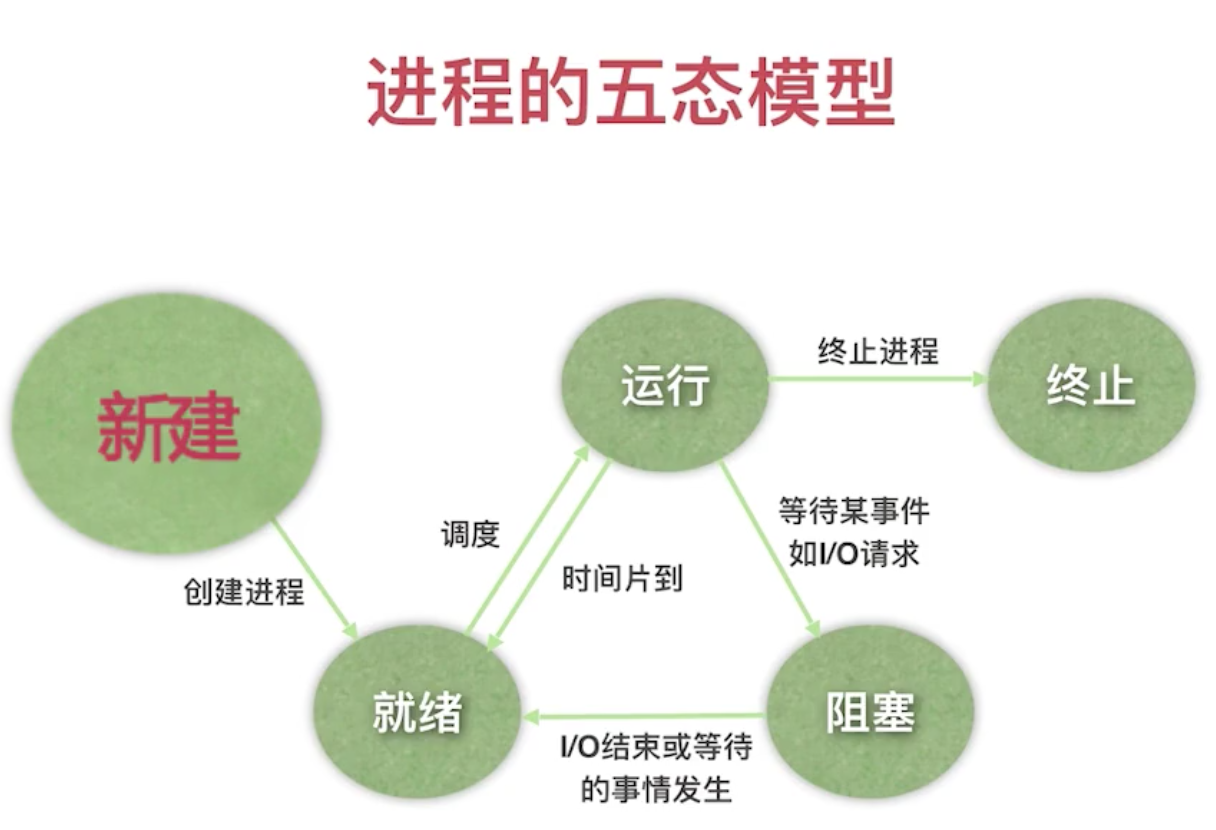
iOS多线程详解

**第一章 了解概念**

**进程：系统中正在运行的一个应用程序**

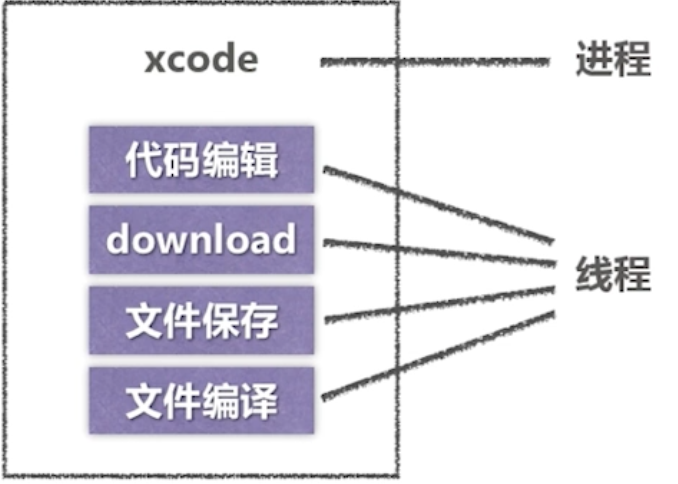
**（交替运行的）**

**五种状态：**

****

**线程：线程是进程的基本执行单元**

**进程的所有任务都在线程中执行**

****

**为什么要学习多线程？**

**——网络请求：**

**请求数据返回数据加载到页面上可能需要几秒钟的时间才能展示，所以我们需要一个异步请求，在不阻塞UI线程的基础下，发出请求完成相应的任务；换言之，在网络请求的同时我们依然可以对UI进行操作保证整个过程的流程性。**

**——图片加载：**

**图片加载一般有这样的过程：首先看缓存有没有该图片，若没有则进行网络请求，缓存到本地，以致于下次不用再请求了。**

**——文件处理**

**——数据存储**

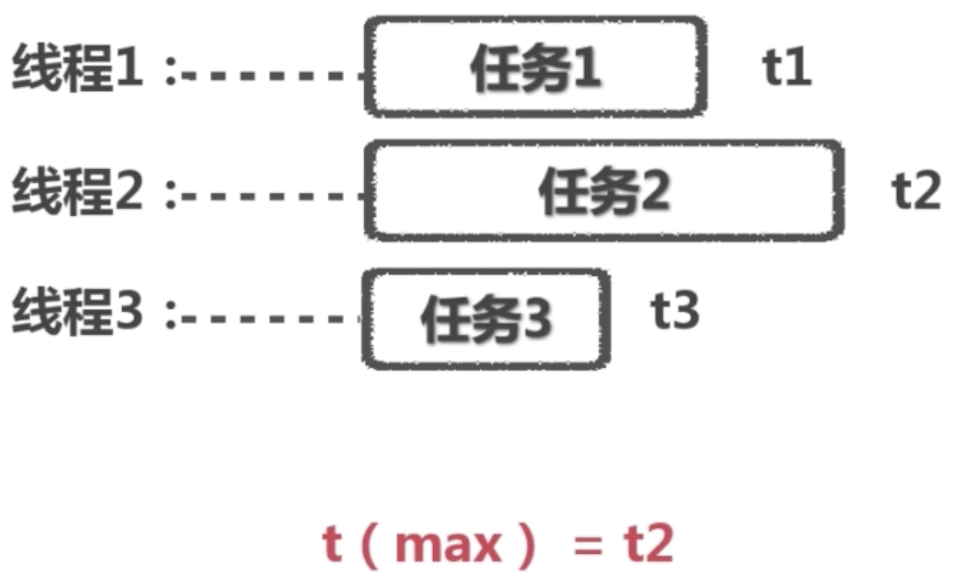
**——任务执行**

**任务的执行方式：**

**串行：**

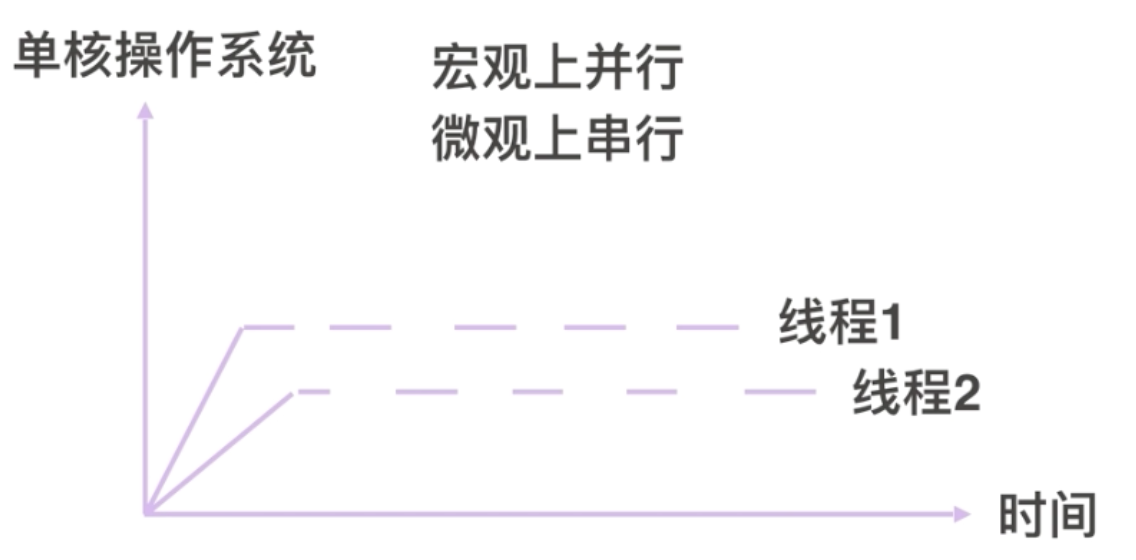
