3 部分内容是 CORBA 技术的基础。

第 4 部分:动态 CORBA。这一部分介绍 CORBA 的动态特性,包括:类型 any、类型代码和类型 DynAny。这些内容是体现 CORBA 灵活性、跨平台技术的主要特性。阅读这部分后,你将学会怎样用 CORBA 的动态特性来处理在编译时类型无法确定的那些值。这些知识对创建类属应用程序,如浏览器或协议网桥是必不可少的。

第 5 部分:CORBA 服务。介绍最重要的 CORBA 服务,如 Naming,Trading 和 Event 服务。几乎所有实际使用的应用程序都会用到这类服务。Naming 和 Trading 服务允许各类应

用程序定位到感兴趣的对象上,而 Event 服务则提供异步通信功能,以便客户机与服务器能够彼此去耦。这些服务对于创建像 B2B(Business to Business),B2C(Business to Customer),C2C(Customer to Customer)这种多点对多点的,跨平台的连接具有十分诱人的应用前景。这些内容都是当前最具生命力的分布式计算技术。阅读这部分内容后,你将明白这些服务的目的,体会这种体系的重要性,掌握在设计过程中所要考虑到的各种折衷方案。

第 6 部分:功能强大的 CORBA。在这部分内容中将讨论如何开发多线程服务器程序以及涉及大量体系上和设计上的问题,这对于创建高性能应用程序来说是非常重要的,它涉及到使用 CORBA 技术实现分布式计算的深层次问题。

附录 A:指令式控制协议仿真器源代码,你可以使用这个仿真器来测试本书的源代码。

附录 B:资源列表。给出了大量很有实用价值的资源,你可以通过这个附录来获取有关 CORBA 各方面内容的详尽资料。

本书的第 1、2、3、4 章由徐金梧教授翻译,第 5、6、7、8、9、10 章由徐科博士翻译,第 11、12、13、14、15、16 章由吕志民博士翻译,第 17、18、19、20 章由张晓彤博士翻译,第 21、22 章及附录由张武军博士翻译。徐金梧教授对全书作了译校。

由于时间紧迫,加上译者水平有限,翻译中不妥和错误之处在所难免,殷切希望广大读者和同仁批评指教。另外,译者日前正从事 CORBA 的开发工作,愿与广大读者共同探讨在开发中所遇到的问题。我们的 Email 地址是:jwxu@USTB.EDU.CN,联系电话:010-62332329。

2000 年 5 月于北京科技大学

前 言

多年来,我们俩一直在全世界各地为使用 C++ 的软件工程师讲授 CORBA 编程。在授课过程中,经常被问及:“什么地方可以找到一本有关这些内容的书?”虽然已出版了几本有关 CORBA 的书,但它们大多局限于高层的概念,并不是软件工程师所需要的。尽管从概念上来说,CORBA 并不复杂,困难在于它的实现细节。或者不客气地讲,当你必须找出为什么你的程序会转储核心时,就别指望那些只局限于介绍高层概念的书对你会有什么帮助。

的确,有许多有关 CORBA 的很有价值的资源,比如新闻组、Web 页和对象管理组(Object Management Group,OMG)规范。但是,这些资源并不能真正地满足程序员的需求,程序员所需要的是实际能用的代码。我们编写这本书就是为了向读者提供一本有关 CORBA 的教材和参考书,它从实际软件开发的细节层面向你揭示如何用 C++ 来编写 CORBA(当然,我们写这本书也是为了给我们的学生有个交待)。

写这样一本书是一件苦差使。解释 CORBA 规范和 API(Application Programming Interface)是必不可少的,它是本书必备的部分。但是,了解各种 API 本身并不能使你成为一名出色的程序员。为了成为一名合格的程序员,你不仅需要掌握有关平台机理的知识,而且还必须了解不同特性之间如何进行交互。你必须有效地组合它们,最终使它们变成一个性能和伸缩性良好的、可维护的、可扩展的、可移植的和可配置的应用程序。

为了帮助你成为一名出色的程序员(不是那种只知道概念的人),我们以不同的方式来讲述超越这些基础知识的内容。首先,我们提供一些我们认为是好的(坏的)设计方法,另外我们也不隐瞒 CORBA 所存在的问题(CORBA 也像任何其他复杂软件系统一样具有它本身的问题)。第二,我们通过解释一些 CORBA 的内部机制来介绍讲述超越 API 的内容。你甚至可以在不知道什么是捕集器(hood)的情况下使用 ORB(对象请求代理),理解这些机制是非常有用的,因为它们对一个应用程序的性能有着至关重要的影响。第三,我们留下足够的空间来讨论各种设计方案的优点,尤其是,当一个设计方案在某些方面是有利的,但在其他方面它是不利的。了解这些折衷方案对于成功地建立应用程序是极其重要的。第四,在合适的地方,我们给你一些建议,使你少走弯路。

