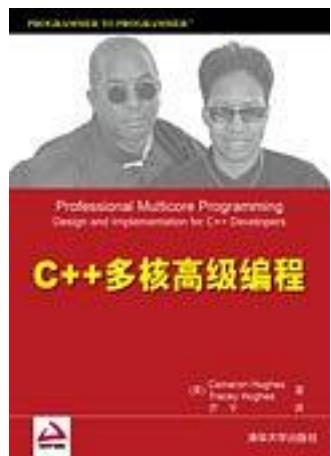


C++多核高级编程



[C++多核高级编程 下载链接1](#)

著者:Cameron Hughes

出版者:清华大学出版社

出版时间:2010年4月第1版

装帧:

isbn:9787302222743

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。

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，对多处理器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 is 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W

rox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。

1.1 什么是多核 1

1.2 多核体系结构 2

1.3 软件开发人员眼中的

多核体系结构 3

1.3.1 基本的处理器体系结构 4

1.3.2 CPU(指令集) 6

1.3.3 内存是关键 8

1.3.4 寄存器 10

1.3.5 cache 11

1.3.6 主存 12

1.4 总线连接 13

1.5 从单核到多核 13

1.5.1 多道程序设计和多处理 14

1.5.2 并行编程 14

1.5.3 多核应用程序的设计与实现 15

1.6 小结 15

第2章 4种有影响的多核设计 17

2.1 AMD Multicore Opteron 19

2.1.1 Opteron的直连

和HyperTransport 19

2.1.2 系统请求接口和交叉开关 20

2.1.3 Opteron使用NUMA结构 21

2.1.4 cache以及多处理器Opteron 22

2.2 Sun UltraSparc T1 多处理器 22

2.2.1 UltraSparc T1内核 24

2.2.2 Cross Talk与Crossbar 25

2.2.3 DDRAM控制器和L2 cache 25

2.2.4 UltraSparc T1、Sun和

GNU gcc编译器 25

2.3 IBM Cell Broadband Engine 25

2.3.1 CBE与Linux 26

2.3.2 CBE内存模型 27

2.3.3 对操作系统隐藏 27

2.3.4 协处理器部件 28

2.4 Intel Core 2 Duo处理器 28

2.4.1 北桥和南桥 29

2.4.2 Intel的PCI Express 29

2.4.3 Core 2 Duo的指令集 29

2.5 小结 30

第3章 多核编程的挑战 33

3.1 什么是顺序模型 33

3.2 什么是并发 34

3.3 软件开发 35

3.3.1 挑战1：软件分解 38

3.3.2 挑战2：任务间通信 43

3.3.3 挑战3：多个任务或agent

对数据或资源的并发访问 47

3.3.4 挑战4：识别并发执行的

任务之间的关系 51

3.3.5 挑战5：控制任务之间的

资源争夺 53

3.3.6 挑战6：需要多少个进程

或线程 53

3.3.7 挑战7和挑战8：寻找可靠

的、可重现的调试和测试 54

3.3.8 挑战9：与拥有多进程组件的设计的相关人员进行沟通 55

3.3.9 挑战10：在C++中实现多处理和多线程 56

3.4 C++开发人员必须学习新的库 56

3.5 处理器架构的挑战 57

3.6 小结 57

第4章 操作系统的任务 59

4.1 操作系统扮演什么角色 59

4.1.1 提供一致的接口 59

4.1.2 管理硬件资源和其他

应用软件 60

4.1.3 开发人员与操作系统

的交互 60

4.1.4 操作系统的核心服务 63

4.1.5 应用程序员的接口 66

程序概要4-1 70

程序概要4-2 74

4.2 分解以及操作系统的任务 75

4.3 隐藏操作系统的任务 77

4.3.1 利用C++抽象和封装的能力 77

4.3.2 POSIX API的接口类 78

4.4 小结 85

第5章 进程、C++接口类和谓词 87

5.1 多核是指多处理器 87

5.2 什么是进程 88

5.3 为什么是进程而不是线程 88

5.4 使用posix_spawn() 90

5.4.1 file_actions参数 91

5.4.2 attrp参数 92

5.4.3 简单的posix_spawn()示例 94

5.4.4 使用posix_spawn的guess_it 95

5.5 哪个是父进程，哪个是
子进程 99

5.6 对进程的详细讨论 99

5.6.1 进程控制块 100

5.6.2 进程的剖析 101

5.6.3 进程状态 103

5.6.4 进程是如何被调度的 105

5.7 使用ps实用工具监视进程 107

5.8 设置和获得进程优先级 110

5.9 什么是上下文切换 112

5.10 进程创建中的活动 112

5.10.1 使用fork()函数调用 113

5.10.2 使用exec()系统

调用系列 113

5.11 进程环境变量的使用 116

5.12 使用system()生成

新的进程 117

5.13 删除进程 118

5.13.1 调用exit()和abort() 118

5.13.2 kill()函数 119

5.14 进程资源 119

5.14.1 资源的类型 120

5.14.2 设置资源限制的POSIX

函数 121

5.15 异步进程和同步进程 124