****

**并行：**

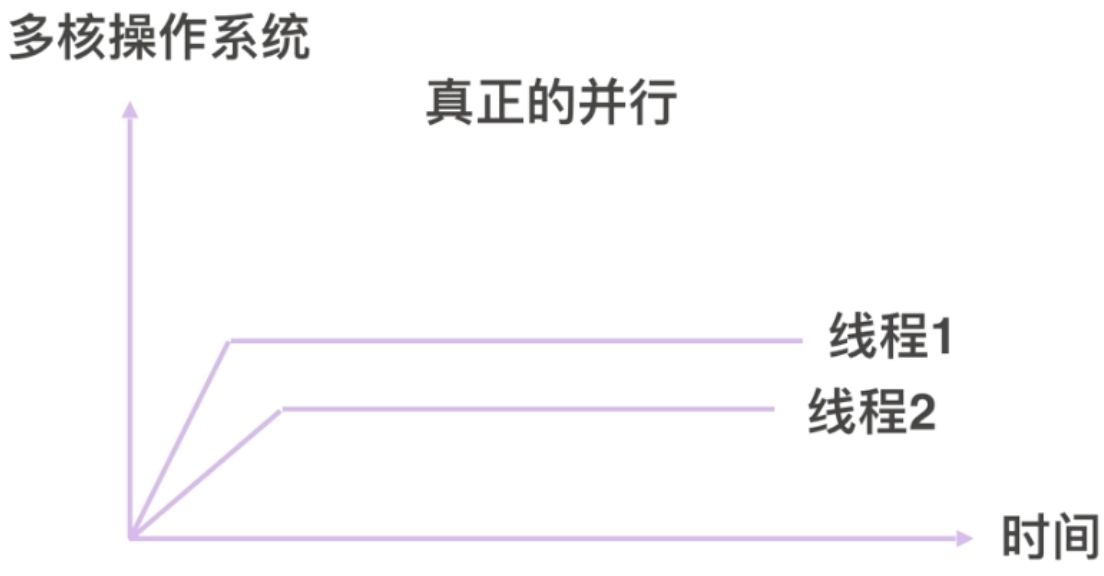
****

**多线程的实现原理：**

**时间片轮转：例如单核操作系统会按照时间片来分配时间给线程来执行，如图：**

****

**然而多核操作系统可以正真意义上的实现并行：**

****

**多线程的优缺点：**

**优点：**

1. **简化了编程模型**
2. **更加的轻量级**
3. **提高了执行效率**
4. **提高了资源利用率**

**缺点：**

**1.增加了程序设计的复杂性（例如：在一个线程中修改了某个变量的值，在另一个线程中也修改了其值，考虑是不是自己预期的值）**

**2.占用内存空间：考虑内存使用率**

**3.增大了CPU调度开销**

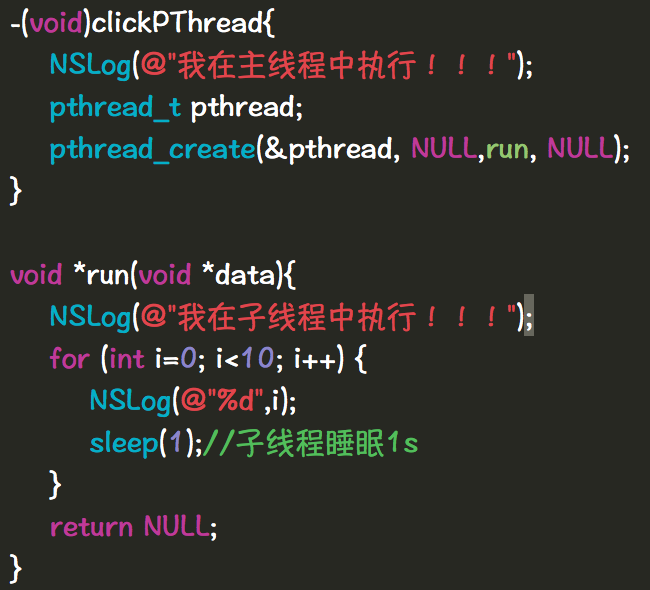
**第二章 多线程实现技术方案**

**pThread、NSThread、GCD、NSOperation这么四种。**

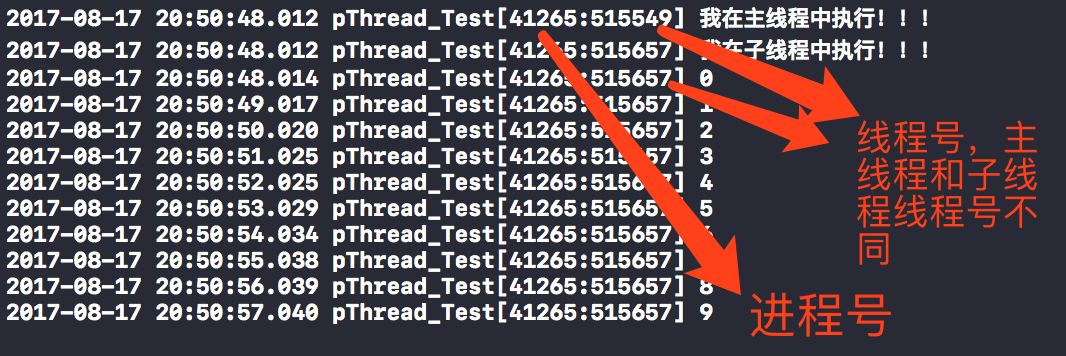
1. **pThread**

**简单来说就是操作系统级别使用的线程，基于c语言实现，我们的OC代码中很少用到，并且不便于管理。**

**代码说明：在视图中添加一个按钮方法是：clickPThread**

****

**打印结果：**

****

1. **NSThread**