必然,上面所提及的这些方法就要求我们对它们的内在价值作出判断,但是可以预料,确实会有不少人并不赞成我们的某些建议。不论你是否同意我们的观点,你总会从我们的方法中受益。如果你同意,你可以按照我们所给的建议去做;如果你不同意,至少这些讨论会鼓励你去思考这些专题,并形成你自己的观点。无论哪种情况,总比那些不着边际地在那里空谈概念的书对你更有用。

预 备 知 识

这本书不是为初学者编写的,从某种意义上讲,我们并没有留出很多篇幅来解释 OMG 的结构或者解释规范的选择过程,没有提供 CORBA 体系结构目标的一个完整的综述,也没

有提供有关所有它的服务和功能软件的概述(有关这方面的综述请参阅文献[3]),相反,我们假定你想要知道如何用 C++ 来编写实际的 CORBA 应用程序。尽管缺乏这方面的综述材料,但如果你以前从未接触过 CORBA 的话,也仍然可以掌握这方面的内容。如果你在网络编程上有丰富的经验或使用过其他的 RPC 平台,你就会发现这些都是驾轻就熟的事。

这本书有很多地方给出了源代码,所以我们希望你已经熟悉 C++ 语言。但是,你也不必一定是 C++ 的专家,我们尽量避免使用那些难懂的、不好理解的 C++ 特性,而使用那些简洁明了的表达方式。如果你了解继承、虚函数、操作符重载和模板(不必拘泥于细节),你就不会有问题。有些源代码使用了标准的模板库(Standard Template Library, STL),这些现在已经是 ISO/IEC C++ 标准的一部分。我们只限定使用这个库的简单用法,所以哪怕你以前从未使用过 STL 代码,你也可以理解这些源代码。

如果你从未编写过多线程代码,你可以在本书的有关章节中找到编写多线程服务器的内容。遗憾的是,书中没有足够的篇幅来介绍有关多线程的编程方法。但是,在本书的文献目录中列出了大量介绍这方面内容的优秀参考书。

尽管我们作了最大的努力来解释这些实际能使用的源代码,但是我们仍然不得不做一些妥协,以使这些代码实例更容易理解,篇幅又不至于很长。当我们解释一个特定的特性时,经常只给出相关代码,而在实际的应用程序中,这些代码应当封装在一个类或辅助函数中。另外,我们还尽量压缩了出错处理,这样避免由于太多的异常处理程序混淆了控制流程。我们选择这种方式主要是为了教学需要,并不意味着实际工程应用中可以这样做。本书的参考文献给出了详细讲述源代码设计的大量优秀的参考书。

本书的范围

OMG 成员正在不断地改善 CORBA,并增添一些新的特性。因此,各种有效的 ORB 实现适用于规范的不同修改版本。本书是按照 CORBA 2.3 版编写的(在写这本书时,CORBA 2.3 正由 OMG 作最后的评估)。在整本书中,我们指明了这些新的特性。这些特性也许不适用于你的 ORB 实现,因此为了保持最大的可移植性,要求你只限于较容易的特性集。

尽管这本书已经很厚了,但我们最大的遗憾仍然是还有很多内容没有写进去。不断增加的页码和越来越临近的交稿最后期限都迫使我们放弃有关章节,像动态激发接口(Dynamic Invocation Interface, DII)、动态框架接口(Dynamic Skeleton Interface, DSI)和接口仓库(Interface Repository, IFR)都没有深入地讨论。所幸的是,绝大多数应用程序并不需要这些特性,所以省略了这些章节并不影响大局。如果你的应用程序遇到需要动态接口,我们在这里所提供的背景材料也许可以使你很容易从 CORBA 规范中挑选出所需要的内容。

另一个没有提及到的特性是 Objects-By-Value(OBV)。我们没有介绍 OBV 是因为它对大家来说还太新,以至于还没有任何使用这个特性的实际的工程实例。另外,在编写本书时,这个特性还存在不少技术上的问题需要解决,我们期望 OBV 规范在它落脚之前能经受住进一步地考验。

时间和篇幅的限定还意味着我们不可能将每个可能的 CORBA 服务都包括在这本书中。例如,我们没有涉及事务服务(Transaction Service)或安全服务(Security Service),因为这些内容都应当单独编成一本书。我们不是去追求完整性,而是限定了建立一个应用程序所

需的最重要的那些服务,像命名服务、交易服务和事件服务。我们将这些服务作了比任何我们所列出的出版物更详细的介绍。

本书中一个重要的部分是介绍可移植对象适配器(Portable Object Adapter, POA), POA 是在 CORBA 2.2 版中新添加的。POA 提供了服务器端源代码的可移植性,这是在 BOA(Basic Object Adapter)中遗漏的。POA 也提供了许多在创建高性能、可伸缩的应用程序时最重要的特性。因此,我们给出了较大的篇幅介绍如何在设计应用程序时有效地使用 POA。

