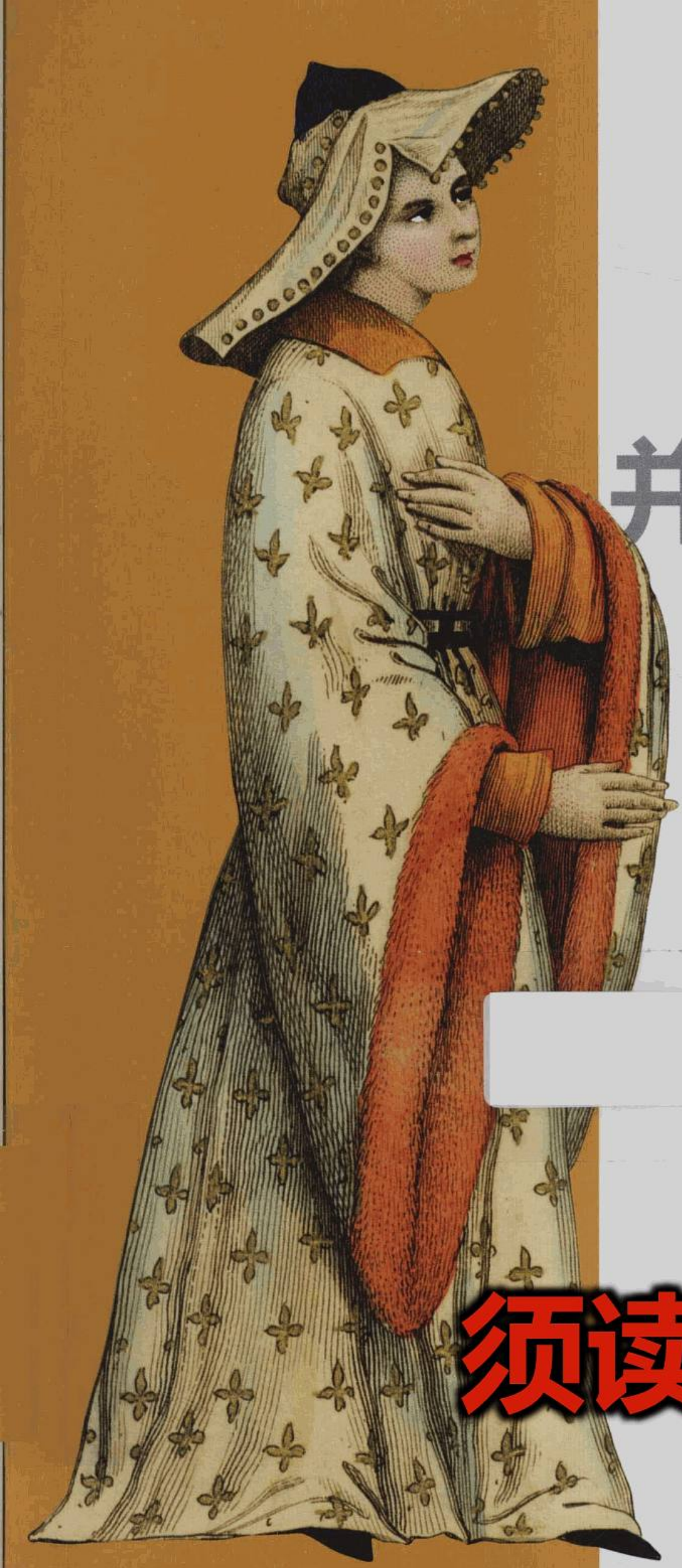


MANNING

C++ 并发编程实战

C++ Concurrency IN ACTION

〔美〕Anthony Williams 著
译

翻译灾难
须读英文原著

人民邮电出版社
POSTS & TELECOM PRESS

C++ 并发编程实战

具有多核的多处理器现已成为标配。C++ 语言的 C++11 版本为多线程应用程序提供了强大的支持，你需要掌握其原理、技巧以及新的并发语言特性，才能独领风骚。

本书帮助你循序渐进地学习用 C++11 编写健壮且优雅的多线程应用程序。你将学习线程内存模型、新的线程支持库，以及基础的线程启动和同步功能。与此同时，你还将学到如何解决并发应用程序中的棘手问题。

本书具有以下特色：

- 针对 C++
- 针对多核
- 用于学习

本书适合新接触并发编程的 C++ 程序员，以及曾经使用别的语言

Anthony Williams 拥有十余年的 C++ 经验，并且是 BSI C++ 专家组的成员。

“有思想、有深度的指南，从专业人士那儿来的第一手资料。”

