PROGRAMMER TO PROGRAMMER™

Professional Multicore Programming
Design and Implementation for C++ Developers

C++多核高级编程

(美) Cameron Hughes 著
Tracey Hughes
齐 宁 译

wrox™
WILEY
清华大学出版社

C++多核高级编程

(美) Cameron Hughes 著
Tracey Hughes
齐 宁 译

清华大学出版社
北 京

Cameron Hughes，Tracey Hughes
Professional Multicore Programming: Design and Implementation for C++ Developers
EISBN：978-0-470-28962-4

北京市版权局著作权合同登记号　图字：　01-2009-2815

图书在版编目(CIP)数据

C++多核高级编程/(美)休斯(Hughes，C.)，(美)休斯(Hughes，T.)　著；齐宁　译.
—北京：清华大学出版社，2010.3
书名原文：Professional Multicore Programming: Design and Implementation for C++ Developers
ISBN 978-7-302-22274-3

I. C…　II. ①休…②休…③齐…　III. C 语言—程序设计　IV. TP312

中国版本图书馆 CIP 数据核字(2010)第 046015 号

责任编辑：王　军　于　平
装帧设计：孔祥丰
责任校对：胡雁翎
责任印制：何　芊
出版发行：清华大学出版社　　**地　　址**：北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**邮　　编**：100084
社　总　机：010-62770175　　**邮　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质　量　反　馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：北京鑫丰华彩印有限公司
装 订 者：三河市新茂装订有限公司
经　　销：全国新华书店
开　　本：185×260　**印　张**：36.25　**字　数**：882 千字
版　　次：2010 年 3 月第 1 版　　**印　　次**：2010 年 3 月第1 次印刷
印　　数：1～4000
定　　价：69.80 元

产品编号：030640-01

译 者 序

随着多核时代的到来，原本只属于高端应用的并行化编程也随之变得越来越普及。可以说，在处理器平台多核的大潮中，单纯的芯片制造工艺和技术已经不足以体现和发挥出多核所带来的更高处理能力以及性能优势，具备在多核环境中多线程工作的软件将会成为发挥多核高效率的保证。

多核技术在单个封装内集成了更多的核，为实现真正并行提供了物质基础。那么究竟如何设计和编写并行应用程序才能充分发挥多核架构的资源优势？这正是软件开发人员所要解决的问题。本书从这一需求出发，期望为面向多核架构设计和编写并行应用程序的开发人员提供一些力所能及的帮助。

在翻译本书的过程中，我们的切身体会就是本书并不仅仅是简单地介绍如何编写多核程序，更重要的是为程序开发人员提供了如何顺利从命令式顺序编程迁移到声明式并行编程的方法。作者以多个应用实例贯穿本书，一步步引导读者领会如何从问题陈述逐步精化，最终实现并行程序。

本书由齐宁和董泽惠翻译完成，Be Flying 工作室(http://blog.csdn.net/be_flying)负责人肖国尊负责翻译质量和进度的控制管理。书中文字和内容力求忠实原著，但限于译者水平和时间紧迫，翻译过程中难免出现不妥之处和错误，敬请广大读者批评指正。

关 于 作 者

Cameron Hughes 是一名专业的软件开发人员。他是 CTEST 实验室的软件工程师，同时还是 Youngstown 州立大学的编程人员/分析师。Cameron Hughes 有着超过 15 年的软件开发经验，参与过各种规模的软件开发工作，从商业和工业应用到航空设计和开发项目。Cameron 是 Cognopaedia 的设计者，目前是运行在 CTEST 实验室的 Pantheon 上的 GRIOT 项目的领导者。Pantheon 是具有 24 个节点的多核集群，用于多线程搜索引擎和文本提取程序的开发。

Tracey Hughes 是 CTEST 实验室的高级图像程序员，负责开发知识和信息的可视化软件。Tracey Hughes 是利用 CTEST 实验室的知识可视化的 M.I.N.D、C.R.A.I.G、NOFAQS 等项目的主要设计人员。她经常致力于 Linux 开发软件。她还是 GRIOT 项目的小组成员。

