
C
C&C++ Practice
C和C++实务精选
WILEY
C++面向对象多线程编程
Object-Oriented
Multithreading Using C++
Cameron Hughes
Tracey Hughes
著
周良忠
译
人民邮电出版社
POSTS & TELECOMMUNICATIONS PRESS

C和C++实务精选

C++面向对象多线程编程

Cameron Hughes
Tracey Hughes 著
周良忠 译

人民邮电出版社

图书在版编目（CIP）数据

C++面向对象多线程编程/（美）休斯（Hughes,C），（美）休斯（Hughes,T.）著；周良忠译. —北京：人民邮电出版社，2003.4

ISBN 7-115-10881-1

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

中国版本图书馆 CIP 数据核字（2003）第 018321 号

版 权 声 明

C 和 C++实务精选

C++ 面向对象多线程编程

◆ 著 Cameron Hughes Tracey Hughes

译 周良忠

责任编辑 陈冀康

◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街 14 号

邮编 100061 电子函件 315@ptpress.com.cn

网址 http://www. ptpress.com.cn

读者热线 010-67132705

北京汉魂图文设计有限公司制作

北京顺义振华印刷厂印刷

新华书店总店北京发行所经销

◆ 开本：800×1000 1/16

印张：33.5

字数：750 千字 2003 年 4 月第 1 版

印数：1-4 000 册 2003 年 4 月北京第 1 次印刷

著作权合同登记 图字：01－2002－4865 号

ISBN 7-115-10881-1/TP・3200

定价：68.00 元

本书如有印装质量问题，请与本社联系 电话：(010)67129223

内容提要

全书共分 13 章，全面讲解构建多线程架构与增量多线程编程技术。第 1 章介绍了用于构建面向对象程序的不同类型 C++组件，以及如何使用这些组件来构建多线程架构。第 2、3、4 章简要介绍进程、线程、多任务处理、多线程化、规划以及线程优先权的概念。第 5 章讨论进程间和线程间通信。第 6 章讨论线程与进程同步与合作。第 6 章详细讨论临界区、死锁、数据竞争以及无限延迟方面的主题。第 7~10 章讲解用于线程同步、线程间通信、进程间通信以及多线程处理的 C++组件。第 11 章讨论 C++对象在多线程环境中的行为和交互方式。第 12 章简单介绍多线程应用程序的测试技术。第 13 章对全书内容进行扼要地回顾与思考。

本书适合用 C++创建多线程组件和应用框架的程序员阅读。

译者简介

周良忠，男，1970年生。本科毕业于武汉化工学院计算机应用专业。1995年毕业于中国科学院武汉岩土力学研究所，获硕士学位，1997年获得博士学位。1998年创办云巅工作室(http://www.cloudcrown.com)，为个人、中小企业提供专业软件定做服务。近几年开发了多款广受欢迎的共享软件。精通C++、C#、Java、Perl等开发语言。2001年开始从事计算机科技图书的创作和翻译工作，最新翻译力作有《C# Primer Plus中文版》、《C++实践之路》等。

译者的话

据我所知，很多程序员惧怕多线程编程，主要因为多线程带来的死锁、数据竞争、优先权倒置、无限延迟等问题让他们望而却步。这些互斥性问题轻则让应用程序的速度不快反慢，重则使程序崩溃。而更多的开发人员义无反顾地拿起了“多线程”这把双刃剑。这也许是技术发展的大势所趋：

- 处理并行计算的需要；
- 随着计算机应用范围的拓展，需要超长运算时间的应用程序越来越多；
- 避免等待网络、文件系统、用户或其他 I/O 操作而耗费大量的执行时间。