——Neil Horlock，Credit Suisse

“简化了 C++ 多线程的黑魔法”。

——Rick Wagner，Red Hat

“读这本书让我头痛，但痛定思痛”。

——Joshua Heyer，Ingersoll Rand

“作者展示了如何将并发变为现实。”

——Roger Orr，OR/2 Limited

MANNING

美术编辑：董志桢

分类建议：计算机／程序设计／ Web 设计
人民邮电出版社网址：www.ptpress.com.cn

ISBN 978-7-115-38732-5

ISBN 978-7-115-38732-5

定价:89.00 元

内容提要

本书是一本基于 C++11 新标准的并发和多线程编程深度指南。内容包括 std::thread、std::mutex、std::future 和 std::async 等基础类的使用，内存模型和原子操作、基于锁和无锁数据结构的构建，以及并行算法和线程管理，最后还介绍了多线程代码的测试。本书的附录部分还对 C++11 新语言特性中与多线程相关的项目进行了简要的介绍，并提供了 C++11 线程库的完整参考。

本书适合于需要深入了解 C++多线程开发的读者，以及使用 C++进行各类软件开发的开发人员、测试人员阅读。使用第三方线程库的读者，也可以从本书后面的章节中了解到相关的指引和技巧。同时，本书

序

我是在离开大学后的第一份工作中遇到多线程编程的概念的。我们当时在编写一个数据处理应用程序，它需要用传入的数据记录来填充数据库。虽然数据很多，但每个数据都是独立的，并且在它被插入数据库前需要大量的处理。为了充分利用 10-CPU UltraSPARC 的能力，我们在多线程中运行代码，让每个线程处理它们自己的一组输入数据。在这个过程中，我们用 C++编写代码，使用 POSIX 线程，并且犯了相当多的错误。尽管多线程对我们而言都是全新的，但我们最终还是完成了。正是在这个项目中的工作，我第一次了解了 C++标准委员会和全新发布的 C++标准。

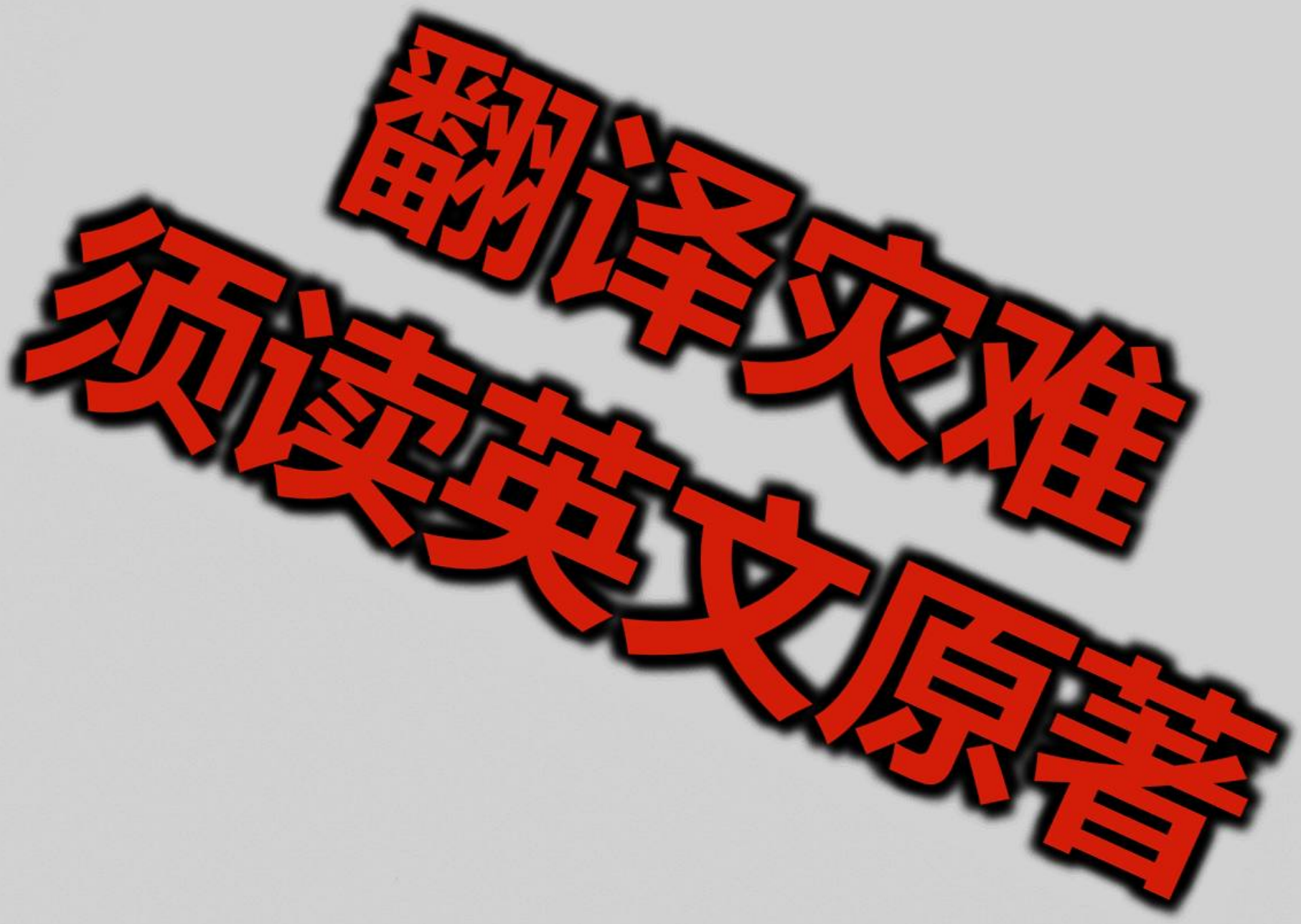
我对多线程和并发有前所未有的兴趣。虽然别人认为它困难、复杂，是问题之源，但我却视它为强大的工具，因为它可以允许你利用现有的硬件让代码运行得更快。随后我了解到它即便是在单核硬件上也能提高应用程序的响应和性能，通过使用多线程来隐藏诸如 I/O 这样耗费时间的操作延迟。我还了解了它怎样在 OS 级别工作以及 Intel CPU 如何处理任务切换。

