

[首页](#) | [技术频道](#) | [51CTO旗下网站](#) | [地图](#) | [RSS](#)[登录](#) | [注册](#) | [招聘](#) | [学院](#) | [下载](#) | [论坛](#) | [博客](#) | [更多](#)

网络安全一个大写的反“作死”  
Android开发120天完成9个APP项目  
微软企业级内训课程免费学  
2016年备战软考-重难点解析

## 移动开发

[首页](#) | [Android](#) | [iOS](#) | [Windows Phone](#) | [BlackBerry](#) | [webOS](#) | [Symbian](#) | [bada](#) | [OPhone](#) | [其他](#)

请输入关键字

搜索

您所在的位置: [移动开发](#) > [热点推荐](#) > [iOS多线程编程指南\(一\)关于多线程编程](#)

## iOS多线程编程指南(一)关于多线程编程

2013-07-16 10:12 佚名 dreamingwish 字号: T | T

收藏 +

多线程是一个比较轻量级的方法来实现单个应用程序内多个代码执行路径。在系统级别内，程序并排执行，系统分配到每个程序的执行时间是基于该程序的所需时间和其他程序的所需时间来决定的。然而在每个应用程序的内部，存在一个或多个执行线程，它同时或在一个几乎同时发生的方式里执行不同的任务。

AD: 51CTO 网+ 第十二期沙龙: 大话数据之美\_如何用数据驱动用户体验

多年来，计算机的最大性能主要受限于它的中心微处理器的速度。然而由于个别处理器已经开始达到它的瓶颈限制，芯片制造商开始转向多核设计，让计算机具有了同时执行多个任务的能力。尽管Mac OS X利用了这些核心优势，在任何时候可以执行系统相关的任务，但自己的应用程序也可以通过多线程方法利用这些优势。

## 1. 什么是多线程

多线程是一个比较轻量级的方法来实现单个应用程序内多个代码执行路径。在系统级别内，程序并排执行，系统分配到每个程序的执行时间是基于该程序的所需时间和其他程序的所需时间来决定的。然而在每个应用程序的内部，存在一个或多个执行线程，它同时或在一个几乎同时发生的方式里执行不同的任务。系统本身管理这些执行的线程，调度它们在可用的内核上运行，并在需要让其他线程执行的时候抢先打断它们。

从技术角度来看，一个线程就是一个需要管理执行代码的内核级和应用级数据结构组合。内核级结构协助调度线程事件，并抢占式调度一个线程到可用的内核之上。应用级结构包括用于存储函数调用的调用堆栈和应用程序需要管理和操作线程属性和状态的结构。

在非并发的应用程序，只有一个执行线程。该线程开始和结束于你应用程序的main循环，一个个方法和函数的分支构成了你整个应用程序的所有行为。与此相反，支持并发的应用程序开始可以在需要额外的执行路径时候创建一个或多个线程。每个新的执行路径有它自己独立于应用程序main循环的定制开始循环。在应用程序中存在多个线程提供了两个非常重要的潜在优势：

1. 多个线程可以提高应用程序的感知响应。
2. 多个线程可以提高应用程序在多核系统上的实时性能。

如果你的应用程序只有单独的线程，那么该独立程序需要完成所有的事情。它必须对事件作出响应，更新您的应用程序的窗口，并执行所有实现你应用程序行为需要的计算。拥有单独线程的主要问题是同一时间里它只能执行一个任务。那么当你的应用程序需要很长时间才能完成的时候会发生什么呢？当你的代码忙于计算你所需要的值的时候，你的程序就会停止响应用户事件和更新它的窗口。如果这样的情况持续足够长的时间，用户就会误认为你的程序被挂起了，并试图强制退出。如果你把你的计算任务转移到一个独立的线程里面，那么你的应用程序主线程就可以自由并及时响应用户的交互。



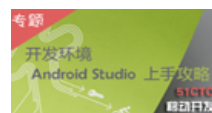
关注有礼  
51CTO官方微信  
weixin51cto



3万免费IT视频课程  
随时随地学习



## 专题 Android Studio上手攻略



既然强大的Android Studio来了，有什么理由不去用呢？

[iOS新语言swift上手指南](#)[MIUI 6 测评：细节的量变](#)