总之,我们相信这本书向你展示了用 C++ 编写 CORBA 程序时所需要的最重要的和最全面的信息。我们妥善地综合了这个材料。这本书既是一本教材,同时还是一本参考书。我们唯一的希望是,当你开始阅读这本书,它能开阔你的视线,并能从中获益。如果真的如此,我们就达到了编写这本书的目的,它确实是为了在一线工作的工程师编写实际的应用程序而编写的。

Michi Henning and Steve Vinoski

1998 年 10 月

目 录

第 1 章 导论	(1)	1.4 源代码示例	(3)
1.1 简介	(1)	1.5 有关软件供应商	(4)
1.2 本书内容的组织	(2)	1.6 如何与作者联系	(4)
1.3 CORBA 版本问题	(3)		

第 1 部分 CORBA 简介

第 2 章 CORBA 概述	(5)	2.5.3 对象引用的内容	(18)
2.1 简介	(5)	2.5.4 引用和代理	(19)
2.2 对象管理组	(6)	2.6 CORBA 应用程序的一般开发过程	(20)
2.3 概念和术语	(7)	2.7 本章小结	(22)
2.4 CORBA 特性	(9)		
2.4.1 一般请求流	(9)	第 3 章 一个最小的 CORBA 应用程序	(23)
2.4.2 OMG 接口定义语言	(10)	3.1 本章概述	(23)
2.4.3 语言映射	(11)	3.2 编写和编译一个 IDL 定义	(23)
2.4.4 操作调用和调度软件	(12)	3.3 编写和编译一个服务器程序	(24)
2.4.5 对象适配器	(13)	3.4 编写和编译一个客户机程序	(28)
2.4.6 ORB 间协议	(14)	3.5 运行客户机和服务器程序	(31)
2.5 请求调用	(14)	3.6 本章小结	(31)
2.5.1 对象引用语义	(15)		
2.5.2 引用的获取	(17)		

第 2 部分 CORBA 的核心

第 4 章 OMG 接口定义语言	(33)	4.4.4 定义的顺序	(37)
4.1 本章概述	(33)	4.5 词法规则	(37)
4.2 简介	(33)	4.5.1 注释	(37)
4.3 编译	(34)	4.5.2 关键字	(37)
4.3.1 单个的客户机和服务器程序的 开发环境	(34)	4.5.3 标识符	(37)
4.3.2 客户机和服务器程序的不同 开发环境	(35)	4.6 基本的 IDL 类型	(38)
4.4 源文件	(36)	4.6.1 整型	(39)
4.4.1 文件的命名	(36)	4.6.2 浮点类型	(39)
4.4.2 文件格式	(36)	4.6.3 字符	(39)
4.4.3 预处理	(37)	4.6.4 字符串	(40)
		4.6.5 布尔量	(40)
		4.6.6 八位字节	(40)

4.6.7 any 类型	(40)	4.19 仓库标识符和 pragma 指令	(79)
4.7 用户定义类型	(41)	4.19.1 IDL 的仓库 ID 格式	(79)
4.7.1 命名类型	(41)	4.19.2 prefix 的附注	(80)
4.7.2 枚举	(41)	4.19.3 版本(version)附注	(81)
4.7.3 结构	(42)	4.19.4 使用 ID 附注来控制仓库的 ID 格式	(81)
4.7.4 联合	(43)	4.20 标准的 include 文件	(82)
4.7.5 数组	(45)	4.21 最新的 IDL 扩展	(82)
4.7.6 序列	(45)	4.21.1 宽位字符和字符串	(82)
4.7.7 序列与数组	(46)	4.21.2 64 位整型	(83)
4.7.8 递归类型	(47)	4.21.3 扩展的浮点类	(83)
4.7.9 常量定义和字面值	(49)	4.21.4 定点十进制类型	(83)
4.7.10 常量表达式	(51)	4.21.5 转义标识符	(84)
4.8 接口和操作	(52)	4.22 本章小结	(85)
4.8.1 接口语法	(53)	第 5 章 一个气温控制系统的 IDL	(86)
4.8.2 接口语义和对象引用	(54)	5.1 本章概述	(86)
4.8.3 接口通信模型	(55)	5.2 气温控制系统	(86)
4.8.4 操作定义	(55)	5.2.1 温度计	(86)
4.9 用户异常	(58)	5.2.2 恒温器	(87)
4.9.1 异常设计问题	(59)	5.2.3 监测站	(87)
4.10 系统异常	(61)	5.3 气温控制系统的 IDL	(87)
4.11 系统异常或用户异常	(63)	5.3.1 温度计的 IDL	(88)
4.12 单向操作(one-way operation)	(64)	5.3.2 恒温器的 IDL	(88)
4.13 上下文(contexts)	(65)	5.3.3 控制器的 IDL	(89)
4.14 属性(Attributes)	(66)	5.4 完整的程序	(92)
4.15 模块(Modules)	(67)	第 6 章 基本的 IDL 到 C++ 的映射	(94)
4.16 前向声明(Forward Declarations)	(68)	6.1 本章概述	(94)
4.17 继承(Inheritance)	(70)	6.2 简介	(94)
4.17.1 从类型 object 中隐含的继承	(70)	6.3 标识符的映射	(95)
4.17.2 空接口(Empty Interface)	(71)	6.4 模块的映射	(96)
4.17.3 接口与实现的继承	(72)	6.5 CORBA 模块	(97)
4.17.4 继承的重定义规则	(73)	6.6 基本类型的映射	(97)
4.17.5 继承的限定	(73)	6.6.1 64 位整型和 long double 类型	(98)
4.17.6 多重继承	(74)	6.6.2 基本类型的重载	(98)
4.17.7 多重继承的限定	(75)	6.6.3 可映射成 char 的类型	(99)
4.18 名称和作用域	(76)	6.6.4 wchar 的映射	(99)
4.18.1 命名作用域	(76)	6.6.5 Boolean 映射	(99)
4.18.2 区分大小写	(76)	6.6.6 字符串和宽位字符串映射	(99)
4.18.3 在嵌套作用域中的名称	(77)		
4.18.4 名称查找规则	(77)		