同时，我对 C++的兴趣给我带来了与 ACCU 以及 BSI 的 C++标准专家组以及 Boost 接触的机会。我凭兴趣跟进了 Boost 线程库的初期开发，当它被最初的开发者抛弃时，我便趁机介入了。我从此成为了 Boost 线程库最主要的开发者和维护者。

当 C++标准委员会的工作从修正现有标准的缺陷转移到编写下一代标准（希望在 2009 年之前完成故命名为 C++0x，现在正式为 C++11，因为最终发布于 2011 年）的提案后，我更多地参与到 BSI 并且开始起草我自己的提案。多线程刚被明确地提上议事日程，我就全力以赴地投入进去，并且撰写或共同撰写了许多与多线程和并发相关的提案，它们组成了新标准的一部分。我感到很荣幸，可以有这个机会用这种方式组合我的计算机相关的两大兴趣——C++和多线程。

这本书借鉴了我在 C++和多线程上的全部经验，其目标是教会其他 C++开发者如何安全并有效地使用 C++11 线程库。我也希望能顺带传授一些我在这方面的热情。

译者简介



致 *Kim*、*Hugh* 和 *Erin*。

致谢

我要首先对我的妻子 Kim 说一声“谢谢”，感谢在我写这本书的时候她给我的爱和支持。写书占用了我最近 4 年很多的空闲时间，没有她的耐心、支持和理解，我不可能做到。

其次，我想感谢 Manning 的团队，包括总编 Marjan Bace、副总编 Michael Stephens、开发编辑 Cynthia Kane、编审 Karen Tegtmeyer、文字编辑 Linda Recktenwald、校对 Katie Tennant 和制作经理 Mary Piergies。他们使得这本书成功面世，没有他们的努力，你现在就读不到这本书了。

我还想感谢 C++标准委员会的其他成员，他们为多线程工具上撰写了文件。正是 Andrei Alexandrescu、Pete Becker、Bob Blainer、Hans Boehm、Beman Dawes、Lawrence Crowl、Peter Dimov、Jeff Garland、Kevlin Henney、Howard Hinnant、Ben Hutchings、Jan Kristofferson、Doug Lea、Paul McKenney、Nick McLaren、Clark Nelson、Bill Pugh、Raul Silvera、Herb Sutter、Detlef Vollmann 和 Michael Wong，以及所有在文件上批注的人们，他们在委员会会议上进行了讨论，还有其他人的帮助才形成了 C++11 中的多线程和并发支持。