**NSThread的三种初始化方法：**

**1、 通过alloc init方式创建并执行线程**

**NSThread \*thread1=[[NSThread alloc]initWithTarget:self selector:@selector(runThread1) object:nil];**

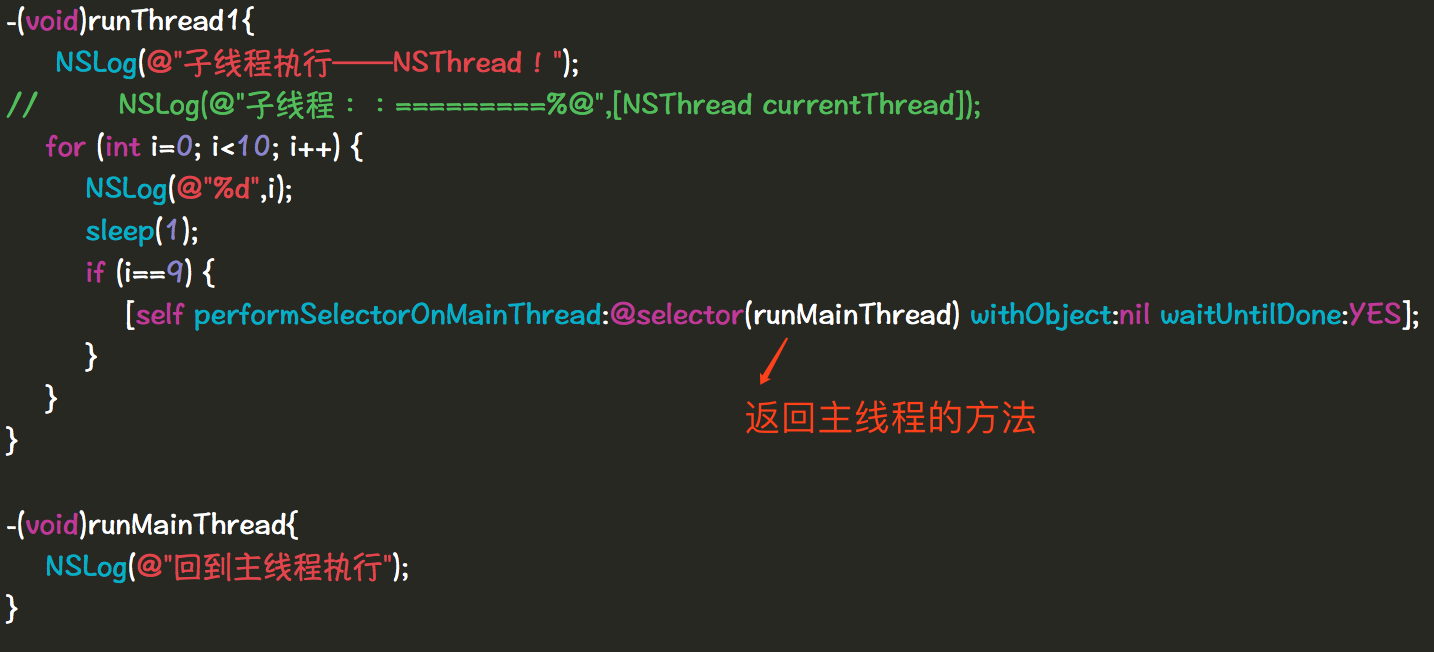
**[thread1 start];**

**2、通过detachNewThreadSelector方式创建并执行线程**

**[NSThread detachNewThreadSelector:@selector(runThread1) toTarget:self withObject:nil];**

**3.通过performSelectorInBackground方式创建线程**

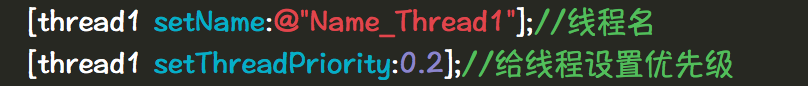
**[self performSelectorInBackground:@selector(runThread1) withObject:nil];**