6.7 常量的映射	(100)	6.16.4 包含复杂成员的联合	(144)
6.8 枚举类型的映射	(102)	6.16.5 使用联合的规则	(145)
6.9 变长度的类型与_var 类型	(102)	6.17 递归结构和递归联合的映射	(146)
6.9.1 _var 类型的使用	(103)	6.18 类型定义的映射	(146)
6.9.2 变长度类型的内存管理	(105)	6.19 用户定义类型和_var 类	(147)
6.10 String-var 封装类	(106)	6.19.1 用于结构、联合和序列的_var 类	(148)
6.10.1 使用 String-var 的缺陷	(109)	6.19.2 _var 类的简单使用	(149)
6.10.2 将字符串作为传递参数以读取字符串	(111)	6.19.3 使用_var 类的一些缺陷	(150)
6.10.3 将字符串作为传递参数以更改字符串	(112)	6.19.4 定长度的结构、联合和序列与变长度的结构、联合和序列之间的区别	(150)
6.10.4 隐式类型转换产生的问题	(113)	6.19.5 数组的_var 类型	(152)
6.10.5 取得对字符串的所有权	(115)	6.20 本章小结	(155)
6.10.6 流运算符	(116)	第7章 客户端的C++映射	(156)
6.11 宽位字符串的映射	(116)	7.1 本章概述	(156)
6.12 定点数类型的映射	(116)	7.2 简介	(156)
6.12.1 构造函数	(117)	7.3 接口的映射	(156)
6.12.2 存取函数	(118)	7.4 对象引用类型	(157)
6.12.3 转换运算符	(118)	7.5 对象引用的生命周期	(158)
6.12.4 截断与舍入	(118)	7.5.1 删除引用	(159)
6.12.5 算术运算符	(119)	7.5.2 引用拷贝	(160)
6.12.6 流运算符	(119)	7.5.3 引用计数值的范围	(161)
6.13 结构的映射	(119)	7.5.4 空引用	(161)
6.13.1 定长度结构的映射	(119)	7.6 _ptr 引用的语义	(163)
6.13.2 变长度结构的映射	(120)	7.6.1 代理与_ptr 引用的映射	(163)
6.13.3 结构的内存管理	(122)	7.6.2 继承与拓展	(165)
6.13.4 包含结构成员的结构	(123)	7.6.3 紧缩转换	(167)
6.14 序列的映射	(124)	7.6.4 类型安全的紧缩(Narrowing)	(167)
6.14.1 无界序列的映射	(124)	7.6.5 非法使用_ptr 引用	(168)
6.14.2 有界序列的映射	(134)	7.7 伪对象	(169)
6.14.3 序列使用中的一些限制	(135)	7.8 ORB 的初始化	(170)
6.14.4 序列的使用规则	(137)	7.9 初始引用	(171)
6.15 数组的映射	(137)	7.9.1 将字符串转换成引用	(172)
6.16 联合的映射	(139)	7.9.2 将引用转换成字符串	(174)
6.16.1 联合的初始化和赋值	(140)	7.10 字符串化引用	(175)
6.16.2 联合的成员与鉴别器的访问	(141)	7.10.1 初始的字符串化引用	(175)
6.16.3 没有 default 语句的联合	(142)	7.10.2 字符串化引用的长度	(175)
		7.10.3 字符串化引用的互用性	