Cameron 和 Tracey Hughes 还是关于软件开发、多线程和并行编程方面的 6 本著作的作者，这 6 本著作是：*Parallel and Distributed Programming Using C++*、*Linux Rapid Application Development*、*Mastering the Standard C++ Classes*、*Object - Oriented Multithreading Using C++*、*Collection and Container Classes in C++*和 *Object-Oriented I/O Using C++ Iostreams*。

前　言

多核革命即将到来。并行处理不再是超级计算机或集群的专属领域，入门级服务器乃至基本的开发工作站都拥有硬件级和软件级并行处理的能力。问题是这对于软件开发人员意味着什么？对软件开发过程会有怎样的影响？在谁拥有速度最快的计算机的竞争中，芯片生产商更倾向于在单独的芯片上放置多个处理器，而不是提高处理器的速度。迄今为止，软件开发人员尚能依赖于新的处理器，在不对软件做出任何实际改进的情况下提高软件的速度，但是这样的情况将成为过去。为了提高总体系统性能，计算机供应商已经决定增加更多的处理器，而不是提高时钟频率。这意味着如果软件开发人员希望应用程序从下一个新的处理器受益，就必须对应用程序进行修改以利用多处理器计算机。

尽管顺序编程和单核应用程序开发仍会有一席之地，但软件开发将向多线程和多进程转变。曾经仅被理论计算机科学家和大学学术界所关注的并行编程技术，现在正处于为大多数人所接受的过程中。多核应用程序设计和开发的思想如今成为人们关注的主流。

学习多核编程

本书使用一般软件开发人员能够理解的术语介绍多核编程的基本知识。为读者介绍了为多处理器和多线程体系结构进行编程的基础知识，对并行处理和软件并发的概念进行了实用的介绍。本书介绍的是深奥的、不易理解的并行编程技术，但将使用一种简单、可理解的方式来介绍它们。我们介绍了并发编程和同步的缺陷与陷阱以及应对之策，对多处理和多线程模型进行了直截了当的讨论。本书提供了大量的编程实例，示范了如何实现成功的多核编程。本书还包含了调试及测试多核编程的方法与技术。最后，我们示范了如何使用跨平台技术来利用处理器的具体特性。

不同的视角

本书的内容是为对多核编程和应用程序开发有着不同切入点的广大读者设计的。本书的读者包含但不限于：

- 库及工具制作人员
- 操作系统程序员
- 内核开发人员
- 数据库服务器及应用服务器的设计人员及实现人员
- 科学应用程序员以及使用计算密集型应用程序的用户
- 应用程序开发人员

- 系统程序员

每组人员都会从不同的视角来了解多核计算机。有些人关心的是使用自底向上的方法，需要开发利用特定硬件和特定供应商的特性的软件。对于上述人员而言，他们希望更加详尽地介绍多线程处理的知识。其他人员可能对自顶向下的方法感兴趣，他们不希望为并发任务同步或线程安全的细节费心，而是倾向于使用高级库和工具来完成工作。还有一些人需要混合使用自底向上和自顶向下的方法。本书提供了对多核编程的多种视角的介绍，涵盖了自底向上和自顶向下的方法。

解决方案是多范型方法

首先，我们承认不是每个软件解决方案都需要多处理或多线程。有些软件解决方案通过使用顺序编程技术能够更好地实现(即使目标平台是多核的)。我们的方法是以解决方案和模型作为驱动。首先，针对问题开发出模型或解决方案。如果解决方案要求某些指令、过程或任务并发地执行，那么就决定了最好使用哪组技术。这个方法同强迫解决方案或模型去适合一些预先选择的库或开发工具的方法相反。技术应当遵从解决方案。尽管本书讨论库和开发工具，但并不偏向任何具体生产商库或工具集。尽管书中包含了利用特定硬件平台的实例，但我们依赖跨平台方法，使用 POSIX 标准操作系统调用和库，并且仅使用国际化 C++标准所支持的 C++特性。

