www.ncpress.com.cn
新世纪书局
PEARSON

技术经典
著作大系
信息科学

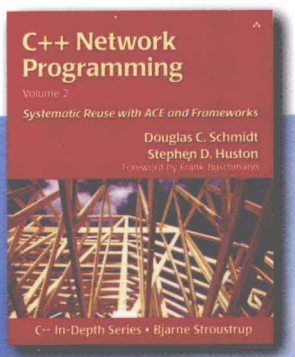
[美] Douglas C. Schmidt, Stephen D. Huston / 编著
王成章 白晓明 彭 雪 / 译

C++网络编程

卷2：基于ACE和框架的系统化复用

C++ Network Programming

Volume 2 Systematic Reuse with ACE and Frameworks



世界级大师提供的专业工具，为您解决并发网络应用的开发难题

科学出版社
PEARSON

C++网络编程

卷2：基于ACE和框架的系统化复用

C++ Network Programming

Volume 2: Systematic Reuse with ACE and Frameworks

[美] Douglas C. Schmidt, Stephen D. Huston 编著

王成章 白晓明 彭 雪 译

科学出版社

图字：01-2012-3454 号

内 容 简 介

自适应通信环境(ADAPTIVE Communication Environment, ACE)软件是一个开源工具包，主要用于构建高性能的网络应用和下一代中间件。面向对象的框架给 ACE 带来了动力和灵活性，使用 ACE 可以实现对网络应用的系统化复用。ACE 框架不但能够处理一般性的网络编程任务，还能够应用 C++编程语言的特征对其进行定制，建立完整的分布式应用。

《C++网络编程 卷 2：基于 ACE 和框架的系统化复用》专注于 ACE 的各种框架，其内容涵盖了构建这些框架结构的概念、模式和使用规则。本书可以作为设计面向对象的框架的实用指南，同时给开发人员展示了如何在并发的网络应用中采用框架结构。《C++网络编程 卷 1：运用 ACE 和模式消除复杂性》介绍了作为网络化计算基本成分的 ACE 和包装器外观方面的知识。卷 2 介绍了如何在包装器外观上构建框架，以提供更高水平的通信服务。

图书在版编目（CIP）数据

C++网络编程. 第 2 卷，基于 ACE 和框架的系统化复用/（美）休斯顿（Huston, S. D.），（美）施密特（Schmidt, D. C.）编著；王成章，白晓明，彭雪译. —北京：科学出版社，2012. 5

ISBN 978-7-03-034198-3

Ⅰ. ①C… Ⅱ. ①休… ②施… ③王… ④白… ⑤彭… Ⅲ. ①C 语言－程序设计 Ⅳ. ①TP312

中国版本图书馆 CIP 数据核字（2012）第 083974 号

责任编辑：何立兵 何 武 / 责任校对：杨慧芳
责任印刷：华 程 / 封面设计：王楠楠

科学出版社 出版

北京东黄城根北街 16 号

邮政编码：100717

http://www.sciencep.com

中国科技出版传媒集团新世纪书局策划

三河市李旗庄少明印装厂

中国科技出版传媒集团新世纪书局发行 各地新华书店经销

*

2012 年 7 月 第 一 版 开本：16 开

2012 年 7 月第一次印刷 印张：21.5

字数：523 000

定价：67.00 元

（如有印装质量问题，我社负责调换）

序 Foreword

自适应通信环境（ADAPTIVE Communication Environment，ACE）软件工具包已经在网络化计算的中间件领域取得了巨大的成功。由于ACE具有很好的灵活性、高性能、平台覆盖和很多其他的关键特性，使得它在网络应用软件领域享有很高的接受度，在成千上万的应用中，在很多国家和大量的领域中都有着ACE的应用。除此之外，ACE还在中间件领域之外获得了相当大的关注度，这是因为它采用了高质量的、设计良好的面向模式的软件架构，而且它是一种开放源代码的角色模型。