5.16 wait()函数调用 125

5.17 谓词、进程和接口类 127

5.18 小结 131

第6章 多线程 133

6.1 什么是线程 133

6.1.1 用户级线程和内核级线程 134

6.1.2 线程上下文 136

6.1.3 硬件线程和软件线程 138

6.1.4 线程资源 138

6.2 线程和进程的比较 139

6.2.1 上下文切换 139

6.2.2 吞吐量 139

6.2.3 实体间的通信 139

6.2.4 破坏进程数据 140

6.2.5 删除整个进程 140

6.2.6 被其他程序重用 140

6.2.7 线程与进程的关键类似

和差别 140

6.3 设置线程属性 142

6.4 线程的结构 143

6.4.1 线程状态 144

6.4.2 调度和线程竞争范围 145

6.4.3 调度策略和优先级 147

6.4.4 调度分配域 148

6.5 简单的线程程序 148

6.6 创建线程 150

6.6.1 向线程传递参数 151

6.6.2 结合线程 153

6.6.3 获得线程id 154

6.6.4 使用pthread属性对象 155

6.7 管理线程 159

6.7.1 终止线程 159

6.7.2 管理线程的栈 168

6.7.3 设置线程调度和优先级 171

6.7.4 设置线程的竞争范围 175

6.7.5 使用sysconf() 175

6.7.6 线程安全和库 177

6.8 扩展线程接口类 179

6.9 小结 187

第7章 并发任务的通信和同步 189

7.1 通信和同步 189

7.1.1 依赖关系 190

7.1.2 对任务依赖进行计数 193

7.1.3 什么是进程间通信 195

7.1.4 什么是线程间通信 215

7.2 对并发进行同步 223

7.2.1 同步的类型 224

7.2.2 同步对数据的访问 224

7.2.3 同步机制 230

7.3 线程策略方法 250

7.3.1 委托模型 251

7.3.2 对等模型 253

7.3.3 生产者-消费者模型 254

7.3.4 流水线模型 255

7.3.5 用于线程的SPMD和

MPMD 256

7.4 工作的分解和封装 258

7.4.1 问题陈述 258

7.4.2 策略 258

7.4.3 观察 259

7.4.4 问题和解决方案 259

7.4.5 流水线的简单agent

模型实例 260

7.5 小结 264

第8章 PADL和PBS：应用程序

设计方法 265

8.1 为大规模多核处理器设计

应用程序 265

8.2 什么是PADL 268

8.2.1 第5层：应用程序

架构选择 271

8.2.2 第4层：PADL中的

并发模型 281

8.2.3 第3层：PADL的

实现模型 284

8.3 谓词分解结构 306

8.3.1 示例：Guess-My-Code

游戏的PBS 307

8.3.2 将PBS、PADL和SDLC

联系起来 307

8.3.3 对PBS进行编码 308

8.4 小结 308

第9章 对要求并发的软件系统

进行建模 311

9.1 统一建模语言 311

9.2 对系统的结构进行建模 313

9.2.1 类模型 313

9.2.2 类的可视化 315

9.2.3 对属性和服务进行排序 320

9.2.4 类的实例的可视化 322

9.2.5 模板类的可视化 324

9.2.6 显示类与对象的关系 325

9.2.7 接口类的可视化 329

9.2.8 交互式对象的组织 331

9.3 UML与并发行为 332

9.3.1 协作对象 332

9.3.2 使用进程与线程的多任务

与多线程 334

9.3.3 对象间的消息序列 335

9.3.4 对象的活动 337

9.3.5 状态机 339

9.4 整个系统的可视化 344

9.5 小结 345

第10章 并行程序的测试和

逻辑容错 347

10.1 能否跳过测试 347

10.2 测试中必须检查的5个

并发挑战 348

10.3 失效：缺陷与故障导致

的结果 350

10.3.1 基本的测试类型 350

10.3.2 缺陷排除与缺陷存活 351

10.4 如何对并行程序实现

缺陷排除 351

10.4.1 问题陈述 352

10.4.2 简单策略和粗解决

方案模型 352

10.4.3 使用PADL第5层的

修正的解决方案模型 352

10.4.4 agent解决方案模型

的PBS 353

10.5 什么是标准软件工程测试 357

10.5.1 软件验证与确认 357

10.5.2 代码不能正常工作

该怎么办 358

10.5.3 什么是逻辑容错 362

10.5.4 谓词异常和可能世界 367

10.5.5 什么是模型检测 368

10.6 小结 368

附录A 并发设计使用的UML 371

附录B 并发模型 379

附录C 线程管理的POSIX标准 393

附录D 进程管理的POSIX标准 535

作者介绍:

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。

目录:

[C++多核高级编程_下载链接1](#)

标签

C++

多核编程

并发编程

多线程

计算机

C/C++

操作系统

并行

评论

组织散乱，不够深入

没多少有含量的内容。。一半的内容基本是对man的翻译

垃圾书，不用浪费时间了

没怎么看

没怎么看，条例不清晰，例子不完整

[C++多核高级编程 下载链接1](#)

书评

[C++多核高级编程_下载链接1](#)