****

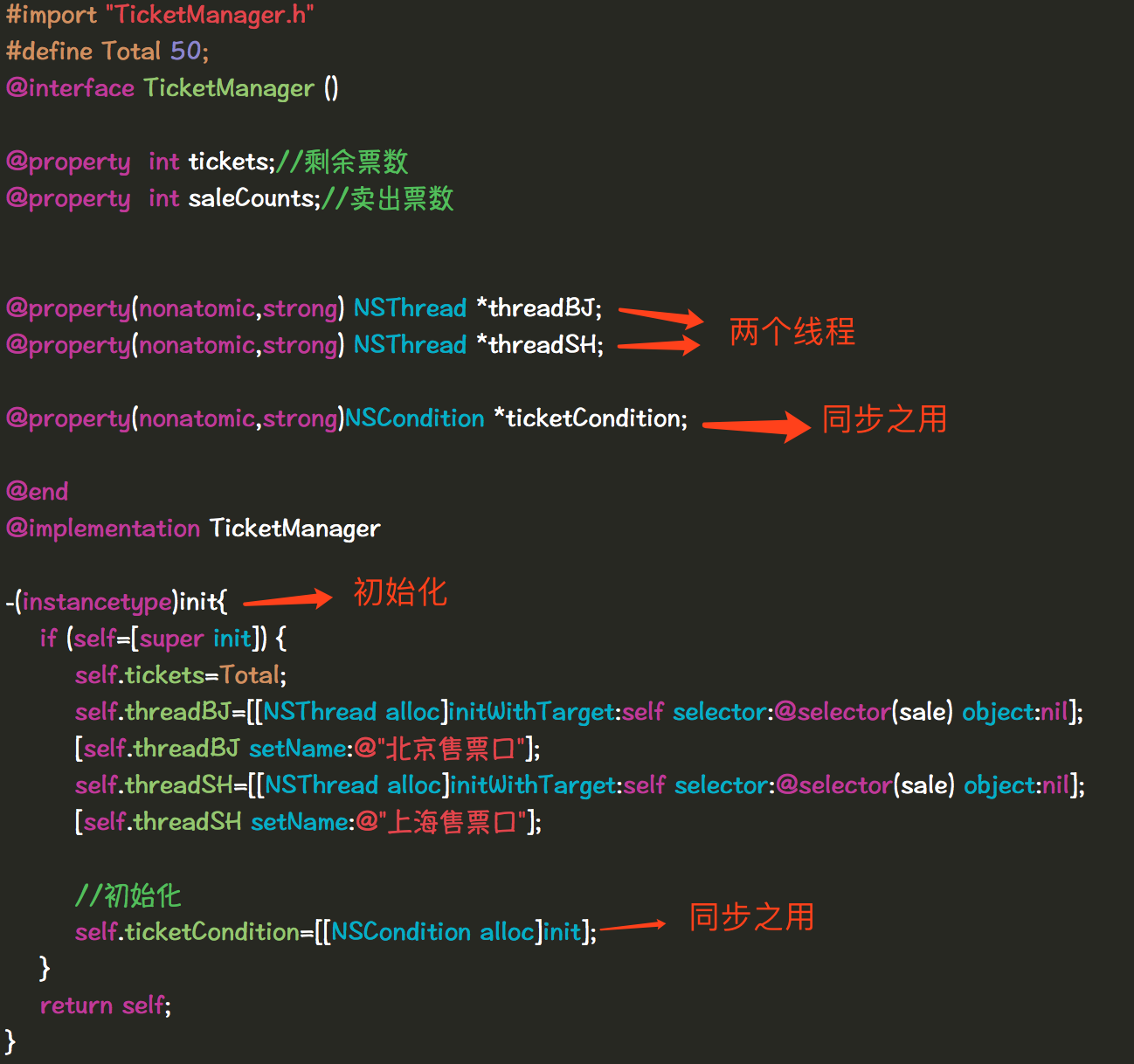
**三者比较：**

**第一种有自己的独立对象， 可以设置自己的属性方法。第二，三种没法实现。**

**例如：**

****

**举一个售票的例子来表示NSThread锁**

****

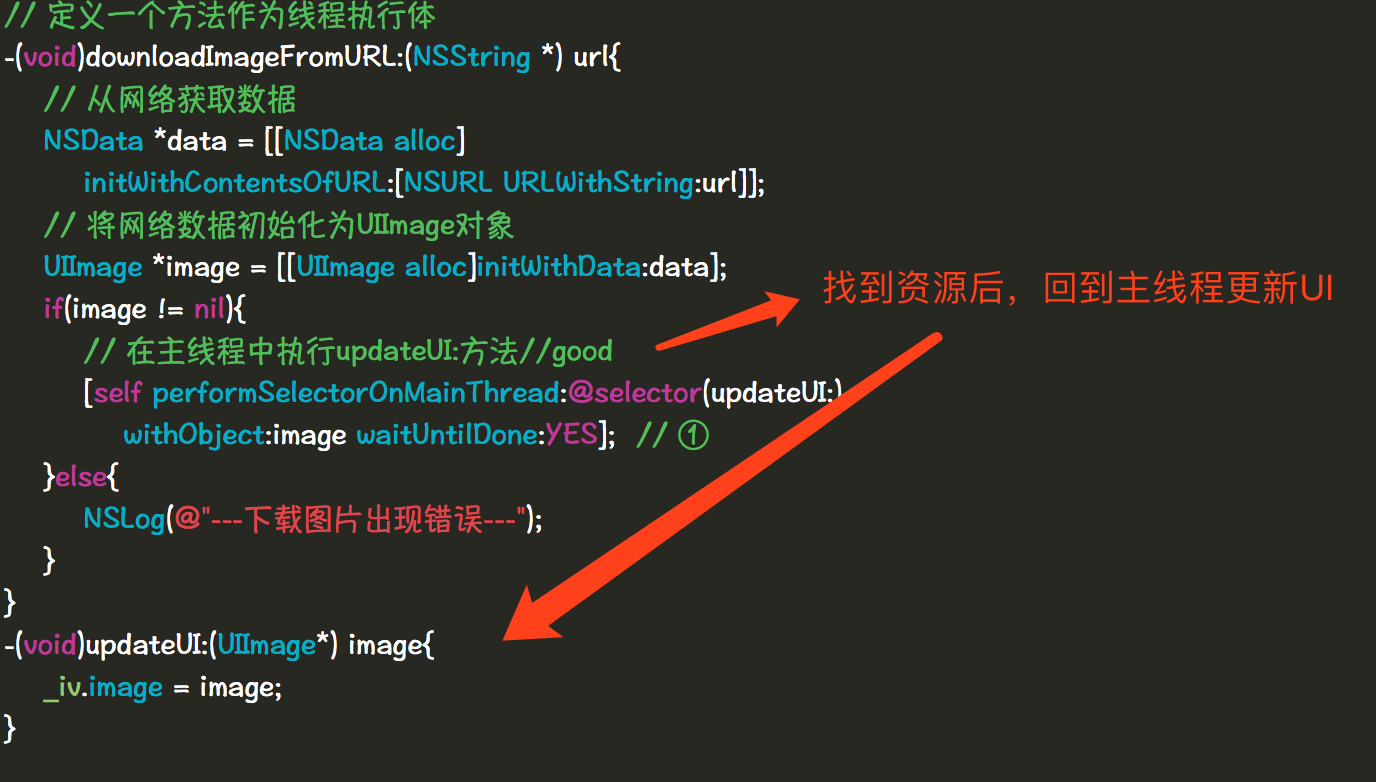
****

**NSThread在APP中的应用实例：**

**子线程中请求网络数据，主线程中更新UI。【故UI的刷新一定是在主线程中】**

**实例如下：**

****

****

**本例使用多线程从网络上下载图片，如果程序直接在UI线程中访问网络数据，由于网络的不确定性，当网络传输速度较慢时，UI线程就会被阻塞，从而导致应用失去响应。因此，程序将通过网络下载图片的操作放在多线程中完成。【注：iOS规定了只能在UI线程中修改UI控件的属性】。所以对于网络这块，最好的使用是【线程+异步】。