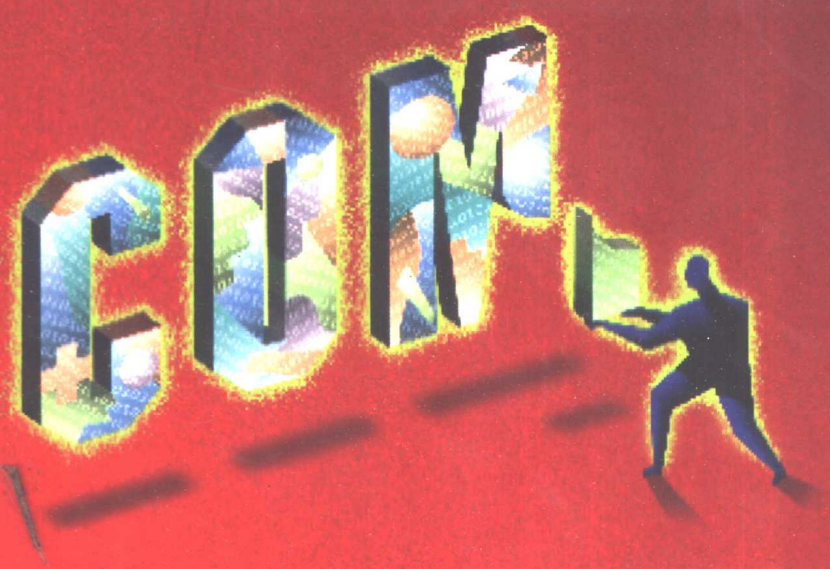
Microsoft®
Windows NT®
Windows®95

CD-ROM
Included

COM
技术内幕

微软组件对象模型



C++程序员设计未来的分布式体系结构的关键：

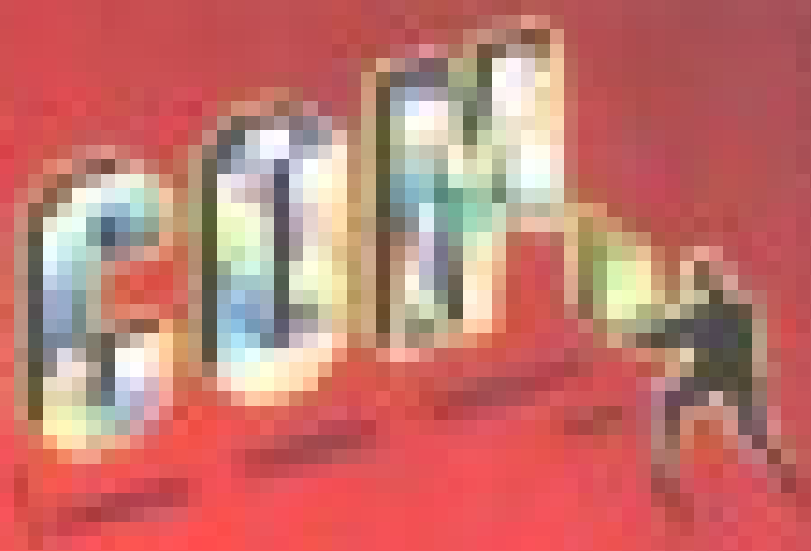
- ActiveX™控件
- OLE组件
- 可复用对象
- 自动化

[美] Dale Rogerson 著
杨秀章 江 英 译
潘爱民 审校

清华大学出版社
http://www.tup.tsinghua.edu.cn

Microsoft® Press

COM技术内幕



[美] Dale Rogerson 著

Microsoft Press

COM技术内幕

——微软组件对象模型

[美] Dale Rogerson 著
杨秀章 江 英 译
潘爱民 审校

清华大学出版社

(京)新登字 158 号

COM 技术内幕

Inside COM

Dale Rogerson

北京市版权局著作权合同登记号 图字 01-98-0270 号

图书在版编目(CIP)数据

COM 技术内幕：微软组件对象模型/(美)罗杰森(Rogerson, D.)著；杨秀章译 .—北京：清华大学出版社，1998.12

ISBN 7-302-03320-X

Ⅰ. C… Ⅱ. ① 罗… ② 杨… Ⅲ. 软件开发 Ⅳ. TP311.52

中国版本图书馆 CIP 数据核字(1999)第 00830 号

出版者：清华大学出版社（北京清华大学学研楼，邮编 100084）
http://www.tup.tsinghua.edu.cn
印刷者：清华大学印刷厂
发行者：新华书店总店北京发行所
开　本：787×960 1/16　**印张**：20　**字数**：422 千字
版　次：1999 年 3 月 第 1 版　2000 年 5 月 第 2 次印刷
书　号：ISBN 7-302-03320-X/TP·1786
印　数：5001～10000
定　价：50.00 元(带光盘)