..... (176)	7.14.15 参数传递的陷阱 (216)
7.10.4 字符串化引用的规则 (176)	7.15 异常映射 (218)
7.11 对象伪接口 (176)	7.15.1 系统异常的映射 (220)
7.11.1 _is-a 操作 (177)	7.15.2 系统异常的语义 (223)
7.11.2 _non-existent 操作 (178)	7.15.3 用户异常的映射 (226)
7.11.3 _is-equivalent 操作 (180)	7.15.4 异常说明 (227)
7.11.4 _hash 操作 (181)	7.15.5 异常和 out 参数 (228)
7.11.5 Object 操作映射小结 (182)	7.15.6 ostream 插入符 (228)
7.12 _var 引用 (182)	7.15.7 不支持异常的编译器中的映射 (229)
7.12.1 _var 引用的映射 (183)	7.16 上下文的映射 (230)
7.12.2 _var 引用与拓展 (186)	7.17 本章小结 (230)
7.12.3 同时使用 _var 和 _ptr 引用 (187)	
7.12.4 嵌套在用户定义类型中的引用 (189)	第 8 章 开发气温控制系统的客户程序 (231)
7.12.5 _var 类型的效率 (190)	8.1 本章概述 (231)
7.13 操作与属性的映射 (191)	8.2 简介 (231)
7.13.1 操作的映射 (191)	8.3 客户程序的总体结构 (231)
7.13.2 属性的映射 (192)	8.4 包含文件 (232)
7.14 参数传递规则 (193)	8.5 辅助函数 (233)
7.14.1 定长度类型与变长度类型 (194)	8.5.1 显示装置的具体内容 (233)
7.14.2 生成的 _out 类型 (195)	8.5.2 打印出错异常信息 (235)
7.14.3 简单类型的参数传递 (195)	8.6 main 函数 (237)
7.14.4 复杂的定长度类型的参数传递 (196)	8.6.1 初始化 (237)
7.14.5 包含定长度元素的数组的参数 传递 (197)	8.6.2 与服务器程序的交互 (238)
7.14.6 变长度参数的内存管理 (200)	8.7 完整的客户程序代码 (243)
7.14.7 字符串和宽位字符串的参数传递 (203)	8.8 本章小结 (248)
7.14.8 复杂变长度类型和 Any 类型的 参数传递 (205)	第 9 章 服务器端 C++ 映射 (250)
7.14.9 包含变长度元素数组的参数传递 (206)	9.1 本章概述 (250)
7.14.10 对象引用的参数传递 ... (208)	9.2 简介 (250)
7.14.11 参数传递规则的小结 ... (209)	9.3 接口的映射 (251)
7.14.12 使用 _var 类型来传递参数 (210)	9.4 伺服类 (252)
7.14.13 释放 out 参数和使用 _out 类型 的目的 (213)	9.5 对象的实体 (253)
7.14.14 参数的只读性质 (215)	9.6 服务器程序的 main 函数 (254)
	9.7 参数传递规则 (256)
	9.7.1 简单类型的参数传递 (256)
	9.7.2 复杂的定长度类型的参数传递 (257)
	9.7.3 包含定长度元素数组的参数传递 (258)
	9.7.4 字符串和宽位字符串的参数传递 (260)
	9.7.5 复杂的变长度类型和 any 类型