但是，为什么ACE会如此成功呢？想要准确地回答这个问题，需要经过一番思考。首先，让我们重新思考一下C++NPv1的前言，并重新以我的同事Steve Vinoski在其中提到的与公共交通系统相类似的东西作为开始。一个高质量的公共交通系统不仅要包含飞机、机场、火车、火车站及轨道，就这一点而言，Steve是正确的。高质量的公共交通系统同样还需要那些表现不是很明显的基础设施，这包括车辆调度、路线安排、票务系统、维护和监控等。然而，即便是将所有的组成成分都收集在一起，对于开发一套有效的公共交通系统也还是不够的。合理地安排这些组成成分，使它们能够无缝地完成它们的主要目标，安全地运送乘客，也是同样重要的事情。如果一个公共交通系统的售票处安排在火车维修处或者飞机机库，或者是一个公共交通系统计划的和实际的调度与路线安排没有对大众公开，那么你会选择这样的公共交通系统吗？我对此表示怀疑！

一个成功的公共交通系统的建立，不但要依靠对所提供的基础设施各个部分的了解，更重要的是这些不同的组成部分应该如何与其周围的环境连接和集成在一起。公共交通系统的架构师掌握了这方面的知识，就能够将那些单独的部分集成到更高水平的结构单元，并将这些结构单元有效地连接在一起。比如说，售票处、信息中心、行李间和进站口都被集成到位于城市中心的火车站，或者是位于主要的城市郊区中心的火车站。类似地，飞机场通常是位于接近大城市的位置，并通过频繁的快速轨道交通相连接。

即便是公共交通系统中心也要进行合理的安排，使中心的各项事宜能够被有效地完成。比如说，当你从主要的入口进入一个火车站或者是飞机场的时候，你就能发现票务代理、信息中心和时间表。你同样也能找到能够满足你的旅行需求的商店。随着你进入主候车室或者飞机场的中央大厅，你还能发现其他的一些信息中心，最新的班次信息和上火车的站台以及登机口。因而，公共交通系统中心不但要

提供旅行出发和到达必需的所有服务，同时还要有效地组织它们内部的控制流。大多数火车站以及飞机场的核心结构以及控制流虽然是相似的，但是，它们的具体实现可能会存在非常大的差别。然而我们仍然能够快速地识别出这些公共交通系统中心的模式，这是因为它们符合一些关键不变的特征，这些特征是我们通过多年的经验学习得来的。

那么，在公共交通系统的成功与ACE的成功之间存在什么联系呢？答案很简单：ACE除了拥有基本的网络化计算的组成成分（Doug和Steve在C++NPv1一书中介绍过的包装器外观），还包括建立在这些包装器外观的基础之上的有用的、面向对象的框架，并能提供一些有用的、更高水平的通信服务，比如事件多路分离和分派（event demultiplexing and dispatching）、连接管理（connection management）、服务配置（service configuration）、并发性（concurrency）和分级的层次流处理（hierarchically layered stream processing）。ACE的框架服务能够满足很多种网络软件的需求，这主要是通过对于你的应用的结构和内部控制流的有效组织来实现的，而实现这种组织则是通过多年的经验所学习到的关键模式来达到的。

ACE框架能够给你带来以下很多重要的益处。

- 你不需要开发ACE所提供的各种功能，而这将为你节省相当可观的时间和精力。因而，你能够专注于自己的关键性任务：实现你的客户和终端用户提出的应用功能。
- ACE框架使得Doug、Steve和他们的同事几十年来所掌握的、大量的网络编程的专业知识得以具体化。尤其是，ACE框架有效地实现了对于网络应用而言常见的、具有典范性的类、类之间的关系和控制流。ACE框架经常由全球各地的、成千上万的用户来测试它的性能，这已经为ACE提供了很多有用的修正和完善。作为一名ACE的用户，你能够直接在你的应用中利用ACE框架的正确性、有效性和高性能。
- 如果一个框架不能够被调整以适应具体的用户需求，那么它就不是真正的框架。这就意味着，你可以在网络应用的各种关键变化点处调整ACE框架，以适应特定的需求。比如说，ACE的Reactor框架就能够通过各种不同的事件多路分离器函数进行调整以适应需求，诸如WaitForMultipleObjects()或者select()函数。类似地，ACE的Acceptor-Connector框架就能够采用不同的

IPC机制来进行配置。虽然这种自适应性本身就很有益处，但ACE还是在此基础上作了更进一步的工作：对于很多的自适应而言，你能够通过各种各样可以利用的、可互换的实现来配置想要的策略。比如说，除了上面所提到的各种不同的Reactor实现之外，ACE还为各种不同的IPC机制提供了外观包装，诸如Sockets、SSL、TLI和内存共享，这些外观包装能够为特定的平台和应用配置ACE的Acceptor-Connector框架提供帮助。