**

**Pthread与NSThread相关链接：**[**http://www.jianshu.com/p/cbaeea5368b1**](http://www.jianshu.com/p/cbaeea5368b1)

1. **GCD**

**Grand Central Dispatch (GCD) 是Apple开发的一个多核编程的较新的解决方法。它主要用于优化应用程序以支持多核处理器以及其他对称多处理系统。【自动管理线程的生命周期：创建线程、调度任务、销毁线程】**

**了解概念**

**学习GCD之前，先来了解GCD中两个核心概念：任务和队列。**

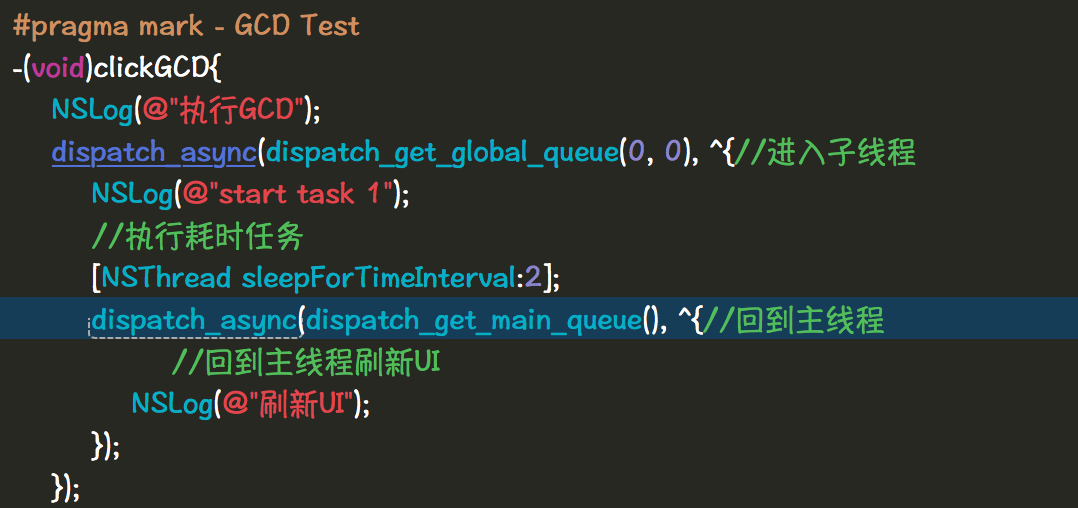
**任务：就是执行操作的意思，换句话说就是你在线程中执行的那段代码。在GCD中是放在block中的。执行任务有两种方式：同步执行&异步执行。两者的主要区别是：是否具备开启新线程的能力。**