的参数传递	(262)	10.9.3 实现 change 操作	(295)
9.7.6 包含变长度元素数组的参数传递	(265)	10.9.4 实现 find 操作	(296)
9.7.7 对象引用的参数传递	(267)	10.10 实现服务器程序的 main 函数	(298)
9.8 引发异常	(270)	10.11 完整的服务器程序代码	(299)
9.8.1 异常发送的具体细节	(271)	10.11.1 server.hh 头文件	(299)
9.8.2 发送 CORBA 系统异常	(272)	10.11.2 server.cc 实现文件	(302)
9.8.3 管理出现异常的内存	(272)	10.12 本章小结	(310)
9.9 Tie 类	(275)	第 11 章 可移植的对象适配器	(311)
9.9.1 tie 类的具体细节	(275)	11.1 本章概述	(311)
9.9.2 tie 伺服程序的具体化	(276)	11.2 简介	(311)
9.9.3 tie 类的评价	(277)	11.3 POA 基本原理	(311)
9.10 本章小结	(278)	11.3.1 基本的请求调度	(313)
第 10 章 开发气温控制系统的服务器程序	(280)	11.3.2 关键的 POA 实体	(313)
10.1 本章概述	(280)	11.4 POA 策略	(314)
10.2 简介	(280)	11.4.1 CORBA 对象生存期范围	(315)
10.3 仪器控制协议的 API	(280)	11.4.2 对象标识符	(316)
10.3.1 添加和删除装置	(281)	11.4.3 对象到伺服程序之间的映射	(318)
10.3.2 读取属性值	(282)	11.4.4 隐式激活	(320)
10.3.3 写属性值	(282)	11.4.5 请求与伺服程序之间的匹配	(320)
10.4 设计温度计的伺服类	(283)	11.4.6 ObjectId 到伺服程序的关联	(321)
10.5 实现温度计的伺服类	(285)	11.4.7 请求到线程的分配	(322)
10.5.1 Thermometer_impl 辅助函数	(285)	11.4.8 策略工厂操作(Policy Factory Operations)	(323)
10.5.2 Thermometer_impl 的 IDL 操作	(286)	11.5 POA 创建	(324)
10.5.3 Thermometer_impl 的构造函数和析构函数	(287)	11.6 Servant IDL 类型	(327)
10.6 设计恒温器的伺服类	(287)	11.6.1 CCS::Thermometer 伺服程序	(328)
10.7 实现 Thermostat 的伺服类	(289)	11.7 对象创建和激活	(330)
10.7.1 Thermostat_impl 辅助函数	(289)	11.7.1 对象创建	(330)
10.7.2 Thermostat_impl 的 IDL 操作	(291)	11.7.2 伺服程序注册	(336)
10.7.3 Thermostat_impl 的构造函数和析构函数	(291)	11.7.3 伺服程序管理器	(340)
10.8 设计控制器的伺服类	(292)	11.7.4 默认的伺服程序	(351)
10.9 实现控制器的伺服类	(294)	11.7.5 伺服程序内存管理	(356)
10.9.1 Controller_impl 辅助函数	(294)	11.7.6 请求处理	(359)
10.9.2 实现 list 操作	(294)	11.8 引用、ObjectId 和伺服程序	(360)
		11.9 对象失效	(362)
		11.10 请求流控制	(364)

11.11 ORB 事件处理	(367)	12.6.3 使用伺服程序定位器实现收回模型	(416)
11.11.1 阻塞事件处理	(368)	12.6.4 对使用伺服程序定位器的收回模型的评价	(419)
11.11.2 非阻塞事件处理	(368)	12.6.5 使用伺服程序激活器来实现收回模型	(420)
11.11.3 应用程序停止运行	(369)	12.6.6 对使用伺服程序激活器的收回模型的评价	(424)
11.12 POA 激活	(372)	12.6.7 与汇集管理器操作的交互	(425)
11.13 POA 析构	(377)	12.7 伺服程序的无用存储单元回收	(427)
11.14 应用 POA 策略	(378)	12.7.1 客户机意外行为的处理	(427)
11.14.1 多线程问题	(379)	12.7.2 通过关机进行无用存储单元回收	(428)
11.14.2 ObjectId 赋值	(380)	12.7.3 使用收回模型进行无用存储单元回收	(429)
11.14.3 激活	(380)	12.7.4 使用超时进行无用存储单元回收	(429)
11.14.4 时空折衷	(380)	12.7.5 显式保持激活	(430)
11.14.5 关于生命范围的考虑	(382)	12.7.6 每个对象逆向保持激活	(430)
11.15 本章小结	(385)	12.7.7 每个客户逆向保持激活	(431)
第 12 章 对象生命周期	(386)	12.7.8 检测客户的断连	(432)
12.1 本章概述	(386)	12.7.9 分布式引用计数	(432)
12.2 简介	(386)	12.7.10 选择方案小结	(433)
12.3 对象工厂	(387)	12.8 CORBA 对象的无用存储单元回收	(433)
12.3.1 工厂设计选项	(388)	12.8.1 太平洋问题	(434)
12.3.2 用 C++ 实现工厂	(393)	12.8.2 引用完整性	(435)
12.4 撤消、拷贝以及移动对象	(396)	12.8.3 无用存储单元回收的未来	(435)
12.4.1 撤消对象	(398)	12.9 本章小结	(435)
12.4.2 拷贝对象	(405)		
12.4.3 移动对象	(407)		
12.4.4 通用工厂	(408)		
12.5 对生命周期服务的评论	(408)		
12.5.1 设计的通则	(408)		
12.5.2 发布日期	(409)		
12.5.3 使用 move 操作的问题	(409)		
12.5.4 接口的粒度	(411)		
12.5.5 在什么情况下使用生命周期服务	(412)		
12.6 Evictor 模式	(412)		
12.6.1 基本的收回策略	(413)		
12.6.2 维护 LRU 顺序	(415)		