译者前言

童年时玩过的积木块、七巧板等玩具至今仍时常引起我美好的回忆，那时的我时常沉浸在一种忘我的想象中，用那些简单的形状构造出自己能够想象得到的各种千奇百怪的物体。上了大学，接触到 UNIX，我又深深地为它所提供的各种功能专一、小巧灵活的实用工具所折服；通过管道，我可以把那些精巧的小程序组合起来，构造出各种功能复杂的工具，实现单个程序达不到的功能。

初次接触到 COM（组件对象模型）这个概念，我首先想到的就是积木玩具和 UNIX 中那些小巧的实用程序。人类孜孜以求的目标就是将自然改造得更加符合自己的需要，让自然物按自己的行为习惯和方式那样工作。而 COM 正是微软公司为了使计算机工业的灵魂——软件——的生产更加符合人类行为方式而开发的一种新的软件开发技术。

在 COM 架构下，人们可以开发出各种各样的功能专一的“软件”积木块（组件），然后将它们按照需要“搭”起来，构成复杂的应用系统。由此带来的好处是多方面的：可以将系统中的组件用新的组件替换掉，以随时进行系统的升级与定制；可以在多个应用系统中重复利用同一个组件；可以方便地将应用系统扩展到网络环境下；COM 与语言、平台无关的特性使得所有的程序员均可充分发挥自己的才智与专长编写出组件模块；等等。

Dale Rogerson 是微软公司的资深技术人员，对微软各种技术的背景有着极为深刻的了解。由他所著的《COM 技术内幕》是一部全面、系统介绍 COM 技术的著作。为了及时将 COM 这一最新的软件开发方法与思想介绍给国内的读者，我们决定将这本书翻译出来，以期能够为国内的软件业的发展做出一定的贡献！

由于时间仓促，同时限于对 COM 的认识与理解程度，书中的错误与不足之处在所难免，恳请读者不吝指正。

译　者

1997.12 于北京　清华园

引　言

作为一个软件开发人员，你是否有过在应用程序被发货之后而又想对它进行修改或给它加上一些新的特性呢？你是否想按照逐步添加的方式开发你的程序而不是每两年就将其完全重写一次呢？你是否想使你的应用程序能够在更高的程度上被定制，使应用更灵活、更具有动态性呢？你是否想加速你的应用程序开发进度呢？你是否需要开发分布式应用程序呢？你是否希望能够按开发非分布式应用程序那样开发分布式应用程序呢？

你对组件编程感兴趣吗？你是否想将你的应用程序划分成组件呢？你想学习 COM 吗？你想学习(OLE)自动化吗？你在学习 OLE 的过程中是否被弄得焦头烂额呢？你是否认为 COM 和 OLE 都是极为困难的呢？你是否想了解 Microsoft 的一些技术，如 ActiveX、DirectX 以及 OLE 的基础呢？你是否想定制 Microsoft 的各种应用程序或操作系统呢？

若你对上述问题的答案都是“是”，那么本书对你而言实在是再合适不过了。上述所有的问题都由一种技术联系起来，即 Microsoft 组件对象模型(Component Object Model)，也即大家都熟知的 COM。本书所介绍的就是如何用 C++ 开发自定义的 COM 组件。

COM 是开发软件组件的一种方法。组件实际上是一些小的二进制可执行程序，它们可以给应用程序、操作系统以及其他组件提供服务。开发自定义的 COM 组件就如同开发动态的、面向对象的 API。多个 COM 对象可以连接起来形成应用程序或组件系统。并且组件可以在运行时刻、在不重新链接或编译应用程序的情况下被卸下或替换掉。Microsoft 的许多技术，如 ActiveX、DirectX 以及 OLE 等都是基于 COM 而建立起来的。并且 Microsoft 的开发人员也大量地使用 COM 组件来定制他们的应用程序及操作系统。