- 最后但并不是最不重要的一条，ACE的各种框架并不是孤立存在的。因而，你能够采用各种新颖的方式来组合它们，以创造出相应的网络应用和完全新式的中间件。比如说，在事件驱动的应用中，你可以将Reactor框架和Acceptor-Connector框架组合在一起，从而将关联的建立从服务处理函数中分离出来。你还可以采用ACE的Task框架在你的应用中引入各种形式的并发机制。

我在过去的多年中指导并且领导了多个软件开发项目，在这个过程中发现ACE能够极大地简化应用可以重复使用的中间件的任务，这些中间件能够很容易被定制出来，以满足网络应用的需求。并非所有的网络应用都需要重量级的中间件，诸如应用服务器、网络服务和复杂的组件模型就不需要它们。然而，大多数网络应用都能够从类似于ACE这样的可移植的、高效率的主机基础架构中间件中受益。这种灵活性正是ACE成功的关键之处，这是因为如果你不需要使用一整套中间件的所有功能，那么你就没有必要去关注所有的中间件。相反，你可以只是将你所需要的那些基本的ACE中间件类组合在一起，来构建小巧的、但是功能能够满足需求的应用。正是出于这个原因，我预测ACE在今天的重量级中间件的影响消逝之后的很长时间内，仍然将会被广泛使用。

ACE无穷的灵活性也不会导致大量的、不兼容中间件的实现。比如说，如果你要创建一个采用CORBA Internet inter-ORB protocol（IIOP）与外部世界进行通信的嵌入式系统，你就可以采用The ACE ORB（TAO）来实现，TAO是一种符合CORBA规则的、开放源代码的、实时的对象请求代理（ORB），它是采用ACE的包装器外观和框架构建的。然而，如果CORBA对于你的应用需求来讲过于强大，那么你也可以采用一些适当的ACE类来建立定制的，但仍然是可互操作的中间件。两种解决方案都能够建立在相同的核心结构和协议的基础上，诸如ACE那些Common Data Representation（CDR）类和它的TCP/IP Socket包装器外观。因而，它们彼此之间能够实现无缝的通信，正如你可以乘坐著名的从法国巴黎驶向伊斯坦布尔的“东

方快车”穿越很多欧洲国家，却不需要因为铁路网络的彼此不兼容而不得不更换火车。

就像Steve Vinoski和我所提到的那样，在高质量的公共交通系统和高质量的网络化中间件之间存在着很多的相似之处。而ACE对于我和世界各地成千上万的其他C++程序开发人员来讲，就是构建高质量的网络化中间件的工具包。但是，在说过ACE这么多的好处之后，让我们重新回到这篇前言的主要目的上来，我们主要是为了介绍C++网络编程系列丛书的第2卷（C++NPv2）。正像所有的软件技术和中间件一样，你对于自己的工具理解得越透，那么你将会越能够更好地应用它们。事实证明，在你的应用中使用ACE仅仅是提高你的网络软件性能的一个方面。如果想要从ACE众多的优点中大大获益，那么你还必须充分理解作为其强有力的框架的支撑的概念、模式和使用规则。

多年以来，一种常用的学习ACE的方式就包括学习ACE的代码、注释和示例应用。毫无疑问，这一学习过程需要花费大量的时间，并且很容易出错。不仅如此，即便是你设法阅读了ACE中数十万行的C++代码，也很有可能是“只见树木，不见森林”。正如古希腊的哲学家Thucydides在2000年前说过的那样：“如果一个人拥有知识，但缺乏清楚表达自己的能力，那么这个人几乎就如同从来没有过任何思想一样。”

因此，我们非常幸运，Doug和Steve从他们繁忙的工作安排中抽出宝贵的时间，撰写了这样一本关于ACE框架的高质量的图书。C++NPv2一书描述了作为ACE框架支撑基础的思想和概念，全书采用了一种来自于POSA[POSA1, POSA2]和“Gang of Four”[GoF]模式丛书的、流行的并发性和网络化模式，以一种易于理解的形式将所要介绍的内容娓娓道来。反过来，这些模式又使得那些常见的网络问题中具有思想性的、被时间证明了的解决方案得以具体化。比如说，它们就告诉你问题是什么，为什么这些问题很困难，能够解决这些问题的方案是什么，以及为什么这些应用于ACE的解决方案是高质量的。如果你想要彻底了解ACE中所有的模式和框架，那么就阅读本书吧。ACE中的这些模式和框架正在形成下一代网络应用软件。我早已经从这本书中学到了很多知识，并且我确信你也会像我一样。