面对多处理和多线程中的挑战和障碍，我们倡导组件方法。主要的目的是利用框架类作为并发的构建块。框架类被面向对象互斥量(mutex)、信号量(semaphore)、管道(pipe)、队列(queue)和套接字(socket)所支持。通过使用接口类，显著降低了任务同步和通信的复杂度。在我们的多线程和多处理应用程序中，控制机制主要是 agent 驱动。这意味着在本书中，您将看到应用程序架构支持软件开发的多范型(multiple-paradigm)方法。

我们对组件实现使用面向对象编程技术，对控制机制主要使用面向 agent 编程技术。面向 agent 编程的思想有时被逻辑编程技术支持。随着处理器上可用内核数目的增加，软件开发模型将会越发地依赖面向 agent 编程和逻辑编程。本书包含了对软件开发的这种多范型方法的简介。

为何使用 C++

事实上，每个平台和操作环境中均有可用的 C++编译器。ANSI(American National Standards Institute，美国国家标准协会)和 ISO(International Organization for Standardization，国际标准化组织)已经为 C++语言和它的库定义了标准。C++语言具有可靠的开源实现以及商业实现，并且已经被全世界的研究人员、设计人员和专业开发人员所广泛接受。C++语言被用于解决各种各样的问题，从设备驱动到大规模的工业应用。C++语言支持软件开发的多范型方法。在 C++中，我们可以无缝地实现面向对象设计、逻辑编程设计和面向 agent

设计。我们还可以在必要时使用结构化编程技术或低级编程技术。这种灵活性正是新的多核技术所需要加以利用的。此外，C++编译器为软件开发人员提供了多核处理器的新特性的直接接口。

UML 图

本书中的很多图使用了 UML(Unified Modeling Language，统一建模语言)标准。特别是使用活动图、部署图、类图和状态图来描述重要的并发架构和类关系。尽管 UML 的知识不是必需的，但熟悉它会对工作和学习有所帮助。

支持的开发环境

本书中的实例均使用 ISO 标准 C/C++开发。这意味着实例和程序能够在所有主要环境中进行编译。完整程序中仅使用 POSIX 兼容的操作系统调用或库，因此，这些程序能够移植到所有兼容 POSIX 的操作系统环境中。本书中的实例和程序在配有 UltraSparc T1 multiprocessor、Intel Core 2 Duo、IBM Cell Broadband Engine 和 AMD Dual Core Opteron 处理器的 SunFire 2000 上都进行了测试。

程序概要

本书中的多数完整程序均伴有一个程序概要(program profile)。概要包含实现细节，如必需的头文件、必需的库、编译指令和链接指令。概要还包括一个注释部分，包含执行程序时需要注意的任何特殊考虑。所有的代码仅用于说明目的。

测试及代码可靠性

尽管本书中的所有实例和应用程序均为了确保正确性而经过了测试，但我们并不保证书中包含的程序没有缺陷或错误，或者同任何特定标准或适销性相一致，或满足您的任何特定应用的要求。您不应依赖于它们来解决这样的问题，即问题的不正确的解决方案可能会导致人员伤害或财产损失。作者和出版商不对使用本书中的实例、程序或应用程序所导致的直接或间接损害承担任何责任。

约定

为了帮助您更好地了解本书的内容，我们在书中使用了如下的约定。

对于正文中的样式：

- 我们对新的术语和重要的词在引入它们时进行了强调。
- 我们采用类似如下的方式显示组合键：Ctrl+A。
- 我们以两种不同的方式显示代码：

```
We use a monofont type with no highlighting for most code examples.

We use gray highlighting to emphasize code that's particularly important
in the present context.
```

本书中既包含程序清单(code listing)，又包含代码示例(code example)。

- 程序清单是完整的、可执行的程序。如前所述，多数情况下，它们会伴有一个程序概要，它会告诉您程序编写时的环境，并给出编译指令和链接指令的描述，等等。
- 代码示例是一些片断，不加修改是不能够运行的。它们用来集中展示某些内容如何被调用或使用。