* **同步执行（sync）：只能在当前线程中执行任务，不具备开启新线程的能力**
* **异步执行（async）：可以在新的线程中执行任务，具备开启新线程的能力**

**队列：这里的队列指任务队列，即用来存放任务的队列。队列是一种特殊的线性表，采用FIFO（先进先出）的原则，即新任务总是被插入到队列的末尾，而读取任务的时候总是从队列的头部开始读取。每读取一个任务，则从队列中释放一个任务。在GCD中有两种队列：串行队列和并行队列。**

* **并行队列（Concurrent Dispatch Queue）：可以让多个任务并行（同时）执行（自动开启多个线程同时执行任务）**
* **并行功能只有在异步（dispatch\_async）函数下才有效**
* **串行队列（Serial Dispatch Queue）：让任务一个接着一个地执行（一个任务执行完毕后，再执行下一个任务）**

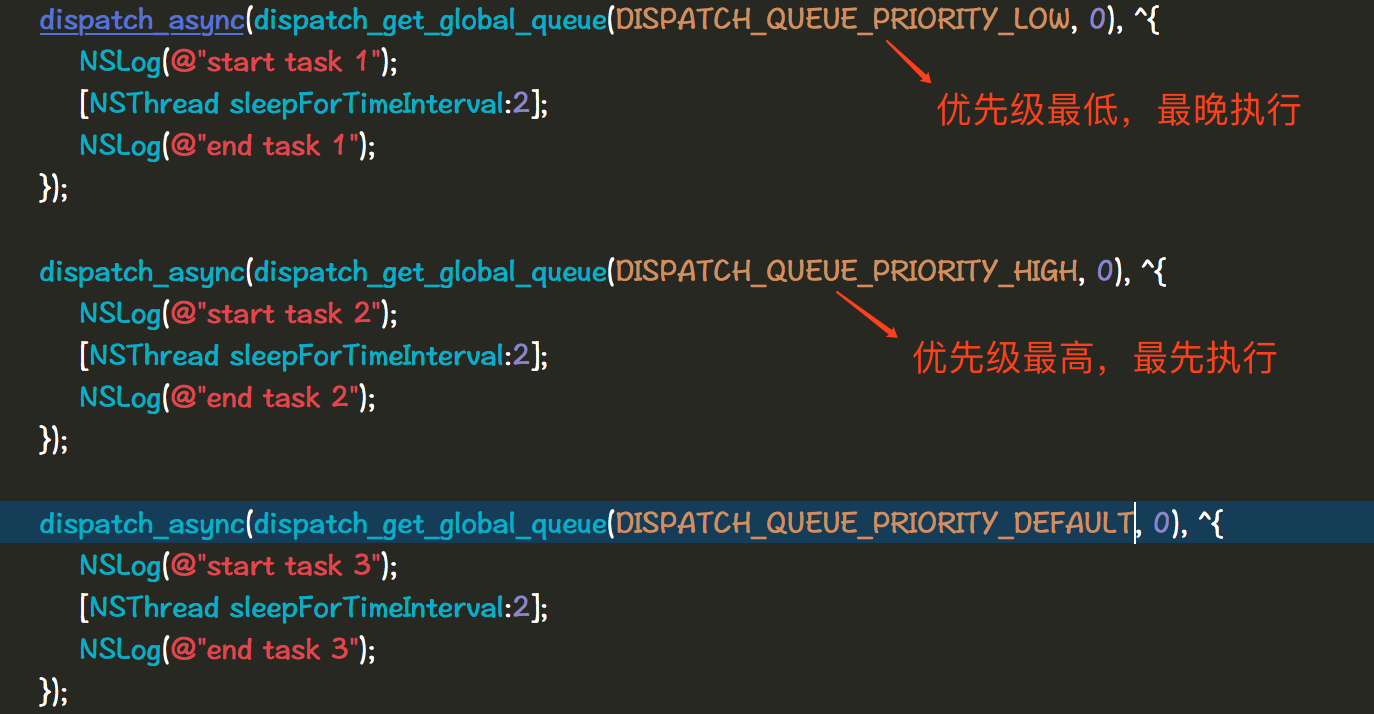
**例如：**

****

**dispatch\_get\_global\_queue（参数一,参数二）；**

**全局并发的queue**

**参数一：可以设置优先级；例如有三个线程，如图：**

****

**dispatch\_queue\_create(参数一【名称】，参数二【串行还是并行，NULL默认为串行，并行：】)**

**如图：**

****

**注意是只有一个线程。