## 文章排行

24小时 | 本周 | 本月

21个免费的UI界面设计工具、资源及网站  
在Eclipse下搭建Android开发环境教程  
人人都是开发者：5款傻瓜式APP开发工具  
三大移动Web开发框架哪个适合你？  
哪个市场最挣钱，腾讯告诉你！  
阿里：这是全世界最大的打假团队  
Android应当从iOS处窃取的五项最佳功能  
iOS 8出色的跨应用通信效果：解读Actio  
看库克如何干掉Android  
为什么说2015年将是微服务架构元年？

## 热点职位

[更多>>](#)

## 移动客户端研发工程师

全职/1-3年/本科 12k-25k 多盟

## SDK开发工程师

全职/1-3年/大专 7k-10k 游众游戏

当然多线程并不是解决程序性能问题的灵丹妙药。多线程带来好处同时也伴随着潜在问题。应用程序内拥有多个可执行路径，会给你的代码增加更多的复杂性。每个线程需要和其他线程协调其行为，以防止它破坏应用程序的状态信息。因为应用程序内的多个线程共享内存空间，它们访问相同的数据结构。如果两个线程试图同时处理相同的数据结构，一个线程有可能覆盖另外线程的改动导致破坏该数据结构。即使有适当的保护，你仍然要注意由于编译器的优化导致给你代码产生很微妙的（和不那么微妙）的Bug。

2. 线程术语

在讨论多线程和它支持的相关技术之前，我们有必要先了解一些基本的术语。如果你熟悉Carbon的多处理器服务API或者UNIX系统的话，你会发现本文档里面“任务(task)”被用于不同的定义。在Mac OS的早期版本，术语“任务(task)”是用来区分使用多处理器服务创建的线程和使用Carbon线程管理API创建的线程。在UNIX系统里面，术语“任务(task)”也在一段时间内被用于指代运行的进程。在实际应用中，多处理器服务任务是相当于抢占式的线程。

由于Carbon线程管理器和多处理器服务API是Mac OS X的传统技术，本文件采用下列术语：

- 1. **线程(线程)**用于指代独立执行的代码段。
- 2. **进程(process)**用于指代一个正在运行的可执行程序，它可以包含多个线程。
- 3. **任务(task)**用于指代抽象的概念，表示需要执行工作。

3. 多线程的替代方法

你自己创建多线程代码的一个问题就是它会给你的代码带来不确定性。多线程是一个相对较低的水平和复杂的方式来支持你的应用程序并发。如果你不完全理解你的设计选择的影响，你可能很容易遇到同步或定时问题，其范围可以从细微的行为变化到严重到让你的应用程序崩溃并破坏用户数据。

你需要考虑的另一个因素是你是否真的需要多线程或并发。多线程解决了如何在同一个进程内并发的执行多路代码路径的问题。然而在很多情况下你是无法保证你所在做的工作是并发的。多线程引入带来大量的开销，包括内存消耗和CPU占用。你会发现这些开销对于你的工作而言实在太，或者有其他方法会更容易实现。

表1-1列举了多线程的替代方法。该表包含了多线程的替代技术(比如操作对象和GCD)和如何更高效的使用单个线程。

Table 1-1 Alternative technologies to threads

Technology	Description
Operation objects	<p>Introduced in Mac OS X v10.5, an operation object is a wrapper for a task that would normally be executed on a secondary thread. This wrapper hides the thread management aspects of performing the task, leaving you free to focus on the task itself. You typically use these objects in conjunction with an operation queue object, which actually manages the execution of the operation objects on one more threads. For more information on how to use operation objects, see <a href="#">Concurrency Programming Guide</a>.</p>