源代码

在您完成本书中的例子时，您可以选择手工输入所有代码或使用伴随本书的源代码文件。本书中所使用的所有源代码可以从 http://www.tupwk.com.cn/downpage 下载，也可以从 http://www.wrox.com 下载。访问到该网站之后，只要找到本书的书名(使用 Search 输入框或使用一个书名列表)，然后在本书的详情页面上单击 Download Code 链接来得到本书的所有源代码。

注意：

由于很多书的书名很类似，最简单的查找方式是通过 ISBN，本书的 ISBN 为 978-0-470-28962-4。

一旦下载了代码，只需要使用您最喜欢的压缩工具对它进行解压。或者，您可以转到 Wrox 的代码下载主页面，网址为 http://www.wrox.com/dynamic/books/download.aspx，查看本书及所有 Wrox 书籍的可用代码。

勘误表

我们尽全力确保正文或代码中没有错误。然而人无完人，错误总会发生。如果您在我们的书中发现了错误，例如拼写错误或错误的代码段，我们将会非常感激您的反馈。通过递交勘误，您可能会帮助另一名读者避免数小时的受挫，同时，也能帮我们提供质量更高的书籍。

为了找到本书的勘误页面，请访问 http://www.wrox.com 并通过 Search 输入框或根据书名列表找到本书的书名。然后，在本书的详情页面上，单击 Book Errata 链接。在这个页面上您可以看到所有已经为本书提交的并由 Wrox 编辑发布的勘误。

如果在 Book Errata 页上没有找到您所发现的错误，请将错误发送至 wkservice@vip.163.com。我们将会查看信息，如果情况属实，则会发布消息到本书的勘误页，并在本书的后续版本中做出修订。

p2p.wrox.com

如果您希望同作者和本书其他读者进行讨论，可以加入到 http://p2p.wrox.com 上的 P2P 论坛。该论坛是一个基于 Web 的系统，您可以在论坛中发布同 Wrox 书籍及相关技术有关的消息，并且可以同其他读者和技术用户交流。论坛提供了订阅特性，可以将论坛上发布的您所感兴趣的话题通过 E-mail 发送给您。论坛中有 Wrox 作者、编辑、其他行业专家以及读者。

在 http://p2p.wrox.com 上，您不仅可以找到许多帮助您阅读本书的不同的论坛，而且还可以开发自己的应用程序。要想加入论坛，应执行下列步骤：

(1) 进入 http://p2p.wrox.com 并单击 Register 链接。

(2) 阅读使用条款并单击 Agree 按钮。

(3) 填写加入论坛所要求的信息以及您希望提供的其他可选信息，然后单击 Submit 按钮。

(4) 您将会收到一封电子邮件，其中的内容描述了如何验证您的账号并完成加入过程。

注意：

即使不加入 P2P，您也可以阅读论坛中的消息，但是如果您希望发布自己的消息，则必须加入 P2P。

一旦加入 P2P 之后，您可以发布新的消息并回复其他用户发布的消息。您可以在任何时候阅读 Web 上的消息。如果您希望特定论坛的新的消息以电子邮件的形式发送给您，可单击论坛列表中论坛名字旁边的 Subscribe to this Forum 按钮。

要想知道更多关于如何使用 Wrox P2P 的信息，可以阅读 P2P FAQ 中关于论坛软件如何工作以及很多关于 P2P 和 Wrox 书籍的其他常见问题的答案。要想阅读 FAQ，可单击 P2P 页面中的 FAQ 链接。

目　录

第 1 章

新的体系结构

在台式计算机的微处理器设计方面的最新进展包括将多个处理器放置到一个计算机芯片上。这些多核设计完全取代了作为台式计算机的基础的单核设计。IBM、Sun、Intel和 AMD 都已经将他们的芯片流水线从生产单核处理器变为生产多核处理器。这已经促使计算机供应商将他们的重心转移到销售拥有多核的台式计算机，如 Dell、HP 和 Apple。在这个新的领域的市场份额的竞争中，每个计算机芯片生产商都在挑战着能够经济地放置到一个芯片上的内核数目的极限。所有这些竞争使得消费者拥有了比以前更多的计算能力。主要的问题在于常规的台式计算机软件没有被设计为利用新的多核架构。实际上，为了能够从新的多核体系结构中得到任何真正的加速，将必须重新设计台式计算机软件。