预备知识要求

本书是针对具有一些 Win32 编程经验的中级 C++ 程序员而写的。但初级 C++ 程序员也不必因此望而生畏。因为使用 C++ 开发 COM 组件实际上并不是很难，也不需要什么高级的 C++ 技巧。举个例子来说，本书中所用到的最高级的 C++ 功能也不过是多重继承，但为进行 COM 编程，读者所需了解的多重继承的细节将不会超出本书所涉及的那些内容。对于初级 C++ 程序员而言，学习 COM 还有一个好处就是 COM 将帮助他们形成良好的程序设计风格。当然初学者阅读 C++ 方面的一些专门著作肯定也是有极大好处的。

Microsoft Windows 的编程经验对于学习本书所介绍的内容将是有帮助的，但这些经验并不是必需的。在本书中作者尽量避免使用特定于 Windows 的代码。因此，UNIX 用户在理解书中的示例代码也应该是没有问题的。Windows 程序员比之非 Windows 程序员具有的一个优势是他们对开发 Windows 应用的各种工具比较熟悉。

另外要说明的是 Microsoft 基本类库(MFC)及其使用经验并不是必需的。实际上 MFC 并没有给开发 COM 组件提供任何好处。正因如此，本书前 12 章所给出的示例代码中并没有使用 MFC。

写给非 Windows 程序员

对于 UNIX、Macintosh、Linux、VMS 或其他操作系统的开发人员，也将能够从本书所介绍的内容获益。因为 COM 所蕴含的概念并不是只在 Microsoft Windows 操作系统下才有效。COM 并不是一个大的 API，它实际上像结构化编程及面向对象编程方法那样，也是一种编程方法。在任何一种操作系统中，开发人员均可以遵循“COM 方法”。当然，Windows 提供了一些按“COM 方法”进行编程时可以使编程任务得以简化的代码。但这些代码有许多是可以在开发人员所偏爱的操作系统上方便地重新建立的。如果不想自己重新建立这些代码，也并不是说 COM 方法不能用了。Microsoft 正在开发 COM 的 Macintosh 版本，并且 Software AG 正在努力将 COM 移植到几乎所有的现有操作系统上。因此在不久的将来，任何操作系统上的开发人员都将可以使用一个标准的、兼容的 COM 版本。

C++

虽然 COM 本身是与编程语言无关的，但开发人员总得选用一种编程语言来编写所需的组件。可供选择的语言可以是 C、Java、Python，甚至可以是 Microsoft Visual Basic。但是许多组件是用 C++ 编写的，而且这种状况还会继续下去。因此，本书所给出的示例代码均为 C++ 代码。使用同一种语言的好处是可以按照一种具体的方式来讲述 COM 规范，这样将使之更容易理解。即使读者最终决定使用 Java 或 Python 来实现所需的组件，但在用 C++ 从头开发 COM 组件的过程中所获得的知识对于读者也将是有莫大好处的。

只使用传统的 C++ 语言

由于并不是所有的 C++ 编译器都支持加入到 C++ 语言中的最新成分，故作者在本书中将尽量避免使用这些最新的功能，例如在本书中将不会用到 bool 关键字及类似 mutable 之类的关键字。除了在第 9 章中用到的智能接口指针类之外，本书中的其他部分都没有用到模板类，因为模板类将使本书所介绍的概念难以理解。

本书中用到了一些新的类型转换操作符，如 static_case、const_cast 以及 reinterpret_cast 等，这主要是考虑到这些操作符在 Microsoft Visual C++ 中出现的时间已经比较长了。读者可以用这些新的类型转换操作符来代替那些老的类型转换方式。例如可以将

```
CFoo* pI = (Cfoo*)this;
```

写成

```
CFoo* pI = static_cast<CFoo*>this;
```

但在某些情况下，为了保证程序的易读性，作者仍将使用老的类型转换方式。

关于本书中的示例代码

本书中的每一章都包含一到两个示例程序。作者尽量使这些程序简短而完整。短的例子阅读起来比较容易，因为所有代码总共也不过那么几页。对于那些较长的例子，则需要读者阅读本书所附 CD 上的那些文件。使用简单的例子的另一个好处是我们可以将讨论的重点放在 COM 组件的需求上，而无需读者费大力去搞清楚复杂例子中那些不必要的细节及复杂逻辑。