第 3 部分 CORBA 机理

第 13 章 GIOP, IIOP 和 IOR	(436)	13.3 公共数据表示	(437)
13.1 本章概述	(436)	13.3.1 CDR 数据对齐	(438)
13.2 GIOP 概述	(436)	13.4 GIOP 消息格式	(440)
13.2.1 传输假设	(436)	13.4.1 Request 消息格式	(443)

13.4.2 Reply 消息格式	(445)	14.4.5 通过实现仓库的绑定	(460)
13.4.3 其他消息格式	(446)	14.4.6 绑定优化	(462)
13.5 GIOP 连接管理	(447)	14.5 迁移、可靠性、性能和可扩展性	(465)
13.6 检测无序的关闭	(448)	14.5.1 小定位域	(465)
13.7 IIOP 综述	(449)	14.5.2 大定位域	(465)
13.8 IOR 的结构	(450)	14.5.3 冗余的实现仓库	(465)
13.9 双向 IIOP	(452)	14.5.4 对象迁移的粒度	(466)
13.10 本章小结	(453)	14.5.5 跨定位域边界的迁移	(467)
第 14 章 实现仓库和绑定	(454)	14.6 激活模式	(467)
14.1 本章概述	(454)	14.7 竞争状态	(468)
14.2 绑定模式	(454)	14.7.1 激活期间的竞争状态	(468)
14.3 直接绑定	(454)	14.7.2 关闭期间的竞争状态	(469)
14.3.1 暂态引用的直接绑定	(455)	14.7.3 服务器程序关闭和重新绑定	(469)
14.3.2 持久引用的直接绑定	(456)	14.8 安全性考虑	(470)
14.4 通过实现仓库的间接绑定	(457)	14.8.1 服务器程序的权限	(470)
14.4.1 实现仓库的标准一致性	(457)	14.8.2 远程仓库访问	(471)
14.4.2 实现仓库结构	(458)	14.8.3 通过防火墙的 IIOP	(472)
14.4.3 定位域	(459)	14.9 本章小结	(472)
14.4.4 服务器程序和实现仓库之间的相互影响	(460)		
 第 4 部分 动态 CORBA			
第 15 章 any 类型的 C++ 映射 ...	(474)	15.3.10 插入和提取异常	(492)
15.1 本章概述	(474)	15.4 类型定义中易出现的问题	(493)
15.2 简介	(474)	15.5 本章小结	(494)
15.3 any 类型 C++ 映射	(477)	第 16 章 类型代码	(495)
15.3.1 构造函数,析构函数和赋值	(478)	16.1 本章概述	(495)
15.3.2 基本类型	(478)	16.2 简介	(495)
15.3.3 重载不可区分的类型	(480)	16.3 TypeCode 伪对象	(495)
15.3.4 无界的字符串的插入和提取	(482)	16.3.1 适用于所有类型代码的类型和操作	(497)
15.3.5 有界的字符串的插入和提取	(483)	16.3.2 类型代码参数	(498)
15.3.6 宽位字符串的插入和提取	(485)	16.3.3 作为值的类型代码	(502)
15.3.7 定点类型的插入和提取	(485)	16.4 TypeCode 伪对象的 C++ 映射	(503)
15.3.8 用户定义类型	(486)	16.5 类型代码比较	(513)
15.3.9 插入和提取 Any	(491)	16.5.1 TypeCode::equal 的语义	(514)
		16.5.2 TypeCode::equivalent 的语义	(515)

16.5.3 为什么让类型代码中的名称是 可选项	(516)	17.3.3 用于 DynEnum 的 IDL	(536)
16.5.4 类型代码比较的可移植性	(517)	17.3.4 用于 DynStruct 的 IDL	(536)
16.5.5 从 any 类型提取的语义	(517)	17.3.5 用于 DynUnion 的 IDL	(537)
16.5.6 结构上的等价	(518)	17.3.6 用于 DynSequence 的 IDL	(537)
16.5.7 get-compact-typecode 操作	(518)	17.3.7 用于 DynArray 的 IDL	(538)
16.6 类型代码常量	(518)	17.3.8 用于 DynFixed 的 IDL	(538)
16.6.1 内置类型的常量	(518)	17.4 DynAny 伪对象的 C++ 映射	(539)
16.6.2 自定义类型的常量	(520)	17.4.1 简单类型的 DynAny 应用	(539)
16.7 any 类型的类型代码比较	(521)	17.4.2 使用 DynEnum	(541)
16.7.1 控制在 Any 类型中插入的别名 信息	(521)	17.4.3 使用 DynStruct	(543)
16.7.2 检验从 Any 类型中提取的别名 信息	(521)	17.4.4 使用 DynUnion	(546)
16.8 动态创建类型代码	(522)	17.4.5 使用 DynSequence	(549)
16.8.1 用于类型代码创建的 IDL	(522)	17.5 用于通用显示的 DynAny	(549)
16.8.2 类型代码创建的 C++ 映射	(525)	17.6 获得类型信息	(551)
16.9 本章小结	(529)	17.6.1 从 OMG 接口仓库获得类型信息	(551)
第 17 章 DynAny 类型	(530)	17.6.2 从转换表中获得类型信息	(552)
17.1 本章概述	(530)	17.6.3 从表达式获得类型信息	(552)
17.2 简介	(530)	17.7 本章小结	(552)
17.3 DynAny 接口	(530)		
17.3.1 局部约束	(531)		
17.3.2 用于 DynAny 的 IDL ...	(531)		