如果能恰当运用多线程编程技术，以上领域的应用程序就会显著提高反应速度、处理的吞吐量以及处理器的有效利用率。当今的流行操作系统基本上都支持多线程。

但这方面能滋补读者的有价值的图书资料实在太少了。Cameron Hughes 与 Tracey Hughes 编撰的 *Object-Oriented Multithreading Using C++*一书在国外读者中享有极高的声誉。能够翻译这样一本优秀的参考书实在是我的荣幸。在翻译过程中，我也为其中精彩的内容所吸引，甚至常常忘记了翻译的“本职工作”，径自捧读而爱不释手。

综观全书，它步步深入地剖析了面向对象多线程架构与增量多线程编程的概念与技术。通读全书，每一种多线程编程中可能遇到的互斥性问题都能找到令人满意的解决方案。因此，这本书将成为那些“惧怕”多线程编程程序员逾越技术障碍的有力武器。

本书与其他类似参考书相比，有两个突出不同点：一是本书所讨论的 C++组件和示例适用于 POSIX pthreads、Win32 API 或 OS/2 任何一种支持多线程的环境；二是丰富而又规范的图示能帮助读者更准确地理解所讲解的内容。另外，附录部分包括 POSIX 线程管理规范、同步类、线程类以及进程间通信类的源代码示例。www.cloudcrown.com 提供源代码下载。这些都是读者实践中的有用资源。

由于译者水平有限，错误在所难免，望广大读者不吝指正。译者 E-mail：web_zhou@21cn.com。

愿与广大读者共同学习、共同进步！

译者

2002 年 10 月

前言

这是一本讲解用于生成可以同时执行多个任务的程序与应用的面向对象C++组件构建的参考书。本书着重讲解支持多线程和并发处理的软件生成机制。本书推荐使用那些能自然适应于并行性的软件架构。本书所讲授的知识与软件“分治（divide and conquer）”技术有关。

学习本书的必要性

当今的应用程序既需要大量的CPU时间，又依赖于高流量I/O。用户不再满足于一次只执行一个任务的应用了。像Web浏览器这样支持上载与下载的应用程序必须允许用户在上载或下载的同时还能够撰写E-mail消息，或者阅览文档。电子数据表必须允许用户在浏览其中一部分占用大量处理器时间的图形的同时，还能执行另一部分冗长的计算。当今的许多应用程序都包含与声音同步的视频序列。视频序列的解码程序必须能够压缩或解压视频，同时要让视频图像与保存在视频图像中的声音数据同步。用户不能容忍无声的视频或只有声音没有图像的视频效果。假如字处理程序正在打印一份长文档的时候，用户不能编辑另外的文件，这是不可接受的。用户不会使用不能同时搜索和排序的数据库应用程序。如果某应用程序包含通信功能、打印功能、数据库功能、数值计算功能等等，用户希望能够并发执行其中的某些功能。一次只做一件事的日子已一去不复返了。我们需要那些可以让我们构建满足用户需求的应用程序的技术。我们需要那些可以让我们解决搜寻互联网上数据时所出现问题的技术。我们需要的是能应付处理器，而且能满足高流量I/O多媒体需求的技术和架构。

发展趋势

与之相应，应用程序的发展趋势就是让应用程序多线程化。多线程化应用程序是那些将任务分成多个线程的应用程序。这些线程就像工人一样。在单线程应用程序中，只有一个工人完成应

用程序将要满足的请求。如果用户发出多个请求，用户必须等待应用程序的唯一工人逐个地完成所有请求。不过，在多线程应用程序中存在多个工人。不仅有多个工人，而且这些工人还可以同时执行他们的功能。可分配工人组来共同完成一项任务，同时分配其他工人独立完成其他任务。在多线程应用程序中，将应用程序执行的任务分配给并发执行的工人集合，它们称做线程。如果用户发出了多个请求，这些请求可以同时得到满足，因为可以分派每个工人来满足用户请求的某一部分。因为应用程序具有多个工人，所以工人可以并发满足多个请求。当应用程序包含多个线程时，该应用程序就是多线程处理的。