APP开发工程师

全职/1-3年/本科10k-20k浙江长龙航空

Android软件工程师

全职/3-5年/大专6k-10k广州太乙科技

安卓工程师

全职/3-5年/大专4k-8k索引教育

热点专题更多>>



**聚焦微信5.0 改变与创新**  
微信自引入语音短信功能用户量爆发之后，发展状况一直



**Windows Phone开发浅**  
Windows Phone开发创建吸引  
人、带给人快乐并保留用户



**Android各类顶级资源**  
作为Android开发人员，在开发项目的过程中，我们往往

热点标签

iOS开发Android开发Symbian开发MeeGo开发

BlackBerry开发Windows Phone开发Phone Club

Android 4.0webOS 3.0智能手机软件下载

点击这里查看样刊

立即订阅

全站热点





《Linux运维趋势》2013 说说HCC2013的那些事儿

能效当先 凌动C2000冷数据处理解析


移动周回顾:马化腾复仇记 周鸿祎的新

浪潮发布凌动微服务器 全球五强地位

横扫三大领域 英特尔“Avoton”看点

如何为虚拟化扩展购买服务器新硬件

读书



**Java网络编程精解**  
本书结合大量的典型实例，详细介绍了用Java来编写网络应用程序的技术。本书的范例都基于最新的JDK 1.5版本，书中内容包括：Java

Grand Central Dispatch (GCD)	<p>Introduced in Mac OS x v10.6, Grand Central Dispatch is another alternative to threads that lets you focus on the tasks you need to perform rather than on thread management. With GCD, you define the task you want to perform and add it to a work queue, which handles the scheduling of your task on an appropriate thread. Work queues take into account the number of available cores and the current load to execute your tasks more efficiently than you could do yourself using threads.</p> <p>For information on how to use GCD and work queues, see <a href="#">Concurrency Programming Guide</a></p>
Idle-time notifications	<p>For tasks that are relatively short and very low priority, idle time notifications let you perform the task at a time when your application is not as busy. Cocoa provides support for idle-time notifications using the <a href="#">NSNotificationQueue</a> object. To request an idle-time notification, post a notification to the default NSNotificationQueue object using the <a href="#">NSPostWhenIdle</a> option. The queue delays the delivery of your notification object until the run loop becomes idle. For more information, see <a href="#">Notification Programming Topics</a>.</p>
Asynchronous functions	<p>The system interfaces include many asynchronous functions that provide automatic concurrency for you. These APIs may use system daemons and processes or create custom threads to perform their task and return the results to you. (The actual implementation is irrelevant because it is separated from your code.) As you design your application, look for functions that offer asynchronous behavior and consider using them instead of using the equivalent synchronous function on a custom thread.</p>
Timers	<p>You can use timers on your application's main thread to perform periodic tasks that are too trivial to require a thread, but</p>

公钥基础设施PKI及其应用  
SQL Server 2005实现与维护 (MCTS教程)  
人月神话：32周年中文纪念版  
C#高级编程 (第4版)

博文推荐

更多>>



nowpaper

Windows  
Phone专家



himi

Android开  
发专家

从奥运会到亚运会 IT技术总是不给力  
RSA2010中国大会的亮点和遗憾之处  
项目管理学习笔记四：立项管理  
央视2.4秒门对管理的启示

最新热帖

更多>>

一般晓通或者神马面试之后 一般几天  
求职： LINUX嵌入式开发  
发一下思科中国的总代以及金牌合作公  
我想找一个综合布线的工作，请问工资  
面试之自我救赎

http://mobile.51cto.com/hot-403061\_all.htm

第 3 页 (共 12 页)

	which still require servicing at regular intervals. For information on timers, see <a href="#">“Timer Sources.”</a>
Separate processes	Although more heavyweight than threads, creating a separate process might be useful in cases where the task is only tangentially related to your application. You might use a process if a task requires a significant amount of memory or must be executed using root privileges. For example, you might use a 64-bit server process to compute a large data set while your 32-bit application displays the results to the user.