第 5 部分 CORBA 服务

第 18 章 OMG 命名服务	(553)	18.5.5 名称的等价性	(558)
18.1 本章概述	(553)	18.5.6 绝对与相对名称	(558)
18.2 简介	(553)	18.5.7 名称解析	(559)
18.3 基本概念	(553)	18.6 命名上下文的 IDL	(559)
18.4 命名服务 IDL 的结构	(555)	18.6.1 命名服务中的异常	(559)
18.5 名称的语义	(555)	18.6.2 上下文的生命周期	(561)
18.5.1 名称结构	(555)	18.6.3 获得初始命名上下文	(562)
18.5.2 名称的表达	(556)	18.6.4 创建一个绑定	(563)
18.5.3 kind 字段的作用	(557)	18.6.5 建立一个命名图	(564)
18.5.4 不支持宽位字符串	(557)	18.6.6 重绑定	(567)

18.6.7 取消绑定	(569)	19.5.1 属性	(598)
18.6.8 正确地撤消上下文	(570)	19.5.2 服务类型的继承	(600)
18.6.9 解析名称	(571)	19.5.3 服务类型仓库的 IDL ...	(602)
18.7 迭代器	(573)	19.5.4 在 C++ 内使用服务类型仓库	(608)
18.7.1 使用迭代器的必要性	(573)	19.6 交易接口	(610)
18.7.2 拉迭代器	(574)	19.6.1 主要接口	(612)
18.7.3 推送迭代器	(575)	19.6.2 抽象基接口	(612)
18.7.4 命名服务迭代器	(576)	19.6.3 迭代器	(614)
18.8 命名服务中容易出错的地方	(579)	19.6.4 公共类型	(614)
18.9 名称库	(580)	19.7 导出服务提供源	(614)
18.10 命名服务工具	(581)	19.7.1 export 操作的 IDL 定义	(614)
18.11 怎样公告对象	(581)	19.7.2 导出服务提供源的 C++ 代码	(617)
18.12 公告的时机	(582)	19.7.3 附加属性	(618)
18.13 联邦化命名	(582)	19.8 收回服务提供源	(619)
18.13.1 完全连接的联邦化结构	(583)	19.9 改变服务提供源	(620)
18.13.2 层次化的联邦结构	(584)	19.10 交易程序约束语言	(621)
18.13.3 混合结构	(585)	19.10.1 字面值	(621)
18.14 给气温控制系统增加命名 ...	(586)	19.10.2 标识符	(622)
18.14.1 通用的辅助函数	(586)	19.10.3 比较运算符	(622)
18.14.2 更新气温控制系统的服务器程序	(589)	19.10.4 算术运算符	(623)
18.14.3 更新气温控制系统的客户程序	(590)	19.10.5 布尔运算符	(623)
18.15 本章小结	(591)	19.10.6 集合成员	(623)
第 19 章 OMG 交易服务	(592)	19.10.7 子串的匹配	(623)
19.1 本章概述	(592)	19.10.8 存在性测试	(623)
19.2 简介	(592)	19.10.9 优先权	(624)
19.3 交易的概念和术语	(592)	19.10.10 约束语言的示例程序	(624)
19.3.1 基本的交易概念	(592)	19.11 导入服务提供源	(625)
19.3.2 服务类型和 IDL 接口类型	(593)	19.11.1 Lookup 接口的 IDL	(625)
19.3.3 服务请求	(594)	19.11.2 编制一个简单的查询(Query) 程序	(627)
19.3.4 约束表达式	(594)	19.11.3 OfferInterator 接口	(629)
19.3.5 联邦	(594)	19.11.4 控制 query 返回的细节	(632)
19.3.6 动态属性	(595)	19.11.5 使用优先权	(633)
19.3.7 代理提供源	(595)	19.11.6 导入策略	(634)
19.3.8 优先权	(596)	19.12 成批收回	(637)
19.3.9 策略	(596)	19.13 Admin 接口	(638)
19.4 IDL 概述	(597)	19.13.1 设定配置值	(638)
19.5 服务类型仓库	(597)	19.13.2 检索服务提供源 ID	(639)