存在的难题

尽管多线程处理应用程序可以用来提高用户的生产力，也能加速程序的运行，但也存在许多缺陷。当应用程序拥有多个工人时，由谁来协调这些工人呢？如何将应用程序的功能性在这些工人间进行分派？哪一个工人在何时用多长时间来完成哪一项任务？谁首先启动？谁最后结束？如果有个工人的任务执行失败了，将会发生什么情况呢？如果工人混淆了信号与优先权，情况将如何？因为多线程程序可能包含上百，有时是上千个线程，所以我们可能应验这句谚语：人多误事(Too many cooks spoil the stew)。也就是说，太多的工人可能导致应用程序执行迟缓，甚至被锁住。有时，工人间为谁首先使用特殊的资源而相互竞争。两个工人可能同时想发送不同的声音、图像或视频文件到同一个多媒体设备。一个工人可能想打开开关，而另一个工人却同时想关掉此开关。在一个应用程序中分派多个工人时可能遇到的难题有：

- 死锁；
- 数据竞争；
- 优先权倒置；
- 无限延迟。

这些难题是在构建多线程程序时必须避免的缺陷。它们带来了程序员经常要逾越的障碍。程序员通过一套构建完美的面向对象技术，可以获得设计和编写中到大型单线程应用程序的指导方针，但对于多线程应用程序，却没有这样的指导方针。软件开发者在构建多线程程序时通常要反复排除。构建包含多个线程的程序，没有好的蓝图必定导致灾难。甚至一个典型的中型单线程应用程序可能有上百个组件，每个组件由上千行代码组成。如果我们可以让所有组件协同工作而不发生故障，我们就可以判断该任务可以圆满完成。对于同样的中型应用程序，如果我们添加多个线程，我们通常会带来极大的复杂性。我们的目标可能只是加速应用程序，或者让它更符合用户的需要，但我们反而创建了一个脆弱的应用程序，容易遭受不可预测的崩溃，而且不可能对它进行维护。

解决方案

我们使用C++组件来消除编写多线程程序时遇到的缺陷。我们通过面向对象技术来实现多线程应用程序。本书讨论用一种架构途径来构建多线程应用程序，还解释了如何构建用于实现支持

并发的软件机制的C++组件。本书讨论的这类软件机制有：

- 互斥量对象；
- 事件互斥量对象；
- 匿名管道对象；
- 命名管道对象；
- 线程对象；
- 容器组件；
- 应用框架。

我们使用这些软件机制来探索增量多线程编程的概念，它使用封装、接口类、宿主类以及域类来构建多线程组件。然后组合多线程组件形成面向对象多线程架构。多线程架构成为构建可靠多线程应用程序的基础。为了使问题简单化，我们构建多线程应用程序时，一次只讨论一个组件。我们从构建支持并发的简单软件组件开始。将运行并发的组件与多线程化的宿主对象结合起来。再结合宿主对象与域对象形成多线程应用程序。

虽然本书主要是讲解C++组件和面向对象技术，但必须注意，C++语言并不包含任何支持多线程编程的特定命令、语句或关键字。多线程能力由现在的操作系统提供。它假定C++组件用于多线程环境中。现代操作环境，如Solaris、Windows NT以及OS/2，它们都提供了支持多线程的系统服务。POSIX标准1003.1c提供了针对线程的API。针对线程的POSIX标准一般指pthreads。C++语言可以利用这些环境所支持的任何线程系统服务。本书讨论的C++组件可用于运行pthreads的任何环境、Win32或OS/2。

本书为谁而写

本书为软件工程师、软件开发人员、程序员、教育工作者、研究人员以及需要一种面向对象方法开发多线程程序和应用的学生而写。本书读者需要对C++语言和标准C++类库有一定的了解，而不必具备多线程编程技术方面的经验。本书不是一本C++编程或面向对象编程的指导书。它假定读者能够基本理解像封装、继承和多态这样的面向对象编程技术。本书介绍和解释不同类型的C++组件。其中对操作系统进程的概念进行了介绍性地综述。我们还对核心线程和用户线程进行了详细的综述。定义和描述了多任务处理以及操作系统规划策略的概念。