注意: 当使用fork函数加载独立进程的时候, 你必须总是在fork后面调用exec或者类似的函数。基于Core Foundation、Cocoa或者Core Data框架（无论显式还是隐式关联）的应用程序随后调用exec函数或者类似的函数都会导出不确定的结果。

别走开，下页内容更精彩

4. 线程支持

如果你已经有代码使用了多线程, Mac OS X和iOS提供几种技术来在你的应用程序里面创建多线程。此外, 两个系统都提供了管理和同步你需要在这些线程里面处理的工作。以下几个部分描述了一些你在Mac OS X和iOS上面使用多线程的时候需要注意的关键技术。

4.1 线程包

虽然多线程的底层实现机制是Mach的线程, 你很少（即使有）使用Mach级的线程。相反, 你会经常使用到更多易用的POSIX 的API或者它的衍生工具。Mach的实现没有提供多线程的基本特征, 但是包括抢占式的执行模型和调度线程的能力, 所以它们是相互独立的。

列表1-2列举你可以在你的应用程序使用的线程技术。

Table 1-2 Thread technologies

Technology	Description
Cocoa threads	Cocoa implements threads using the <a href="#">NSThread</a> class. Cocoa also provides methods on <a href="#">NSObject</a> for spawning new threads and executing code on already-running threads. For more information, see <a href="#">“Using NSThread”</a> and <a href="#">“Using NSObject to Spawn a Thread.”</a>
POSIX threads	POSIX threads provide a C-based interface for creating threads. If you are not writing a Cocoa application, this is the best choice for creating threads. The POSIX interface is relatively simple to use and offers ample

	flexibility for configuring your threads. For more information, see <a href="#">“Using POSIX Threads”</a>
Multiprocessing Services	Multiprocessing Services is a legacy C-based interface used by applications transitioning from older versions of Mac OS. This technology is available in Mac OS X only and should be avoided for any new development. Instead, you should use the NSThread class or POSIX threads. If you need more information on this technology, see <i>Multiprocessing Services Programming Guide</i> .

在应用层上，其他平台一样所有线程的行为本质上是相同的。线程启动之后，线程就进入三个状态中的任何一个:运行(running)、就绪(ready)、阻塞(blocked)。如果一个线程当前没有运行，那么它不是处于阻塞，就是等待外部输入，或者已经准备就绪等待分配CPU。线程持续在这三个状态之间切换，直到它最终退出或者进入中断状态。

当你创建一个新的线程，你必须指定该线程的入口点函数（或Cocoa线程时候为入口点方法）。该入口点函数由你想要在该线程上面执行的代码组成。但函数返回的时候，或你显式的中断线程的时候，线程永久停止，且被系统回收。因为线程创建需要的内存和时间消耗都比较大，因此建议你的入口点函数做相当数量的工作，或建立一个运行循环允许进行经常性的工作。

为了获取更多关于线程支持的可用技术并且如何使用它们，请阅读“线程管理部分”。

4. 2Run Loops

**注:**为了便于记忆，文本后面部分翻译Run Loops的时候基本采用原义，而非翻译为“运行循环”。

一个run loop是用来在线程上管理事件异步到达的基础设施。一个run loop为线程监测一个或多个事件源。当事件到达的时候，系统唤醒线程并调度事件到run loop, 然后分配给指定程序。如果没有事件出现和准备处理，run loop把线程置于休眠状态。

你创建线程的时候不需要使用一个run loop，但是如果你这么做的话可以给用户带来更好的体验。Run Loops可以让你使用最小的资源来创建长时间运行线程。因为run loop在没有任何事件处理的时候会把它线程置于休眠状态，它消除了消耗CPU周期轮询，并防止处理器本身进入休眠状态并节省电源。

为了配置run loop, 你所需要做的是启动你的线程，获取run loop的对象引用，设置你的事件处理程序，并告诉run loop运行。Cocoa和Carbon提供的基础设施会自动为你的主线程配置相应的run loop。如果你打算创建长时间运行的辅助线程，那么你必须为你的线程配置相应的run loop。

关于run loops的详细信息和如何使用它们的例子会在“Run Loops”部分介绍。

4. 3同步工具

线程编程的危害之一是在多个线程之间的资源争夺。如果多个线程在同一个时间试图使用或者修改同一个资源，就会出现问题。缓解该问题的方法之一是消除共享资源，并确保每个线程都有在它操作的资源上面的独特设置。因为保持完全独立的资源是不可行的，所以你可能必须使用锁，条件，原子操作和其他技术来同步资源的访问。