Frank Buschmann
高级首席工程师
西门子技术公司
慕尼黑，德国

关于本书 About This Book

在当今竞争激烈、快速发展的计算技术中，网络应用软件必须具备下列品质才能够立于不败之地。

- **可负担性（affordability）**：确保软件购置和升级的总费用不致于高到难以承受。
- **可扩展性（extensibility）**：支持连续的快速更新和发展，以应对新的需求并抢占新兴市场。
- **灵活性（flexibility）**：支持规模不断增长的多媒体数据类型、传输模式，以及端到端的服务质量（QoS）需求。
- **可移植性（portability）**：降低在不同种类的操作系统平台和编译器上支持各种应用所需要的工作量。
- **可预测性（predictability）和高效性（efficiency）**：对于延迟敏感的、实时的应用，能够提供低延迟；对于带宽密集的应用，能够提供高性能，能够在诸如无线连接的低带宽的网络上提供可用性。
- **可靠性（reliability）**：确保各种应用是健壮的、容错的、高度可用的。
- **可伸缩性（scalability）**：使得应用能够同时处理大量的客户端。

想要编写出高质量的网络应用，展示出软件的这些品质是非常困难的，这个过程代价昂贵、复杂，且容易犯错。在C++NPv1一书中介绍的模式、C++语言的特征和面向对象的设计规则，可以帮助人们在网络应用中将复杂度和出现的错误降低到最小，这种功能的实现是通过将常见的结构和功能在可复用的包装器外观类库中进行重构来完成的。然而，如果每一个项目中大部分使用这些类库的应用软件都必须重写，或者更糟糕的，这些类库本身都必须重写，那么我们将会失去复用带来的关键益处。

从传统上来讲，很多网络应用软件项目都是以此作为开端的：

（1）设计并实现多路分离机制和分派基础机制，用来处理定时事件以及多个socket句柄上的I/O事件。

（2）在多路分离层和分派层之上增加服务的实例化和处理机制，并随之一起增加信息缓冲和队列机制。

（3）应用这种特殊的主机基础中间件来执行大量的、与特定应用相关的代码。

很多公司已经多次在很多项目中并行地采用这种开发过程。更糟糕的是，同一个团队在一系列项目中都采用这样的开发模式。可悲的是，这种对核心概念和代码

的连续的重新发现和重复发明，已经使得在整个软件开发生命周期内的代价产生了不必要的增加。这一问题因为当今的硬件、操作系统、编译器和通信平台固有的多样化而变得更加恶化，这使得网络应用软件开发的基础不断地发生变化。

面向对象的框架[FJS99b, FJS99a]是用来处理上面所列出的问题中最灵活且最强有力的技术之一。一个框架是一种可复用的、半完成的应用，它可以被特定化，以生成一些定制的应用[JF88]。各种框架能够帮助人们减少网络应用的开发成本，提高其质量，这主要是通过将被公认的软件设计和模式具体化到实际的源代码中来实现的。各种框架强调的是与应用相关的类以及与独立于应用的类之间的集成与协作，通过这种方式实现更大规模的软件复用，这种规模化要比复用单独的类或者独立的函数带来更大的益处。

在20世纪90年代早期，Doug Schmidt开始了开放源代码的ACE项目，从而将模式和框架的力量与效率带给了网络应用的开发。基于Doug的大量工作，ACE致力于处理专业的、软件开发人员所面临的很多实际问题。在随后的十年里，在加利福尼亚大学Irvine分校、圣路易斯华盛顿大学和范德比尔特大学的Doug的团队，开发出了一个包含这个世界上最为强大的、被广泛应用的、具有并发性的面向对象的网络编程框架的C++工具包，这其中也包括了ACE用户群以及Riverace的Steve Huston所作出的贡献。ACE工具包中的各种框架采用可复用的软件模式和一种轻量级的操作系统可移植层，来实现同步和异步事件处理（synchronous and asynchronous event processing）、并发和同步（concurrency and synchronization）、连接管理（connection management）和服务配置，初始化以及分级的集成功能（hierarchical integration）。