下列特点是本书中所有示例程序所共有的：

- 所有示例代码在 CD 上都是预编译好的，均可以在 Microsoft Windows 95 或 Microsoft Windows NT 上直接运行。
- 所有在正常工作前需要进行系统注册的示例程序都有一个名为 REGISTER.BAT 的批处理文件。使用此文件可以完成所需的注册。
- 所有的示例代码均尽量少用 Win32 API。
- 示例代码均没有用到 Microsoft 基本类。
- 所有示例的完整代码均可在所附的 CD 上找到。为编译这些文件，只需要一个 C++ 编译器及 Win32 SDK 中的头文件及库文件。若读者安装有 Visual C++，则不需要 Win32 SDK，因为 Visual C++ 包含了所需的所有成分。实际上几乎所有 Windows 兼容的编译器都包含这些成分。
- 各示例程序都可以用 Visual C++ 4.x 或 5.0 进行编译。当然这并不是说一定要用 Visual C++ 来编译这些代码。作者在编写这些代码的时候正好在进行 Visual C++ 5.0的开发工作，故选择了 Visual C++ 5.0 来编译它们。
- 所有示例均可以方便地从命令行进行编译。例如若使用的是 Visual C++，则输入 *cl <filename>* 即可完成大多数示例的编译工作。
- 对于复杂一些的例子，作者编写了简单的可供 Visual C++ 使用的 makefile。为编译这些示例代码，可输入 *nmake -f makefile* 或 *nmake*。为了保证简单性和易读性，这些 makefile 只能编译调试版本的示例程序。

Tangram 示例应用程序

在本书所附 CD 中还包含一个使用 COM 组件构造的完整的应用程序：Tangram。但 Tangram 可以说违反了上述大多数的规则。首先它不仅大量地使用了 Win32 API，特别是 GDI，并且它还使用了 MFC 及 OpenGL。其次，它并不简单，其中包含分布在几个 DLL 及 EXE 中的多个组件。可以说 Tangram 向读者展示了“现实世界”中的 COM，而书中的其他示例则更像“象牙塔”中的 COM。Tangram 的源程序及可执行代码均可以在所附 CD 上找到。

代码风格

虽然在大多数例子中并没有使用 MFC，但各示例程序均是按 MFC 代码风格写成的。例如在成员变量的前面均加有前缀 *m_*。这样对于类似于 *m_SleepyBear* 的变量，读者一看就可知道它是一个成员变量。另外所有类的名字前面均有一个大写的字母 *C*。例如 *CCozyBear* 表示的是类 Cozy Bear 的名字。表 I-1 列出了书中所用的其他一些前缀。

表 I-1 MFC 编码风格前缀示例

前缀	含 义	示 例
C	类	*CConnectionPoint*
I	接口	*IConnectionPoint*
*m_*	成员变量	*BOOL m_bSplleyBear;*
*s_*	静态成员变量	*static int s_iBears;*
*g_*	全局变量	*int g_Bears[100];*

有 Windows 编程经验的读者对于匈牙利命名法一定不会陌生。匈牙利命名法实际上是一套用变量类型来标记变量名称的约定。在本书中作者用了匈牙利命名法的一个子集，如表 I-2 所列。可以看到，本书所用的命名方法带有一定的任意性，因为它部分是由其他 COM、OLE 及 ActiveX 的开发人员所建议的匈牙利命名法而得到的。

表 I-2 本书所用匈牙利命名规范

前缀	含 义	示 例
p	指针	*int * pCount;*
pI	指向接口的指针	*Ibear * pIBear;*
b	布尔型	*BOOL bBear;*
I	整型	*int iNumberOfBears;*
dw	DWORD	*DWORD dwBear;*

续表

前缀	含 义	示 例
c	计数	*DWORD cRefs*;
sz	字符数组	*char szName*[ ] = *"Fuzzy"*;
wsz	宽字符数组	*wchar_t wszName*[ ] = *L"Fuzzy"*;

为什么要写这本书?