设计和实现利用多核处理器的应用软件的技术，和在单核开发中使用的技术有着根本的区别。软件设计和开发的焦点将必须从顺序编程技术转变为并行和多线程编程技术。

现在，一般开发人员的工作站以及入门级的服务器都采用具有硬件级多线程、多处理和并行处理能力的多处理器。尽管顺序编程和单核应用开发仍将保留有一席之地，但是多核应用程序设计和开发的思想现在已经成为主流。

本章将会带您了解多核编程，介绍的内容有：

- 什么是多核？
- 有哪些多核体系结构以及它们之间有什么区别？
- 作为软件的设计人员和开发人员，当从顺序编程和单核应用开发转移到多核编程时，需要知道哪些知识？

1.1 什么是多核

多核是一种将多个处理器放置到一个计算机芯片上的架构设计。每个处理器被称作一个核。随着芯片容量的增加，在一个芯片上放置多个处理器变得可行。这些设计被称为芯片多处理器(Chip Multiprocessor，CMP)，是因为它们允许单芯片进行多处理。多核是 CMP 或单芯片多处理器的流行的名字。单芯片多处理的概念并不是新的，早在 20 世纪 90 年代

初期，芯片生产商就已经开始探索在单芯片上放置多个内核的想法。最近，CMP 已经成为改进总体系统性能的首选方法。这种方法与通过增加时钟频率或处理器速度来获得总体系统性能收益是截然不同的。增加时钟频率的方法已经开始在成本效益方面达到其极限了。较高的频率要求更多的能耗，使得系统制冷变得困难且代价高昂。这还会影响确定尺寸和封装(sizing and packaging)方面的考虑。所以，现在的做法是增加更多的处理器，而不是令处理器运行得更快以获得性能上的提升。由于这种方法实现简单，因此是更好的方法，所以该方法驱动了多核革命。当前，多核体系结构在增强总体系统性能方面成为了焦点。

对于熟悉多处理的软件开发人员，对多核开发应当也不陌生。从逻辑的观点，对分开封装的多个处理器进行编程与对包含在单独芯片上的单个封装上的多个处理器进行编程，两者之间并没有很大的区别。当然可能会存在性能差异，因为新的 CMP 利用了总线体系结构以及处理器之间连接等方面的进步。在某些环境下，这可能会使得原本为多个处理器编写的程序能够在 CMP 上更快地执行。除了潜在的性能收益，设计和实现都是非常类似的。在本书中，我们将讨论其中存在的微小差别。对于只熟悉顺序编程和单核开发的开发人员，多核方法提供了很多新的软件开发范型。

1.2 多核体系结构

CMP 有多种形式：两个处理器(双核)、四个处理器(四核)和八个处理器(八核)结构。有些结构是多线程，有些结构不是。在新的 CMP 中，高速缓冲存储器(cache)和内存的处理方式有着几种变体，在不同的实现中，处理器与处理器之间的通信方法也不同。来自各大主要芯片生产商的 CMP 实现中，在处理 I/O 总线和前端总线(Front Side Bus，FSB)上均不相同。