ACE的成功已经从根本上改变了在补充栏2中所列出的很多操作系统上，人们设计和实施网络应用以及中间件的方式。ACE正在被成千上万的开发团队所使用，这些团队包括大型的财富500强公司、小型的新型企业，还有大学以及工业实验室的高级研究团队。ACE开放源代码的模型和自我支持的文化，与促使Linus Torvalds的流行的Linux操作系统发展的因素在精神和积极性方面是类似的。

本书描述了各种ACE框架是如何设计的，它们是怎样帮助开发人员行走在下面的两种限制之间。

（1）**低级的本地操作系统的API**。这些API既不具备灵活性，又不具备可移植性。

（2）**高级中间件**。诸如分布式中间件和通用中间件服务，为了支持能够满足要求严格的QoS以及可移植性需求的网络应用，这些高级中间件往往缺乏效率和灵活性。

创建和使用网络应用框架的技能通常是为专门的开发人员所特有的，或者是深藏于大量项目的源代码中，这些项目却散布于整个公司或者企业当中。当然，任何一种情况都是不理想的，这是因为对于任何一个新的应用或者新的项目来讲，这种知识的重新设计都是非常消耗时间且易于出错的。本书为了处理这一问题，阐明了作为ACE框架之基础的结构和功能的那些关键模式[POSA2, POSA1, GoF]。我们对于这些模式的覆盖，同样也使对开放源代码的ACE工具包本身的设计、实施和有效使用的理解变得更加容易。

目标读者

本书的目标读者是正从事C++编程的开发人员，或者是想要了解如何设计面向对象的框架，并将它们应用于网络应用的高水平学生。本书是在C++NPv1一书中列出的材料的基础上组织起来的，这些材料向我们展示了开发人员是如何应用模式来克服由于使用本地操作系统的API进行网络应用编程而带来的复杂性的。所以，非常重要的一点是，在阅读本书之前要对C++NPv1一书中所涉及的下列主题有一个实实在在的领悟。

网络应用设计空间：包括在C++NPv1一书的第1章中讨论过的各种通信协议和数据传输机制。

Internet编程机制：诸如在C++NPv1一书的第2章中讨论过的TCP/IP连接管理以及数据传输API[Ste98]。

并发设计空间：包括在C++NPv1一书的第5章中讨论过的进程和线程的使用、循环式服务器、并发式服务器、反应式服务器，以及各种线程模型[Ste99]。

同步技术：这些技术指的是在C++NPv1一书的第10章中讨论过的，为了协调各种不同的操作系统平台上的进程和线程之间的交互而必须使用的技术[KSS96, Lew95, Ric97]。

面向对象的设计和编程技术：这些技术指的是在C++NPv1一书的第3章以及附录A中讨论过的，能够简化操作系统上的API，并且通过使用诸如包装器外观[POSA2]和代理[POSA1, GoF]等各种模式来避免编程错误的技术[Boo94, Mey97]。

关于本书 About This Book

在很大程度上由于使用了C++编程语言的特征[Bja00]，ACE框架都具有很高的灵活性和强大的功能。所以，你应该熟悉C++类的继承和虚函数（动态绑定）的概念，同时包括C++的模板（参数化类型）以及你的编译器提供的对这些模板进行实例化的机制。ACE为使用者提供了大量的帮助去克服不同的C++编译器之间的差别。然而，无论如何，你都需要了解你的开发工具的功能，并知道如何使用它们。了解你的开发工具可以令你更容易理解本书中提供的源代码示例，以及在你的系统上建立并运行它们。最后，在你阅读本书中提供的示例时，要紧记补充栏7中提到的关于UML图表和C++代码的要点。

结构和内容

我们的C++NPv1一书专注于如何克服在开发网络应用过程中带来的某些复杂性，关注的焦点是如何使用ACE的各种包装器外观来避免采用C语言编写的操作系统API存在的问题。本书（我们称之为C++NPv2）将我们的关注点转移到与开发和使用ACE的各种框架相关的模式、设计技术和C++语言特征，专注于介绍它们的动机，消除它们的神秘感。这些框架可以帮助我们减少网络应用的开销，并提升网络应用的质量，ACE通过在框架中具体化已经被验证过的软件设计和模式，使得那些能够在不同的项目和企业中系统化复用的框架具备这种能力。ACE的各种框架拓展了复用技术，其复用产生的效果要远超过复用单独的类，甚至是复用类库所能达到的程度。