本书的组织

第1章概览用于构建面向对象程序的不同类型C++组件。第1章还简要介绍了如何使用这些组件来构建多线程架构。第2、3和4章对进程、线程、多任务处理、多线程化、规划以及线程优先权的概念进行了扼要的介绍。如果读者熟悉基本C++类类型以及基本操作系统概念（这些都是理解线程的必需的知识），那么就可以跳过第1章到第4章的内容。第5章讨论进程间和线程通信。第6章讨论线程与进程同步以及合作。第6章应用了第5章讨论的概念，定义和讨论运行并发的程序中碰到的主要问题。也就是详细讨论了临界区、死锁、数据竞争以及无限延迟方面的主题。

第 7 章到第 10 章讲解用于线程同步、线程间通信、进程间通信以及多线程处理的 C++组件。第 11 章讨论 C++对象在多线程环境中的行为和交互方式。第 12 章给予读者测试多线程应用程序方面的提示。要应用我们所讨论的增量多线程处理技术，就必须完全理解第 5 章到第 11 章所讲解的知识。

类关系图

本书运用类关系图来描述相关类家族与集合、容器类层次之间的关系。对类关系图的详细解释参见配套资源中的附录 A。

支持的线程环境和编译器

本书中的所有主要代码示例都使用 POSIX pthreads、Win32 API 线程以及 OS/2 线程实现。尤其使用了针对 Linux 2.0 的 pthreads、Win32、OS/2 的核心线程。所有示例在运行 POSIX pthreads、Win32 API 或 OS/2 的任何环境中都能正常工作。虽然我们使用 Gnu C++、IBM 的 Viusal Age 以及 Borland 的 C++来测试这些例子，但支持 ANSI/ISO C++标准的任何 C++编译器都可以拿来编译本书中的示例。而且，本书中的大部分示例还可以运行在 VMS 和 AIX 上。

测试与代码可靠性

虽然我们对本书中的所有示例和应用都进行了测试，以确保其正确性，但我们不能保证本书所包含的程序没有任何缺陷与错误、与任何特殊商业标准一致，或者满足任何特殊应用的要求。不要依赖于它们来解决问题，不正确的解决方案可能会伤害到应用者本人，或者导致利益上的损失。

读者由于使用本书中包含的示例或应用程序带来直接或间接的损失，作者与出版商并不承担任何赔偿义务。

致谢

感谢 Paul Mullins 博士、Ray Sweeney 教授、Raoul Hewitt 教授和 Debra Whitfield 博士，他们审阅了本书的初稿。还要感谢 IEEE 的 Mary Shepard，她帮助我们搜集 POSIX pthread 信息。特别要感谢的是我的家庭、朋友，以及在本书撰写过程中，参与“并行”深入讨论的相关人士。

目录

第 1 章　C++组件简介……1

1.1　既是好消息，也是坏消息……1
1.2　面向对象方法……2
1.3　面向对象架构……2
1.4　C++组件……3
1.5　面向对象软件组件……3
1.5.1　什么是类……4
1.5.2　抽象数据类型……4
1.5.3　类作为模型……8
1.5.4　类类型……9

第 2 章　进程解剖……25

2.1　什么是进程……25
2.2　进程状态……28
2.3　进程优先权……34
2.4　上下文切换……35
2.5　进程关系……35
2.5.1　进程终止……39
2.5.2　同步和异步进程……39
2.6　进程映射……41
2.7　进程资源……42