锁提供了一次只有一个线程可以执行代码的有效保护形式。最普遍的一种锁是互斥排他锁，也就是我们通常所说的“**mutex**”。当一个线程试图获取一个当前已经被其他线程占据的互斥锁的时候，它就会被阻塞直到其他线程释放该互斥锁。系统的几个框架提供了对互斥锁的支持，虽然它们都是基于相同的底层技术。此外Cocoa提供了几个互斥锁的变种来支持不同的行为类型，比如递归。获取更多关于锁的种类的信息，请阅读“锁”部分内容。

除了锁，系统还提供了条件，确保在你的应用程序任务执行的适当顺序。一个条件作为一个看门人，阻塞给定的线程，直到它代表的条件变为真。当发生这种情况的时候，条件释放该线程并允许它继续执行。POSIX级别和基础框架都直接提供了条件的支持。（如果你使用操作对象，你可以配置你的操作对象之间的依赖关系的顺序确定任务的执行顺序，这和条件提供的行为非常相似）。

尽管锁和条件在并发设计中使用非常普遍，原子操作也是另外一种保护和同步访问数据的方法。原子操作在以下情况的时候提供了替代锁的轻量级的方法，其中你可以执行标量数据类型的数学或逻辑运算。原子操作使用特殊的硬件设施来保证变量的改变在其他线程可以访问之前完成。

获取更多关于可用同步工具信息，请阅读“同步工具”部分。

4. 4线程间通信

虽然一个良好的设计最大限度地减少所需的通信量，但在某些时候，线程之间的通信显得十分必要。（线程的任务是为你的应用程序工作，但如果从来没有使用过这些工作的结果，那有什么好处呢？）线程可能需要处理新的工作要求，或向你应用程序的主线程报告其进度情况。在这些情况下，你需要一个方式来从其他线程获取信息。幸运的是，线程共享相同的进程空间，意味着你可以有大量的可选项来进行通信。

线程间通信有很多种方法，每种都有它的优点和缺点。“配置线程局部存储”列出了很多你可以在Mac OS X上面使用的通信机制。（异常的消息队列和Cocoa分布式对象，这些技术也可在iOS用来通信）。本表中的技术是按照复杂性的顺序列出。

Table 1-3Communication mechanisms

Mechanism	Description
Direct messaging	Cocoa applications support the ability to perform selectors directly on other threads. This capability means that one thread can essentially execute a method on any other thread. Because they are executed in the context of the target thread, messages sent this way are automatically serialized on that thread. For information about input sources, see <a href="#">“Cocoa Perform Selector Sources.”</a>
Global variables, shared memory,	Another simple way to communicate information between two threads is to use a global variable, shared object, or shared block of memory. Although shared variables are fast and simple, they are also more

and objects	fragile than direct messaging. Shared variables must be carefully protected with locks or other synchronization mechanisms to ensure the correctness of your code. Failure to do so could lead to race conditions, corrupted data, or crashes.
Conditions	Conditions are a synchronization tool that you can use to control when a thread executes a particular portion of code. You can think of conditions as gate keepers, letting a thread run only when the stated condition is met. For information on how to use conditions, see <a href="#">“Using Conditions.”</a>
Run loop sources	A custom run loop source is one that you set up to receive application-specific messages on a thread. Because they are event driven, run loop sources put your thread to sleep automatically when there is nothing to do, which improves your thread’s efficiency. For information about run loops and run loop sources, see <a href="#">“Run Loops.”</a>
Ports and sockets	Port-based communication is a more elaborate way to communication between two threads, but it is also a very reliable technique. More importantly, ports and sockets can be used to communicate with external entities, such as other processes and services. For efficiency, ports are implemented using run loop sources, so your thread sleeps when there is no data waiting on the port. For information about run loops and about port-based input sources, see <a href="#">“Run Loops.”</a>
Message queues	The legacy Multiprocessing Services defines a first-in, first-out (FIFO) queue abstraction for managing incoming and outgoing data. Although message queues are simple and convenient, they are not as efficient as some other communications techniques. For more information about how to use message queues, see <i>Multiprocessing Services Programming Guide</i> .