不知读者学过物理没有? 如果读者修过基于微积分的物理课程,那么很明显,微积分将是一门所需的先修课。在微积分课上,读者将学会如何将这方面的知识应用到其他各种各样的领域中。只有在学会并理解了微积分之后,才能将所得到的这些数学技能应用到物理问题中。但并不是在所有的情况下都需要遵循这种学习次序。实际上当初牛顿发明微积分的主要目的是要用它来解决经典物理中的机械及动力学问题。只是在后来微积分才作为一种有效的工具被用于物理之外的学科中。

COM 和 OLE 的关系非常类似于上面我们所讲到的微积分与物理之间的关系。所不同的是,微积分是用来解决物理问题的,而 COM 则是用来解决诸如如何将一个电子表格嵌入到字处理程序中之类的问题的。此类问题的解决方案正是大家所熟知的 OLE。关于 OLE 的著作已经是相当丰富了,但讲 COM 的却没有。关于 OLE 的第一本最好、最全面的著作当数 Kraig Brockschmidt 的《Inside OLE》了。

当 Kraig 在编写他那本书时,COM 只有唯一的一个应用,即 OLE。任何一个想学习 COM 的人也都需要学习 OLE,这两部分内容是相互关联的。这一点同早期的微积分与物理的关系是类似的。那时候,如果不学习物理,则用不着学习微积分。

到今天,可以说 COM 已经是无处不在了,我们很快可以看到,COM 将变得比 OLE 更为重要。Microsoft 现在已经开发出了许多同 OLE 没有任何关系的 COM 接口及组件,如 Microsoft的三维图形 API Direct3D。正因如此,在 Nigel Thompson 编写《3D Graphics Programming for Windows 95》时,他要在书中加入一章来讲述如何使用 COM,这有点像老师在正式进入物理课程的讲授之前要花一点时间来复习一下微积分一样。这样做的结果是学生并没有理解物理课的内容,他们只是在那儿生搬硬套地处理那些公式。

这本书的目的之一就是将 COM 同 OLE 分开,并重点讲述 COM 的有关概念。在此作者将根本不会涉及到任何有关 OLE 的内容,而只是讲述有关 COM 的一般性机制。在读者学会了这些机制之后,就可以方便地将它们用到 OLE、DirectX 以及 ActiveX 组件的开发中,就如同可以用微积分来解决除物理问题之外的许多其他问题一样。

所以,若读者真的有兴趣搞清楚建立 COM 组件的方法,那么这本书将再适合不过了。读者可以把所学到的技术应用于 ActiveX、OLE 及自己的组件开发中。未来将是 COM 的

大下，而这本书将帮助读者把握未来。至少它可以帮助读者在 C++ 程序中更有效地使用多重继承。

技术支持

虽然作者竭尽全力保证书中内容及所附 CD 上内容的准确性，但错误可能还是在所难免。为此 Microsoft Press 在如下的 WWW 站点提供了关于它所出版的书籍的更正内容：

http://www.microsoft.com/mspress/support/

读者关于本书及所附 CD 的评论、疑问及其他意见可以使用下列方法反馈给 Microsoft Press：

邮件：

Microsoft Press
Attn: *Inside COM* Editor
One Microsoft Way
Redmond, WA 98052 - 6399

电子邮件：

MSPINPUT@MICROSOFT.COM

应说明的是，上述邮件地址并不提供产品支持。当读者需要这方面的支持时，可以使用如下的一些方法。

首先是 Microsoft Developer Network(MSDN) Web 站点，WWW 地址为：

http://www.microsoft.com/MSDN/

为充分利用 MSDN 所提供的各种服务，你可以订阅这些服务。订阅信息可以在此站点获得，或者拨打(800)759-5474。

在下述 Web 站点，Microsoft 提供了大量的支持信息(包括已发现的问题及解决方法等)：

http://www.microsoft.com/support/

有关 COM 的问题，可以打电话给 Microsoft 的 Win32 SDK AnswerPoint 人员，电话号码是(800) 936-5800。

有关 Microsoft Visual C++ 的问题，可以于工作日早六点至晚六点(西部时间)拨打标准的支持电话(206)635 - 7007。

目　录

引言 …… XI
第1章　组件 …… 1
1.1　使用组件的优点 …… 2
1.1.1　应用的定制 …… 2
1.1.2　组件库 …… 3
1.1.3　分布式组件 …… 3
1.2　对组件的要求 …… 4
1.2.1　动态链接 …… 4
1.2.2　封装性 …… 5
1.3　COM …… 6
1.3.1　COM 组件是…… …… 7
1.3.2　COM 不是…… …… 7
1.3.3　COM 库 …… 8
1.3.4　COM 方法 …… 8
1.3.5　COM 超越了用户的需要 …… 8
1.4　本章小结 …… 9
第2章　接口 …… 11
2.1　接口的作用 …… 11
2.1.1　可复用应用架构 …… 12
2.1.2　COM 接口的其他优点 …… 13
2.2　COM 接口的实现 …… 13
2.2.1　代码约定 …… 14
2.2.2　一个完整的例子 …… 15
2.2.3　非接口通信 …… 18
2.2.4　实现细节 …… 18
2.3　接口理论:第二部分 …… 20
2.3.1　接口的不变性 …… 20
2.3.2　多态性 …… 20
2.4　接口的背后 …… 21