最后，我还想感谢下面的人：Dr. Jamie Allsop、Peter Dimov、Howard Hinnant、Rick Molloy、Jonathan Wakely 和 Dr. Russel Winder，他们的建议极大地改进了这本书。另外特别感谢 Russel 的详细审阅，还有技术校对 Jonathan，在编辑出版过程中极其用心地检查了终稿中所有内容中的错误（当然所有的错误都是由我造成的）。此外我想感谢我的审稿专家组：Ryan Stephens、Neil Horlock、John Taylor Jr.、Ezra Jivan、Joshua Heyer、Keith S. Kim、Michele Galli、Mike Tian-Jian Jiang、David Strong、Roger Orr、Wagner Rick、Mike Buksas 和 Bas Vodde。还要谢谢 MEAP 版的读者，他们花费时间来指出错误或是强调了需要阐明的部分。

前言

这本书是对新C++标准中的并发和多线程工具的深度指南，内容包括从 `std::thread`、`std::mutex` 和 `std::async` 的基本用法，到复杂的原子操作与内存模型。

路线图

前 4 章介绍了由类库提供的各种类库工具以及他们如何使用。

第 5 章涵盖了内存模型和原子操作的低阶基础，包括原子操作怎样在其他代码上强制实行排序约束，并标志着导言章节的结束。

第 6 章和第 7 章开始涵盖高阶的论题，包括一些如何使用基础工具来构造更复杂的数据结构的示例——第 6 章中基于锁的数据结构，以及第 7 章中无锁的数据结构。

第 8 章继续高阶论题，包括如何设计多线程代码的指南，涵盖了影响性能的论点，以及各种并行算法的示范实现。

第 9 章涵盖了线程管理——线程池、工作队列和中断操作。

第 10 章包括了测试和调试——bug 的类型，定位它们的技巧，如何测试它们，等等。

附件包含了对由新标准引入的与多线程相关的一些新语言工具的简要介绍，第 4 章中提到的消息传递库的具体实现，以及 C++11 线程库的完整参考。

谁应该阅读本书

如果你打算用 C++编写多线程代码，你就应该阅读本书。如果你正要使用 C++标准库中新的多线程工具，这本书是必备的指南。如果你正使用替代的线程库，后面几章中的指引和技巧应该也是有用的。

假设你对 C++已经有了很好的了解，但对新的语言特性却不甚熟悉，这些在附录 A 中也能找到答案。假定你之前没有多线程编程的知识和经验，那就更应该阅读本书。

如何使用本书

如果你以前从未写过多线程代码，我建议你按顺序从头到尾阅读本书，可以跳过第5章中的细节部分，但第7章大量依赖第5章中的材料，所以如果你跳过了第5章，你一定要阅览第7章，除非你曾读过。

如果你之前未曾使用过C++11语言工具，在你开始确定准备快速开始书中例子之前最好浏览一下附录A。新语言工具的使用凸显在文字之中，然而，当你遇到了之前没有见过的东西时，总是可以翻看附录的。

如果你在其他环境中拥有大量编写多线程代码的经验，开始的几章可能让你值得浏览一遍，以便你可以看看你了解的工具怎样映射到新C++标准中。如果你打算用原子变量做一些低阶的工作，第5章就是必需的。为了确认你熟悉多线程C++中类似异常安全的东西，值得阅览一下第8章。如果你在脑海中有特定的任务，索引和目录可以帮助你快速找到相关的章节。

一旦你打算促进C++线程库的使用，附录D应该仍然有用，比如查询每个类和函数调用的细节。你可能会想一次又一次地翻回主章节，来刷新你对某一概念的使用或者看一看示例代码。

代码约定和下载

所有代码清单和正文中的源代码，出现等宽字体（`like this`），是为了便于从常规文本中区分出来。代码注解伴随着很多清单，指出重要的概念。在有些情况下，数字符号往往指向清单后面的注释。

本书中所有的工作示例源代码可以在出版社网址下载，www.manning.com/CPlusPlusConcurrencyinAction。

软件需求

为了直截了当地使用本书中的代码，你需要一个最新的C++编译器，它支持示例中使用的新的C++11语言特性（参见附录A），并且你需要C++标准线程库的副本。

在撰写本书的时候，g++是我所知道的唯一带有标准线程库实现的编译器，尽管Microsoft Visual Studio 2011预览版也包括了实现。线程库的g++实现最早是在g++ 4.3中引入了基本形式，并且在后来的版本中进行了扩展。g++ 4.3也引入了一些新C++11语言特性的初次支持；对新语言特性更多的支持在各个后续版本中。详情参见g++ C++11状态页面[1]。