Cocoa distributed objects

Distributed objects is a Cocoa technology that provides a high-level implementation of port-based communications. Although it is possible to use this technology for inter-thread communication, doing so is highly discouraged because of the amount of overhead it incurs. Distributed objects is much more suitable for communicating with other processes, where the overhead of going between processes is already high. For more information, see *Distributed Objects Programming Topics*.

别走开，下页内容更精彩

## 5. 设计技巧

以下各节帮助你实现自己的线程提供了指导，以确保你代码的正确性。部分指南同时提供如何利用你的线程代码获得更好的性能。任何性能的技巧，你应该在你更改你代码之前、期间、之后总是收集相关的性能统计数据。

### 5.1 避免显式创建线程

手动编写线程创建代码是乏味的，而且容易出现错误，你应该尽可能避免这样做。Mac OS X 和 iOS 通过其他 API 接口提供了隐式的并发支持。你可以考虑使用异步 API，GCD 方式，或操作对象来实现并发，而不是自己创建一个线程。这些技术背后为你做了线程相关的工作，并保证是无误的。此外，比如 GCD 和操作对象技术被设计用来管理线程，比通过自己的代码根据当前的负载调整活动线程的数量更高效。关于更多 GCD 和操作对象的信息，你可以查阅“并发编程指南 (Concurrency Programming Guid)”。

### 5.2 保持你的线程合理的忙

如果你准备人工创建和管理线程，记得多线程消耗系统宝贵的资源。你应该尽最大努力确保任何你分配到线程的任务是运行相当长时间和富有成效的。同时你不应该害怕中断那些消耗最大空闲时间的线程。线程使用一个平凡的内存量，它的一些有线，所以释放一个空闲线程，不仅有助于降低您的应用程序的内存占用，它也释放出更多的物理内存使用的其他系统进程。线程占用一定量的内存，其中一些是有线的，所以释放空闲线程不但帮助你减少了你应用程序的内存印记，而且还能释放出更多的物理内存给其他系统进程使用。

**重要:** 在你中断你的空闲线程开始之前，你必须总是记录你应用程序当前的性能基线测量。当你尝试修改后，采取额外的测量来确保你的修改实际上提高了性能，而不是对它操作损害。

### 5.3 避免共享数据结构

避免造成线程相关资源冲突的最简单最容易的办法是给你应用程序的每个线程一份它需求的数据的副本。当最小化线程之间的通信和资源争夺时并行代码的效果最好。

创建多线程的应用是很困难的。即使你非常小心，并且在你的代码里面所有正确的地方锁住共享资源，你的代码依然可能语义不安全的。比如，当在一个特定的顺序里面修改共享数据结构的时候，你的代码有可能遇到问题。以原子方式修改你的代码，来弥补可能随后对多线程性能产生损耗的情况。把避免资源争夺放在首位通常可以得到简单的设计同样具有高性能的效果。



## 5.4多线程和你的用户界面

如果你的应用程序具有一个图形用户界面，建议你在主线程里面接收和界面相关的事件和初始化更新你的界面。这种方法有助于避免与处理用户事件和窗口绘图相关的同步问题。一些框架，比如Cocoa, 通常需要这样操作，但是它的事件处理可以不这样做，在主线程上保持这种行为的优势在于简化了管理你应用程序用户界面的逻辑。

有几个显著的例外，它有利于在其他线程执行图形操作。比如，QuickTime API包含了一系列可以在辅助线程执行的操作，包括打开视频文件，渲染视频文件，压缩视频文件，和导入导出图像。类似的，在Carbon和Cocoa里面，你可以使用辅助线程来创建和处理图片和其他图片相关的计算。使用辅助线程来执行这些操作可以极大提高性能。如果你不确定一个操作是否和图像处理相关，那么你应该在主线程执行这些操作。