2.4.1 虚函数表 …… 21
2.4.2 vtbl 指针及实例数据 …… 23
2.4.3 多重实例 …… 24
2.4.4 不同的类,相同的 vtbl …… 25
2.5 本章小结 …… 26
第 3 章 QueryInterface 函数 …… 27
3.1 接口查询 …… 28
3.1.1 关于 IUnknown …… 28
3.1.2 IUnknown 指针的获取 …… 29
3.1.3 关于 QueryInterface …… 29
3.1.4 QueryInterface 的用法 …… 30
3.1.5 QueryInterface 的实现 …… 31
3.1.6 关于类型转换 …… 32
3.1.7 一个完整的例子 …… 35
3.2 关于 QueryInterface 的实现规则 …… 40
3.2.1 同一 IUnknown …… 40
3.2.2 客户可以获取曾经得到过的接口 …… 41
3.2.3 可以再次获取已经拥有的接口 …… 41
3.2.4 客户可以从任何接口返回到起始接口 …… 42
3.2.5 若能够从某接口获取某特定接口,则从任意接口都将能够获取此接口 …… 42
3.3 QueryInterface 定义了组件 …… 43
3.3.1 接口集 …… 44
3.4 组件新版本的处理 …… 44
3.4.1 何时需要建立一个新版本 …… 46
3.4.2 不同版本接口的命名 …… 46
3.4.3 隐含合约 …… 46
3.5 本章小结 …… 47
第 4 章 引用计数 …… 49
4.1 生命期控制 …… 49
4.2 引用计数简介 …… 50
4.2.1 引用计数接口 …… 53
4.2.2 AddRef 和 Release 的实现 …… 54
4.3 何时进行引用计数 …… 61
4.3.1 引用计数的优化 …… 61

4.3.2 引用计数规则 …… 64
4.4 本章小结 …… 66
第 5 章 动态链接 …… 67
5.1 组件的创建 …… 67
5.1.1 从 DLL 中引出函数 …… 68
5.1.2 DLL 的装载 …… 70
5.2 客户和组件的划分 …… 72
5.2.1 程序清单 …… 73
5.3 对象串 …… 78
5.4 本章小结 …… 79
第 6 章 关于 HRESULT、GUID、注册表及其他细节 …… 81
6.1 HRESULT …… 81
6.1.1 HRESULT 值的查找 …… 83
6.1.2 HRESULT 值的使用 …… 85
6.1.3 用户自己代码的定义 …… 86
6.2 GUID …… 88
6.2.1 为什么要使用 GUID …… 88
6.2.2 GUID 的声明和定义 …… 89
6.2.3 GUID 的比较 …… 91
6.2.4 将 GUID 作为组件标识符 …… 91
6.2.5 通过引用传递 GUID 值 …… 92
6.3 Windows 注册表 …… 92
6.3.1 注册表的组织 …… 92
6.3.2 注册表编辑器 …… 92
6.3.3 CLSID 关键字结构 …… 93
6.3.4 关于注册表的其他细节 …… 94
6.3.5 ProgID …… 95
6.3.6 自注册 …… 97
6.3.7 组件类别 …… 98
6.3.8 OleView …… 99
6.4 COM 库函数 …… 100
6.4.1 COM 库的初始化 …… 100
6.4.2 内存管理 …… 101
6.4.3 将字符串转换成 GUID …… 101
6.5 本章小结 …… 103