Microsoft Visual Studio 2010提供了一些新C++11语言特性，例如右值引用和lambda

[1] GNU Compiler Collection C++0x/C++11状态页面，http://gcc.gnu.org/projects/cxx0x.html。

函数，但并不带有线程库的实现。

作者的公司 Just Soft Solutions Ltd，出售为 Microsoft Visual Studio 2005、Microsoft Visual Studio 2008、Microsoft Visual Studio 2010 和 g++的各种版本[1]设计的 C++11 标准线程库的完整实现。这些实现已被用来测试本书中的示例。

Boost 线程库[2]提供了基于 C++标准线程库提案的 API，可以移植到许多平台。本书中的大部分示例可以通过审慎地将 `std::`替换成 `boost::`进行修改，并使用适当的 `#include` 指令，来与 Boost 线程库一起工作。在 Boost 线程库中有少数工具不受支持（比如 `std::async`）或者拥有不同的名称（比如 `boost::unique_future`）。

作者在线

购买 C++ Concurrency in Action 包括了免费访问由 Manning 出版社运营的私有网络论坛，在这里你可以对本书做出评论，提问技术问题，并且从作者和其他用户那里得到帮助。如果要访问此论坛并订阅它，可以访问网址 www.manning.com/CPlusPlusConcurrencyinAction。该页面提供了论坛的使用指南以及论坛上的行为规则。

Manning 对我们读者的承诺是提供一个场所，在这里每个读者之间以及读者和作者之间可以进行有意义的对话。就作者而言并没有承诺任何规定的参与量，作者对本书论坛的贡献只是义务的（且无偿的）。我们建议你试着向作者提问一些有挑战性的问题，以免他失去兴趣！

作者在线论坛以及过往讨论的归档，在本书在印期间都可以从出版社网站进行访问。

---

[1] C++标准线程库的 just::thread 实现，http://www.stdthread.co.uk。

[2] Boost C++库集合，http://www.boost.org。

资源

印刷资源

Cargill, Tom, "Exception Handling: A False Sense of Security,"in *C++ Report* 6, no. 9, (November-December 1994). Also available at http://www.informit.com/content/images/020163371x/supplements/Exception_Handling_Article.html.

Hoare, C.A.R., *Communicating Sequential Processes* (Prentice Hall International, 1985), ISBN 0131532898. Also available at http://www.usingcsp.com/cspbook.pdf.

Michael, Maged M., "Safe Memory Reclamation for Dynamic Lock-Free Objects Using Atomic Reads and Writes"in *PODC'02: Proceedings of the Twenty-first Annual Symposium on Principles of Distributed Computing* (2002), ISBN 1-58113-485-1.

———. U.S. Patent and Trademark Office application 20040107227, "Method for efficient implementation of dynamic lock-free data structures with safe memory reclamation."

Sutter, Herb, *Exceptional C++: 47 Engineering Puzzles, Programming Problems, and Solutions* (Addison Wesley Professional, 1999), ISBN 0-201-61562-2.

———. "The Free Lunch Is Over: A Fundamental Turn Toward Concurrency in Software,"in *Dr. Dobb's Journal* 30, no. 3 (March 2005). Also available at http://www.gotw.ca/publications/concurrency-ddj.htm.

在线资源

Atomic Ptr Plus Project Home, http://atomic-ptr-plus.sourceforge.net/.

Boost C++ library collection, http://www.boost.org.

C++0x/C++11 Support in GCC, http://gcc.gnu.org/projects/cxx0x.html.

C++11—The Recently Approved New ISO C++ Standard, http://www.research.att.com/~bs/C++0xFAQ.html.

Erlang Programming Language, http://www.erlang.org/.

GNU General Public License, http://www.gnu.org/licenses/gpl.html.

Haskell Programming Language, http://www.haskell.org/.

IBM Statement of Non-Assertion of Named Patents Against OSS, http://www.ibm.com/ibm/licensing/patents/pledgedpatents.pdf.

Intel Building Blocks for Open Source, http://threadingbuildingblocks.org/.

The just::thread Implementation of the C++ Standard Thread Library, http://www.stdthread.co.uk.

Message Passing Interface Forum, http://www.mpi-forum.org/.

Multithreading API for C++0X—A Layered Approach, C++ Standards Committee Paper N2094, http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2006/n2094.html.

OpenMP, http://www.openmp.org/.

SETI@Home, http://setiathome.ssl.berkeley.edu/.

翻译灾难
须读英文原著

简要目录

目录