关于QuickTime线程安全的信息，查阅Technical Note TN2125: “QuickTime的线程安全编程”。关于Cocoa线程安全的更多信息，查阅“线程安全总结”。关于Cocoa绘画信息，查阅Cocoa绘画指南 (Cocoa Drawing Guide)。

## 5.5了解线程退出时的行为

进程一直运行直到所有非独立线程都已经退出为止。默认情况下，只有应用程序的主线程是以非独立的方式创建的，但是你也可以使用同样的方法来创建其他线程。当用户退出程序的时候，通常考虑适当的立即中断所有独立线程，因为通常独立线程所做的工作都是可选的。如果你的应用程序使用后台线程来保存数据到硬盘或者做其他周期行的工作，那么你可能想把这些线程创建为非独立的来保证程序退出的时候不丢失数据。

以非独立的方式创建线程（又被称为可连接的）你需要做一些额外的工作。因为大部分上层线程封装技术默认情况下并没有提供创建可连接的线程，你必须使用POSIX API来创建你想要的线程。此外，你必须在你的主线程添加代码，来当它们最终退出的时候连接非独立的线程。更多关于创建可连接的线程信息，请查阅“设置线程的脱离状态”部分。

如果你正在编程Cocoa的程序，你也可以通过使用applicationShouldTerminate:的委托方法来延迟程序的中断直到一段时间后或者完成取消。当延迟中断的时候，你的程序需要等待直到任何周期线程已经完成它们的任务且调用了replyToApplicationShouldTerminate:方法。关于更多信息，请查阅NSApplication Class Reference。

## 5.6处理异常

当抛出一个异常时，异常的处理机制依赖于当前调用堆栈执行任何必要的清理。因为每个线程都有它自己的调用堆栈，所以每个线程都负责捕获它自己的异常。如果在辅助线程里面捕获一个抛出的异常失败，那么你的主线程也同样捕获该异常失败：它所属的进程就会中断。你无法捕获同一个进程里面其他线程抛出的异常。

如果你需要通知另一个线程（比如主线程）当前线程中的一个特殊情况，你应该捕捉异常，并简单地将消息发送到其他线程告知发生了什么事。根据你的模型和你正在尝试做的事情，引发异常的线程可以继续执行（如果可能的话），等待指示，或者干脆退出。

**注意：**在Cocoa里面，一个NSException对象是一个自包含对象，一旦它被引发了，那么它可以从一个线程传递到另外一个线程。

在一些情况下，异常处理可能是自动创建的。比如，Objective-C中的@synchronized包含了一个隐式的异常处理。

## 5.7干净地中断你的线程

线程自然退出的最好方式是让它达到其主入口结束点。虽然有不少函数可以用来立即中断线程，但是这些函数应仅用于作为最后的手段。在线程达到其自然结束点之前中断一个线程阻碍该线程清理完成它自己。如果线程已经分配了内存，打开了文件，或者获取了其他类型资源，你的代码可能没办法回收这些资源，结果造成内存泄漏或者其他潜在的问题。

关于更多正确退出线程的信息，请查阅“中断线程”部分。

## 5.8 线程安全的库

虽然应用程序开发人员控制应用程序是否执行多个线程，类库的开发者则无法这样控制。当开发类库时，你必须假设调用应用程序是多线程，或者多线程之间可以随时切换。因此你应该总是在你的临界区使用锁功能。

对类库开发者而言，只当应用程序是多线程的时候才创建锁是不明智的。如果你需要锁定你代码中的某些部分，早期应该创建锁对象给你的类库使用，更好是显式调用初始化类库。虽然你也可以使用静态库的初始化函数来创建这些锁，但是仅当没有其他方式的才应该这样做。执行初始化函数需要延长加载你类库的时间，且可能对你程序性能造成不利影响。

**注意：**永远记住在你的类库里面保持锁和释放锁的操作平衡。你应该总是记住锁定类库的数据结构，而不是依赖调用的代码提供线程安全环境。

如果你真正开发Cocoa的类库，那么当你想在应用程序变成多线程的时候收到通知的话，你可以给NSWillBecomeMultiThreadedNotification 注册一个观察者。不过你不应用依赖于这些收到的通知，因为它们可能在你的类库被调用之前已经被发出了。