第 7 章 类厂 …… 105
7.1 CoCreateInstance …… 105
7.1.1 CoCreateInstance 的声明 …… 106
7.1.2 CoCreateInstance 的用法 …… 106
7.1.3 类环境 …… 107
7.1.4 客户程序清单 …… 108
7.1.5 CoCreateInstance 的不灵活性 …… 110
7.2 类厂 …… 110
7.2.1 CoGetClassObject …… 110
7.2.2 IClassFactory …… 111
7.2.3 CoCreateInstance 与 CoGetClassObject 的比较 …… 112
7.2.4 类厂的若干特性 …… 113
7.3 类厂的实现 …… 113
7.3.1 DllGetClassObject 的使用 …… 114
7.3.2 组件的创建过程 …… 114
7.3.3 组件代码清单 …… 115
7.3.4 流程控制 …… 122
7.3.5 组件的注册 …… 123
7.4 同一 DLL 中的多个组件 …… 124
7.4.1 类厂实现的复用 …… 125
7.5 DLL 的卸载 …… 126
7.5.1 DllCanUnloadNow 的使用 …… 126
7.5.2 LockServer …… 126
7.6 本章小结 …… 127
第 8 章 组件复用:包容与聚合 …… 129
8.1 包容和聚合 …… 130
8.1.1 包容简介 …… 130
8.1.2 聚合简介 …… 131
8.1.3 包容与聚合的比较 …… 131
8.2 包容的实现 …… 132
8.2.1 接口扩展 …… 135
8.3 聚合的实现 …… 136
8.3.1 QueryInterface 的实现 …… 137
8.3.2 不正确的 IUnknown …… 138
8.3.3 聚合的未知接口 …… 140

8.3.4 内部组件的创建 …… 144
8.3.5 外部组件中指向内部组件接口的指针 …… 147
8.4 一个完整的例子 …… 149
8.4.1 盲聚合 …… 165
8.5 现实世界中的聚合和包容 …… 167
8.5.1 组件的内部状态信息 …… 167
8.5.2 虚函数的模拟 …… 169
8.6 本章小结 …… 170
第 9 章 编程工作的简化 …… 171
9.1 客户端的简化 …… 171
9.1.1 智能接口指针 …… 172
9.1.2 C++ 包装类 …… 183
9.2 服务器端的简化 …… 184
9.2.1 unknown 接口基类 …… 185
9.2.2 类厂基类 …… 189
9.2.3 CUnknown 和 CFactory 的使用 …… 195
9.2.4 集成步骤 …… 200
9.3 本章小结 …… 201
第 10 章 EXE 中的服务器 …… 203
10.1 不同的进程 …… 203
10.1.1 本地过程调用 …… 204
10.1.2 列集(marshaling) …… 205
10.1.3 代理/存根 DLL …… 205
10.2 IDL/MIDL 简介 …… 207
10.2.1 关于 IDL …… 207
10.2.2 IDL 接口描述举例 …… 208
10.2.3 MIDL 编译器 …… 213
10.3 本地服务程序的实现 …… 217
10.3.1 示例程序的运行 …… 217
10.3.2 去掉入口点函数 …… 218
10.3.3 类厂的启动 …… 218
10.3.4 对 LockServer 的修改 …… 222
10.4 远程访问能力 …… 224
10.4.1 DCOMCNFG.EXE 所完成的工作 …… 225
10.4.2 工作机理 …… 226

10.4.3 其他 DCOM 信息 …… 227
10.5 本章小结 …… 229
第 11 章 分发接口与自动化 …… 231
11.1 一种新的通信方式 …… 232
11.1.1 旧的通信方式 …… 232
11.1.2 IDispatch 接口 …… 232
11.2 IDispatch 的使用 …… 236
11.2.1 Invoke 函数的参数 …… 238
11.2.2 示例 …… 243
11.2.3 VARIANT 类型 …… 245
11.2.4 BSTR 数据类型 …… 247
11.2.5 SAFEARRAY 类型 …… 248
11.3 类型库 …… 249
11.3.1 类型库的创建 …… 250
11.3.2 类型库的使用 …… 252
11.3.3 注册表中的类型库 …… 253
11.4 IDispatch 接口的实现 …… 254
11.4.1 异常的引发 …… 256
11.4.2 参数列集 …… 257
11.5 本章小结 …… 258
第 12 章 多线程 …… 259
12.1 COM 线程模型 …… 260
12.1.1 Win32 线程 …… 260
12.1.2 COM 线程 …… 260
12.1.3 套间 …… 261
12.1.4 套间线程 …… 263
12.1.5 自由线程 …… 264
12.1.6 列集与同步 …… 264
12.2 套间线程的实现 …… 266
12.2.1 自动列集 …… 267
12.2.2 手工列集 …… 267
12.2.3 编码 …… 268
12.2.4 对套间线程例子的说明 …… 269
12.3 自由线程的实现 …… 276
12.3.1 对自由线程例子的说明 …… 277