2.7.1 硬件资源……43
2.7.2 数据资源……44
2.7.3 软件资源……44
2.7.4 优先权与资源……44

第3章 轻量级进程：线程……47

3.1 多线程处理……49
3.2 线程与进程的相似之处……52
3.3 线程与进程的不同之处……52
3.4 线程的优点……53
3.5 线程的缺点……54
3.6 线程类型……54
3.6.1 休眠（sleeper）和单步（one-shot）……54
3.6.2 先占工作……55
3.6.3 延迟工作……55
3.7 线程相关信息……55
3.8 线程创建……57
3.8.1 谁可以终止线程……58
3.8.2 分离线程……60
3.8.3 远程线程……60
3.9 线程堆栈……60
3.10 线程控制……61
3.10.1 临界区……61
3.10.2 挂起和恢复线程……63
3.11 线程优先权……64
3.12 线程状态……70
3.13 线程与资源……71
3.14 线程的实现模型：用户级线程……71
3.14.1 核心级线程……71
3.14.2 混合线程……71

第4章 多任务与多线程编程……73

4.1 什么是多任务编程……73
4.1.1 对话级多任务编程……74
4.1.2 进程级多任务编程……74

4.1.3 多线程编程……75
4.2 合作和抢占式多任务……75
4.2.1 合作多任务……76
4.2.2 抢占式多任务……78
4.2.3 时间片的大小……79
4.3 多处理器下的多线程……80
4.3.1 非对称多处理器处理……81
4.3.2 对称多处理器处理……81
4.3.3 具有多处理器的多线程处理模型……82
4.4 规划策略……84
4.4.1 规划策略目标……84
4.4.2 规划策略准则……85
4.4.3 轮询和 FIFO 规划……86
4.4.4 最短任务优先规划法……87
4.4.5 最短剩余时间规划法……88

第 5 章 进程间和线程间通信……89

5.1 依赖关系……89
5.1.1 通信依赖性……90
5.1.2 合作依赖性……91
5.1.3 计数线程与进程依赖性……91
5.2 进程间和线程间通信……94
5.2.1 什么是进程间通信……94
5.2.2 进程间通信类型……95
5.3 线程间通信……114

第 6 章 合作与同步……123

6.1 竞争条件……123
6.1.1 数据同步……127
6.1.2 硬件同步……127
6.1.3 任务同步……128
6.2 同步关系……129
6.3 进程同步机制……130
6.3.1 信号量提供钥匙……130
6.3.2 信号量类型……132

6.3.3 自愿互斥量策略……136
6.3.4 使用互斥量锁定防止竞争条件……136
6.3.5 临界区……145
6.4 避免竞争条件……147
6.5 死锁必需的条件……148
6.6 远离死锁……148

第 7 章 接口类与进程间通信……149

7.1 接口类详解……149
7.1.1 接口类的类型……149
7.1.2 减小参数和全局变量的数量……155
7.2 C++没有多线程处理的关键字……157
7.3 面向对象接口到管道……158
7.4 使用接口类来实现面向对象命名管道……173
7.4.1 相关客户/服务器术语……174
7.4.2 名字包含哪些内容……174
7.4.3 命名管道和 iostream 复合……175
7.4.4 npstream 接口类……176
7.4.5 命名管道与 STL istream_iterator 和 ostream_iterator……181

第 8 章 同步对象……185

8.1 初识 mutex 类……186
8.1.1 命名互斥量类……195
8.1.2 同步和依赖性关系（示例）……199
8.1.3 表示条件的类……206
8.1.4 等待多个事件或互斥量……213
8.1.5 通过类成员函数锁定和取消锁定……217
8.1.6 小结……219

第 9 章 线程处理面向对象架构……221

9.1 什么是多线程架构……221
9.2 使用多线程的常见架构……223
9.2.1 文件服务器……225
9.2.2 数据库服务器和事务服务器……227
9.2.3 应用服务器……229

9.2.4 事件驱动架构……233
9.3 黑板架构……235
9.4 途径上的不同（面向对象与过程化）……237
9.4.1 封装是关键（保护和数据隐藏）……239
9.4.2 类成员函数 CREW 策略……251
9.5 增量多线程处理……252