本书通过展示如何使用ACE框架的具体例子来强化对设计的讨论，其中提供了大量的C++应用。这些例子详细地给出了每一步的指导，这些指导能够帮助你在自己的网络应用中使用关键的面向对象的技术和模式。本书还向你展示了如何提高你的设计技巧，关注的焦点是关键的概念和规则，正是它们造就了用于网络应用和中间件的、成功的面向对象的框架设计。

本书各章的组织如下。

- **第1章**：介绍面向对象的框架的概念，展示各种框架是如何区别于其他的复用技术的，比如说类库、组件、模式和模型集成式计算。接着，我们列出了随后章节中涉及的ACE工具包中的各种框架。
- **第2章**：完善在C++NPv1一书中开始讨论的领域分析，包括通信协议、通

信机制和网络应用中采用的并发架构。本书关注的焦点是服务和配置设计空间，这些空间处理的是网络应用的关键特性，诸如持续时间和结构，如何标识网络服务，以及这些服务是在何时被绑定在一起形成完整的应用的。

- **第3章**：描述了ACE Reactor框架的设计和使用，这一框架实现了Reactor模式[POSA2]，该模式允许事件驱动的应用多路分离和分派服务请求，这些服务请求可以是从一个或者多个客户端传递到一个应用的。
- **第4章**：描述了ACE_Reactor接口的最为常见的实现的设计和应用，这一接口广泛支持大量操作系统的事件多路分离机制，包括select()、WaitForMultipleObjects()、XtAppMainLoop()和/dev/poll。
- **第5章**：介绍ACE Service Configurator框架的设计和使用。这一框架实现了Component Configurator模式[POSA2]，允许一个应用在不进行静态的修改、重新编译或者重新连接的前提下，在运行时就可以连接/断开它的组件服务的执行。
- **第6章**：介绍了ACE Task框架的设计和使用。这一框架能够被应用于诸如Active Object和Half-Sync/Half-Async[POSA2]之类的关键并发模式的具体实现。
- **第7章**：介绍了ACE Acceptor-Connector框架的设计和使用。这一框架实现了Acceptor-Connector模式[POSA2]，用以将一个网络系统中协同操作的对等服务的连接和初始化，从这种连接和初始化一旦建立后它们就必须执行的处理中解耦出来。
- **第8章**：介绍了ACE Proactor框架的设计和使用。这一框架实现了Proactor和Acceptor-Connector模式[POSA2]，允许事件驱动的应用有效地实现多路分离和分派由异步发起的操作的完成而触发的服务请求。
- **第9章**：介绍了ACE Streams框架的设计和使用。这一框架实现了Pipes和Filters模式[POSA1]，用于为那些处理数据流的系统提供一种结构。

本书中各个章节都是在其前一章节的基础上组织起来的，并且将提前引用做到了最小。所以，我们建议你按照章节的顺序阅读本书。

纵然本书向大家展示了ACE最为重要的那些框架的关键功能，但是我们并未覆盖到那些框架的所有的用途和方法。如果想要了解ACE更多的知识，我们向你推荐*The ACE Programmer's Guide*[HJS]和ACE在线参考文档，该文档是由Doxygen[Dim01]生成的。ACE的参考文档可以在http://www.riverace.com/docs/网站上得到。

关于本书 About This Book

相关材料

本书是基于ACE 5.3版本撰写的，该版本的ACE在2002年秋天发布。ACE 5.3和我们书中所给出的所有示例应用都是开放源代码的软件。补充栏3的内容阐释了怎么样你才能得到ACE的一个副本，从而你能够跟得上本书进度，彻底详细地查看真正的ACE的类和框架，并且在你阅读本书的过程中交互地执行那些代码示例。

如果想要学习更多的关于ACE的知识，或者是想要报告你在本书中发现的错误，我们推荐你去订阅ACE邮件列表：ace-users@cs.wustl.edu。你可以通过向ace-users-request@cs.wustl.edu发送一份请求信来订阅此邮件列表。在电子邮件的正文内容部分应包括如下的命令（邮件的标题会被忽略）：

subscribe ace-users [emailaddress@domain]

当你发送消息的From地址不是你想要订阅的地址时，那么你就必须提供emailaddress@domain。如果你是采用的这种替代地址的方法，那么列表服务器将会在允许你加入到邮件列表之前要求进行一项额外的认证。