### 【编辑推荐】

1. [iOS多线程初体验](#)
2. [iOS多线程编程之NSThread的使用](#)
3. [iOS多线程编程知多少](#)
4. [GCD实战一:使用串行队列实现简单的预加载](#)
5. [GCD实战二:资源竞争](#)

【责任编辑：milk TEL：（010）68476606】

原文：[iOS多线程编程指南\(一\)关于多线程编程](#)

[返回移动开发首页](#)



微信扫一扫  
移动首页版

优质技术内容尽收掌中

同时，您也可以在移动端浏览器上输入“[www.51cto.com](http://www.51cto.com)”随时随地浏览和分享最具价值的技术内容

### 微博推荐



51CTO移动开  
51CTO移  
动开发频道



51CTO官方微  
51cto官方  
微博



51CTO技术博  
51CTO技  
术博客官方



51CTO技术社  
51CTO技  
术社区(ho



51CTO熊平  
51CTO传  
媒总裁熊平



小林51CTO  
北京无忧创  
想信息技术

一键关注

[注册微博](#)

分享到:

0

[收藏](#) | [打印](#) | [复制](#)

给力

(0票)



动心

(0票)



废话

(0票)



专业

(0票)



标题党

(0票)



路过

(0票)

0 条评论, 0 人参与。

★ 0



我有话说...

[使用社交帐号登录](#)

发布前请先点击左边的按钮登录



## 最新评论

还没有评论

友言?

关于 [iOS多线程](#) [多线程概念](#) [多线程入门](#) 的更多文章[iOS多线程编程指南（拓展篇）](#)[iOS多线程编程指南\(四\)线程同步](#)[iOS多线程编程指南\(三\)Run Loop](#)[iOS多线程编程指南\(二\)线程管理](#)[iOS多线程编程指南\(一\)关于多线程编程](#)

## 全面掌握iOS多线程攻略



多线程是一个比较轻量级的方法来实现单个应用程序内多个代码执行[详细]

## 栏目热门

[更多>>](#)

100分的输家：一个146年历史的诺基亚为何4

李开复：移动互联网时代该如何创业

开源免费天气预报接口API以及全国所有地区

三星或将推“土豪金”Galaxy S4 死磕iPhone 5

微软收购诺基亚：移动开发者应该看到什么

## 同期最新

[更多>>](#)

写给初级游戏设计师的12条建议

支付宝的超级App野心：首屏App设置 打通支

逆天了！一个iOS 0-day漏洞卖出50万美元！

锤子 Smartisan OS α 版现正式发布

大爆炸：三星要和苹果“离婚”？

## 移动开发 频道导航

平台

[移动Web](#) | [Android](#) | [iOS](#) | [Windows Phone](#)

应用

[移动应用](#) | [移动团队](#) | [应用商店](#) | [专题汇总](#) | [Phone Club](#)

观察

[业界观察](#) | [调查数据](#) | [移动信息化](#)

Android

[热点](#) | [资讯](#) | [基础](#) | [多媒体](#) | [数据库](#) | [设计](#) | [工具](#) | [编译](#)

### 热门推荐



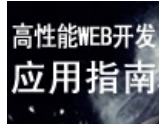
Android开发应用详解



那些性感的让人尖叫的程序员



HTML5 下一代Web开发标准详解



高性能WEB开发应用指南



Ubuntu开源技术交流频道

**热门标签:** windows频道 移动开发 云计算 objective-c tp-link路由器设置图解 html5

[弥合信息鸿沟，共享知识社会](#)

打造公益平台，传播公益资讯

[gongyi.baidu.com](#)

### 51CTO旗下网站

领先的IT技术网站 51CTO

领先的中文存储媒体 WatchStor

中国首个CIO网站 CIOage

中国首家数字医疗网站 HC3i

Copyright©2005-2016 51CTO.COM 版权所有 未经许可 请勿转载