第 10 章 类层次和线程处理 C++组件……255

10.1 抽象基类……255
10.2 具体类——理想终结者……258
10.2.1 多线程层次中的节点类……260
10.2.2 线程与容器和集合类……261
10.2.3 应用框架类……276

第 11 章 类行为和线程处理……295

11.1 线程、对象和作用域……295
11.1.1 连接与作用域……296
11.1.2 线程和类作用域……297
11.2 同步关系和对象成员函数……297
11.3 在多线程环境中构建和析构对象……306
11.3.1 exit()和 abort()……306
11.3.2 构造函数和 SS 关系……307
11.3.3 析构函数与 FF 关系……308
11.3.4 线程集合与对象……309
11.3.5 线程与异常处理……311
11.4 线程安全函数……318
11.5 多线程环境中的不安全函数……319
11.6 在多线程架构中使用 STL 算法……320

第 12 章 测试多线程应用程序……323

12.1 软件测试的目标……323
12.1.1 分而治之（divide and conquer）……324
12.1.2 软件测试类型……326
12.1.3 对象的组件复合……327
12.1.4 成员函数访问数据组件……328

12.1.5 成员函数正确性……328
12.1.6 对象的过渡状态……329
12.1.7 成员函数调用序列……329
12.1.8 对象完整性……330
12.2 对象的测试实例……336
12.2.1 对象构建的测试实例……336
12.2.2 析构函数的测试实例……337
12.2.3 赋值的测试实例……338
12.2.4 对象派生子类……339
12.2.5 成员函数性能的测试实例……339
12.2.6 对象资源需求和测试实例……340
12.2.7 测试公有对象访问、受保护对象访问以及线程化对象访问……340
12.3 测试多线程架构的问题……341
12.3.1 开放层次问题……341
12.3.2 规划问题……341
12.4 使用常用模型和架构……342

第 13 章 实现并发的最后思考……345

附录 A POSIX 线程管理规范……349

附录 B 类关系图规范……387

附录 C POSIX 线程管理函数……391

附录 D Win32 线程管理函数……415

附录 E OS/2 线程管理函数……439

附录 F 线程和同步类（POSIX, Win32 以及 OS/2）……473

参考文献……503

索引……509

第1章

C++组件简介

相对于普通语言的一般表达能力而言，通过这种表示法可以让我们更清楚地表述我们希望表达的内容。同时可以准确表达复杂的思想而避免歧义，这是日常语言不能轻松解决的。不过，为以上优点必须付出代价：必须了解不熟悉的特性；必须掌握一种新的表示法。

Symbolic Notation, Haddock's Eyes and the Dog-Walking Ordinance
——*Ernest Nagel*[1]

最近 POSIX 线程标准在许多 UNIX 环境中（例如 Sunsoft 的 Solaris、IBM 的 AIX 和 Linux）的实现，以及 OS/2、NT 等其他多线程操作系统的成功，它们都为 C++程序员提供了真正的多线程环境。多任务处理（multitasking）允许同一时刻执行多个进程，而多线程处理（multithreading）允许一个进程在同一时刻执行多个任务。单个进程中执行的每个任务称做一个线程（thread）。当一个进程使用了多个线程时，该进程就称做多线程化（multithreaded）。程序员通过多线程可以将一个进程分解成一系列可以同时执行的线程。如果进程运行于拥有多个处理器（processor）的计算机上，每个线程可以在单独的处理器上执行。借此，程序员可以实现在一个程序中能够并行执行的任务。

1.1 既是好消息，也是坏消息

多线程环境的实现带给 C++程序员的既是好消息，也是坏消息。好消息是，由于为 C++程序员提供了软件设计和软件开发阶段应用并发（concurrency）和并行（parallelism）操作的能力，多线程为解决软件操作性能问题和软件组织问题开辟了一条全新的途径。对于 C++程序员来说，并

1译者注：摘自 Ernest Nagel 发表于 *World of Mathematics* 杂志上的文章 *Symbolic Notation, Haddock's Eyes and the Dog-Walking Ordinance*。