发送到ace-users列表的内容也会被发送到comp.soft-sys.ace USENET新闻组，发送到其他几个与ACE相关的邮件列表的内容也是如此。如果你不需要马上就看到邮件列表上每天发送的30～50条消息，一种很好的与ACE的新闻和活动保持同步的方法就是通过新闻组来了解信息。

在网站http://groups.google.com/上可以找到发送到comp.soft-sys.ace新闻组的消息的存档。在搜索框中输入comp.soft-sys.ace就可以找到存档消息的列表。Google有着一个完整的、可以搜索的ACE存档，其中的消息超过40 000条。你也可以通过Google的站点给新闻组发送消息。

致谢

要向Alain Decamps、Don Hinton、Alexander Maack、Chris Uzdavinis和Johnny Willemsen致以最高的评阅人的荣誉，他们多次评阅本书并且提出了广泛、详细的建议，这些建议对于本书形式和内容的改进给予了极大的帮助。对于官方的评阅人员Timothy Culp、Dennis Mancl、Phil Mesnier和Jason Pasion，同样也要致以十分的感谢，他们评阅了整本图书并且给我们提出了很多有帮助的建议。包括Marc M.

Adkins、Tomer Amiaz、Vi Thuan Banh、Kevin Bailey、Stephane Bastien、John Dilley、Eric Eide、Andrew Finnell、Dave Findlay、Jody Hagins、Jon Harnish、Jim Havlicek、Martin Johnson、Christopher Kohlhoff、Alex Libman、Harald Mitterhofer、Llori Patterson、Nick Pratt、Dieter Quehl、Tim Rozmajzl、Irma Rastegayeva、Eamonn Saunders、Harvinder Sawhney、Christian Schuhegger、Michael Searles、Kalvinder Singh、Henny Sipma、Stephen Sturtevant、Leo Stutzmann、Tommy Svensson、Bruce Trask、Dominic Williams和Vadim Zaliva在内的很多其他的ACE用户针对本书给我们提供了很多反馈意见。

我们要对圣路易斯华盛顿大学和加利福尼亚大学Irvine分校的DOC团队的所有成员，包括以前的成员和现在的成员，致以深深的感激之情，同样的感激还要给予Riverace Corporation和Object Computing Inc.的团队成员，他们开发、提炼、优化了本书中提到的很多ACE功能。这个团队的成员包括Everett Anderson、Alex Arulanthu、Shawn Atkins、John Aughey、Luther Baker、Jaiganesh Balasubramanian、Darrell Brunsch、Don Busch、Chris Cleeland、Angelo Corsaro、Chad Elliot、Sergio Flores-Gaitan、Chris Gill、Pradeep Gore、Andy Gokhale、Priyanka Gontla、Myrna Harbibson、Tim Harrison、Shawn Hannan、John Heitmann、Joe Hoffert、James Hu、Frank Hunleth、Prashant Jain、Vishal Kachroo、Ray Klefstad、Kitty Krishnakumar、Yamuna Krishnamurthy、Michael Kircher、Fred Kuhns、David Levine、Chanaka Liyanaarachchi、Michael Moran、Ebrahim Moshiri、Sumedh Mungee、Bala Natarajan、Ossama Othman、Jeff Parsons、Kirthika Parameswaran、Krish Pathayapura、Irfan Pyarali、Sumita Rao、Carlos O'Ryan、Rich Siebel、Malcolm Spence、Marina Spivak、Naga Surendran、Steve Totten、Bruce Trask、Nanbor Wang和Seth Widoff。

我们还要对来自超过50个国家的成千上万的C++开发人员表示感谢，感谢他们十多年来为ACE所作出的贡献。ACE的非凡功能和成功是很多优秀的开发人员和具有前瞻性的公司的技术和慷慨奉献的见证，他们远见性地将自己的工作奉献给了ACE的开放源代码库。正是在他们的支持、持续的反馈和激励下，我们才能写出这本书。为了表示对ACE开放源代码的团体的工作的认可，我们在http://ace.ece.uci.edu/ACE-members.html保留了一个囊括所有贡献者的名单。