如果严格地从逻辑视图上查看被设计为利用多核体系结构的应用程序时，并看不出大多数区别。图 1-1 示范了支持多处理的 3 种常见配置。

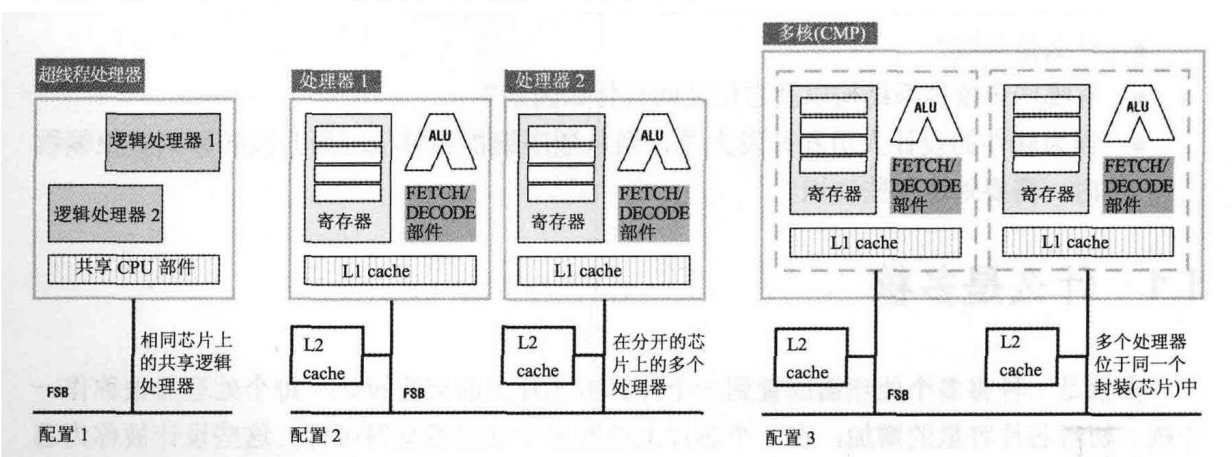

图 1-1

- 图 1-1 中的配置 1 使用超线程(hyperthreading)。与 CMP 类似，使用超线程的处理器允许在一个芯片上执行两个或多个线程。然而，在超线程的封装中，多个处理器指的是逻辑上的，而不是物理上的。这些封装中有着两套硬件，但是不足以构成一个单独的物理处理器。这样，超线程允许处理器在将自身呈现给操作系统时，好像是完备的多处理器，但实际上只是在一个处理器上运行多个线程。
- 图 1-1 中的配置 2 是经典的多处理器。在配置 2 中，每个处理器位于一个独立的芯片上，而且有着自己的硬件。
- 配置 3 代表了当前多处理器的发展趋势，它在一个芯片上提供完整的多个处理器。

如同您将在第 2 章所看到的，一些多核设计在核的内部支持超线程。例如，一个使用了超线程技术的双核处理器可以将自己作为四核处理器呈现给操作系统。

混合型多核体系结构

混合型多核体系结构(hybrid multicore architecture)在一个封装中混合了多种处理器类型和/或线程模式。这样能够通过将多种独特的性能合并到一个功能内核中，提供有效的代码优化和特化(specialization)的方法。混合型多核架构的最常见示例是 IBM 的 Cell broadband engine(Cell)。我们将在下一章中介绍 Cell 的体系结构。

需要注意的是每种配置是作为由两个或更多个能够并发执行多个任务的一组逻辑处理器来呈现给开发人员的。对系统程序员、内核程序员和应用程序开发人员的挑战是需要了解何时以及如何利用这一点。

1.3　软件开发人员眼中的多核体系结构

CMP 的低成本和广泛可用性，使得一般的软件开发人员能够进行各种级别的并行处理。并行处理不再是超级计算机或集群的专属领域。基本的开发工作站和入门级服务器现在都具有软件级和硬件级的并行处理能力。这意味着程序员和软件开发人员可以无需牺牲设计或性能，即可根据需要部署利用多处理和多线程的应用。然而，需要注意的是，并非每个软件应用都需要多处理或多线程。实际上，一些软件解决方案和计算机算法最好使用顺序编程技术来实现。在某些情况下，在软件中引入并行编程技术的开销会使软件性能降级。并行性和多处理是需要一定成本的。如果软件中顺序地解决问题需要的工作量少于创建额外线程和进程的开销，或者少于协调并发执行的任务之间通信的工作，则应选择顺序的方法。

有时可以较容易地确定何时及何地应当使用并行性，因为软件解决方案本身可能会要求支持并行性。例如在很多客户端—服务器配置中，很显然是需要并行性的。可能有一个服务器，例如数据库，还有很多可以同时对数据库发起请求的客户端。在多数情况下，您不希望一个客户端被要求等待，直到另外一个客户端的请求被满足。可接受的解决方案允许软件并发地处理客户端的请求。另一方面，有时候可能会在不需要并行性时面对并行性