关于本书 About This Book

我们还要对我们的同事，以及资助我们对模式和ACE工具包进行开发研究的人士表示感谢。尤其是下面这些作出贡献的人员：Ron Akers（Motorola）、Steve Bachinsky（SAIC）、John Bay（DARPA）、Detlef Becker（Siemens）、Frank Buschmann（Siemens）、Dave Busigo（DARPA）、John Buttitto（Sun）、Becky Callison（Boeing）、Wei Chiang（Nokia Inc.）、Joe Cross（Lockheed Martin）、Lou DiPalma（Raytheon）、Bryan Doerr（Savvis）、Karlheinz Dorn（Siemens）、Scott Ellard（Madison）、Matt Emerson（Escient Convergence Group, Inc.）、Sylvester Fernandez（Lockheed Martin）、Nikki Ford（DARPA）、Andreas Geisler（Siemens）、Helen Gill（NSF）、Jody Hagins（ATD）、Andy Harvey（Cisco）、Sue Kelly（Sandia National Labs）、Gary Koob（DARPA）、Petri Koskelainen（Nokia Inc.）、Sean Landis（Motorola）、Patrick Lardieri（Lockheed Martin）、Doug Lea（SUNY Oswego）、Joe Loyall（BBN）、Kent Madsen（EO Thorpe）、Ed Margand（DARPA）、Mike Masters（NSWC）、Major Ed Mays（U.S. Marine Corps）、John Mellby（Raytheon）、Jeanette Milos（DARPA）、Stan Moyer（Telcordia）、Ivan Murphy（Siemens）、Russ Noseworthy（Object Sciences）、Adam Porter（U. of Maryland）、Dieter Quehl（Siemens）、Vijay Raghavan（Vanderbilt U.）、Lucie Robillard（U.S. Air Force）、Craig Rodrigues（BBN）、Rick Schantz（BBN）、Andreas Schulke（Siemens）、Steve Shaffer（Kodak）、Tom Shields（Raytheon）、Dave Sharp（Boeing）、Naval Sodha（Ericsson）、Paul Stephenson（Ericsson）、Tatsuya Suda（UCI）、Umar Syyid（Storetrax, Inc.）、Janos Sztipanovits（Vanderbilt U.）、Gautam Thaker（Lockheed Martin）、Lothar Werzinger（Krones）和Don Winter（Boeing）。

我们还要对我们的文字编辑Susan Cooper致以特别的谢意，正是她为我们图书的内容进行润色加工。我们还要对我们的编辑Debbie Lafferty、我们的出版协作人Elizabeth Ryan，以及丛书编辑和C++创造者Bjarne Stroustrup，还有Addison-Wesley出版社其他任何一位为本书的出版作出贡献的人士表示感谢，谢谢他们的鼓励和耐心。

最后，我们还要将我们的感谢和感激之情献给已故的网络编程著作之父W. Richard Stevens。由Samuel Butler题写的下面这首诗歌概括了我们对Richard的持久影响的观点。

Not on sad Stygian shore, nor in clear sheen
Of far Elysian plain, shall we meet those
Among the dead whose pupils we have been...
Yet meet we shall, and part; and meet again,
Where dead men meet, on lips of living men.

Steve的致谢

哇哦……C++NPv1差不多用了3年的时间才写完，这一卷书花费了大约9个月的时间。要感谢我的妻子Jane，她高兴地渡过了这一段时间。你不断的劝导我要保持生活的平衡，要做龟兔赛跑中的乌龟，正是这些劝导帮助我坚持到底。如果没有你这么长时间无限的耐心，我可能也无法完成这本书，谢谢你！要感谢Doug Schmidt，他不但撰写了这本书的大部分内容，还以最快的速度对全书的内容进行了编排，要知道，这些工作都是在他完成全职工作，以及日常的、数量惊人的与ACE相关的工作之余做完的。最后，要对如此热心地支持这一工作的Riverace的顾客表示感谢。为你们服务是一种特殊的待遇。

Doug的致谢

我的妻子Sonja和我的父母在撰写这本书期间给予了我很大的关爱和支持，我要谢谢他们。现在，这本书已经写完，我们将会有更多的时间一起愉快地生活。同样，还要感谢Steve Huston，他从本已紧张的时间表中抽出了时间来撰写这本书。我还想感谢我在威廉和玛丽大学、圣路易斯华盛顿大学、加利福尼亚大学Irvine分校、范德比尔特大学、DARPA，以及西门子的朋友和同事们，还有成千上万的ACE和TAO的开发人员及全世界的用户，他们在过去的20年里，极大地丰富了我的脑力生活与人际关系。我希望在将来能与你们所有人一起工作。

目 录