**

**若改为并行，如图：**

****

**注意是三个线程了，且并行执行。**

**dispatch\_group\_async（参数一，参数二，参数三）**

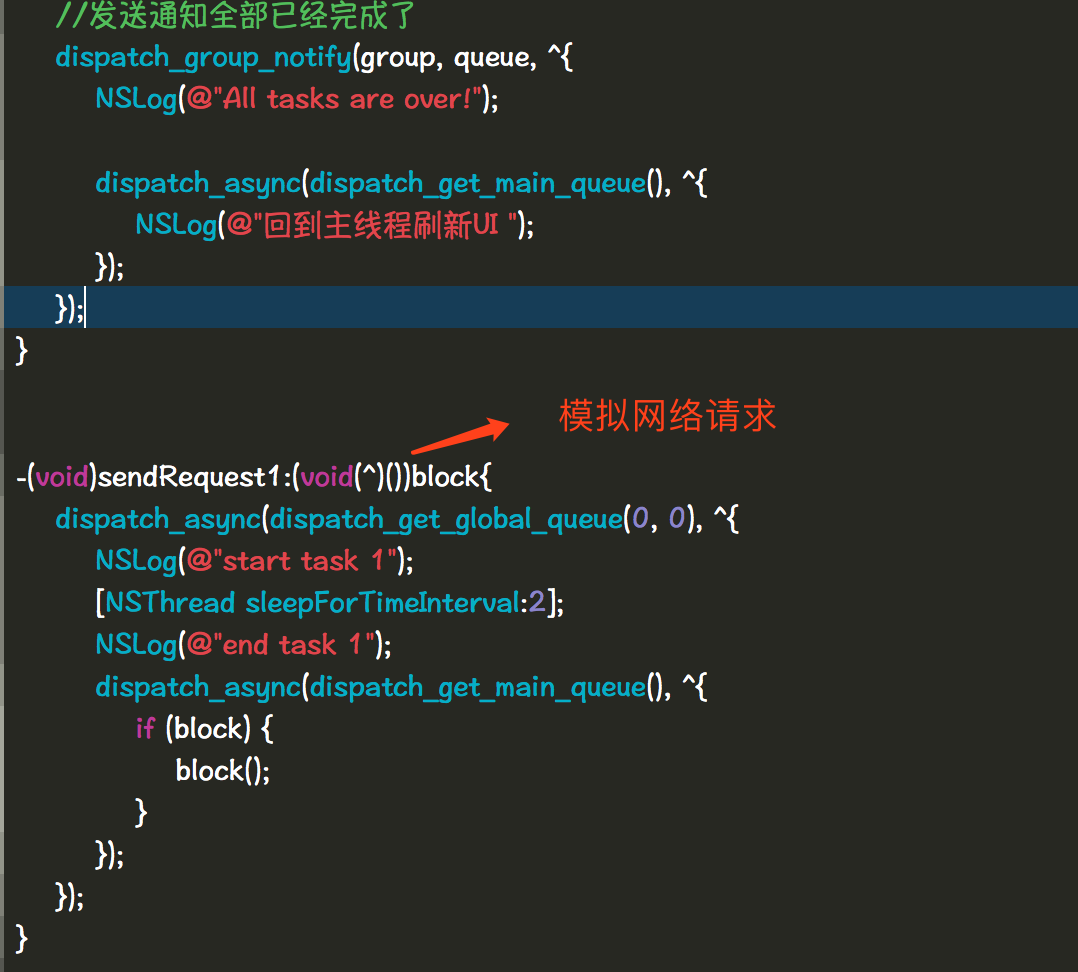
**如图：**

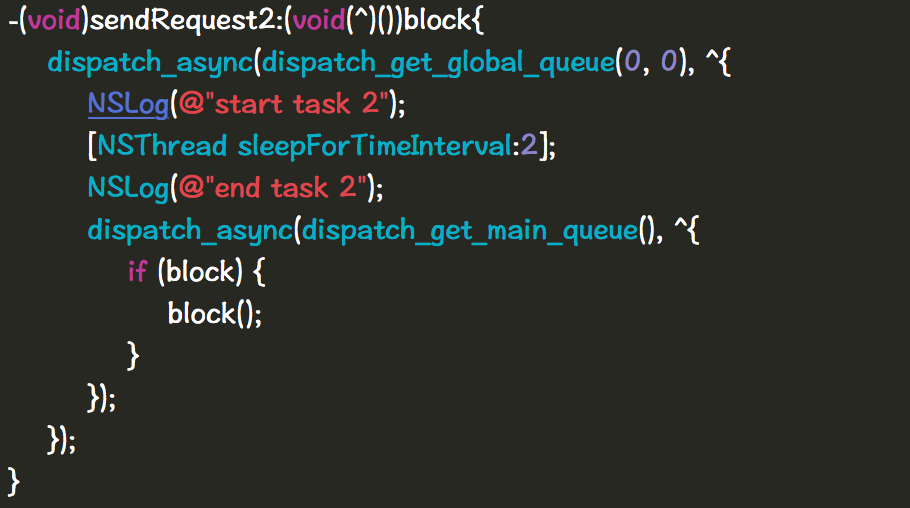
****

****

**如果我们有这样的一个需求，task1和task2分别都向后台异步请求数据，而且任务完成后返回主线程，并刷新UI操作，该如何在group中实现。**

****

****

****

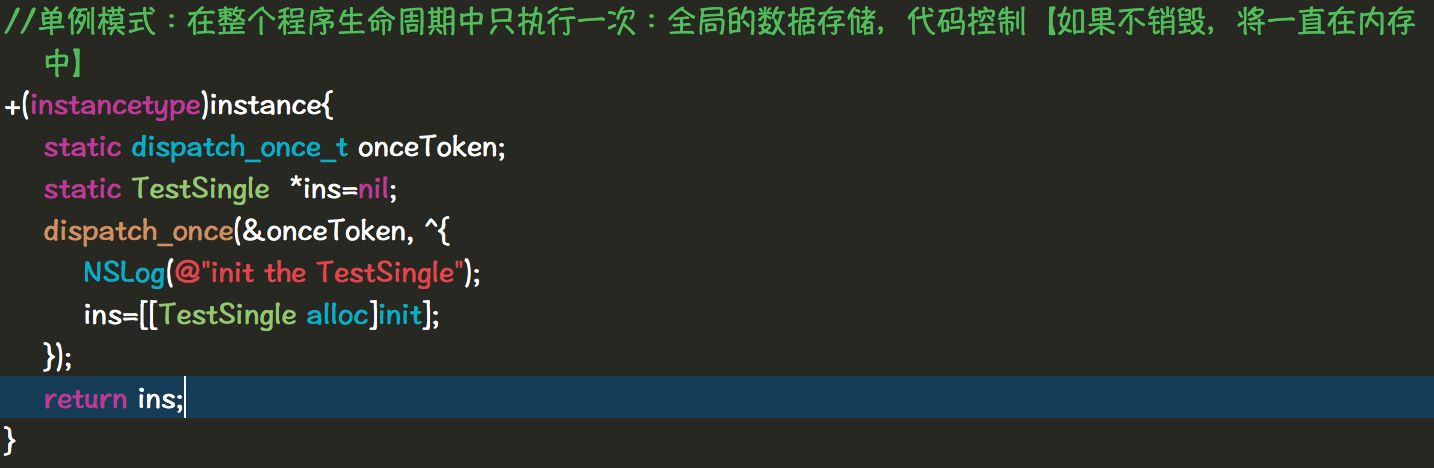
**GCD的一次性代码(只执行一次【单例】)dispatch\_once**

**新建一个类：**

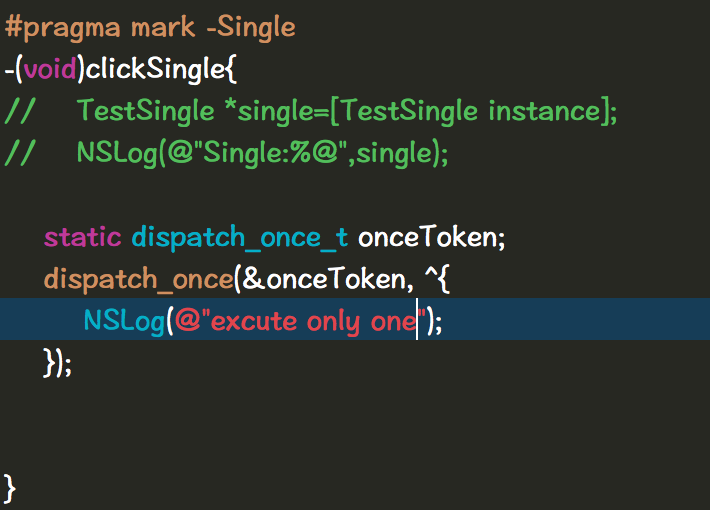
**.h文件**

****

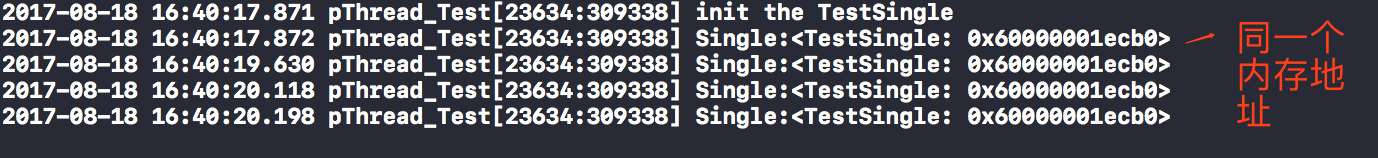
**.m文件**

****

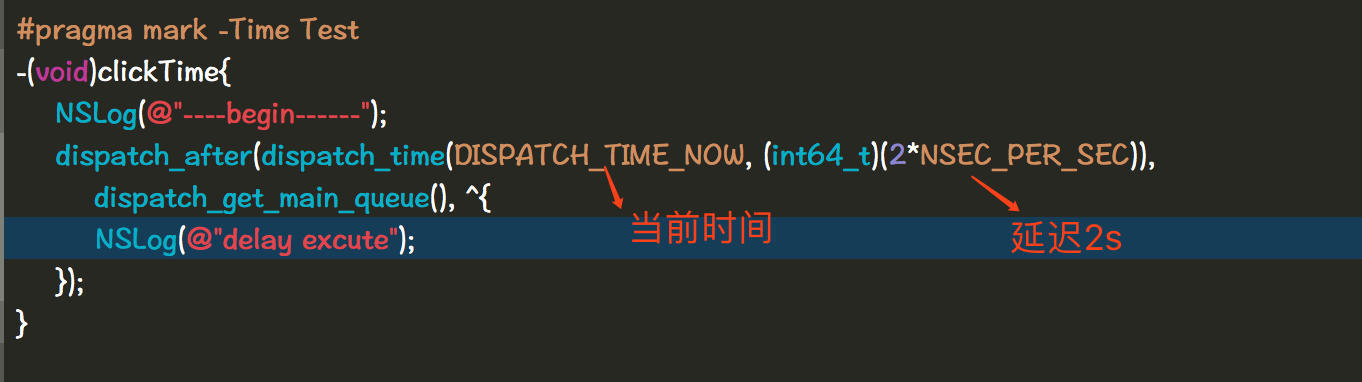
**使用：**

****

**你会发现，当点击按钮后，其生成的对象总归只有一个。**

****

**GCD的的延时执行方法dispatch\_after**

****

**GCD之间的通讯：**

**在iOS开发过程中，我们一般在主线程里边进行UI刷新，例如：点击、滚动、拖拽等事件。我们通常把一些耗时的操作放在其他线程，比如说图片下载、文件上传等耗时操作。而当我们有时候在其他线程完成了耗时操作时，需要回到主线程，那么就用到了线程之间的通讯。**

****

**GCD相关链接：**[**http://www.jianshu.com/p/2d57c72016c6**](http://www.jianshu.com/p/2d57c72016c6)

1. **NSOperation**

**NSOperation是苹果提供给我们的一套多线程解决方案。实际上NSOperation是基于GCD更高一层的封装，但是比GCD更简单易用、代码可读性也更高。**

**NSOperation需要配合NSOperationQueue来实现多线程。因为默认情况下，NSOperation单独使用时系统同步执行操作，并没有开辟新线程的能力，只有配合NSOperationQueue才能实现异步执行。可以将创建的NSOperation对象放在NSOperationQueue队列之中。**

**相关概念：**

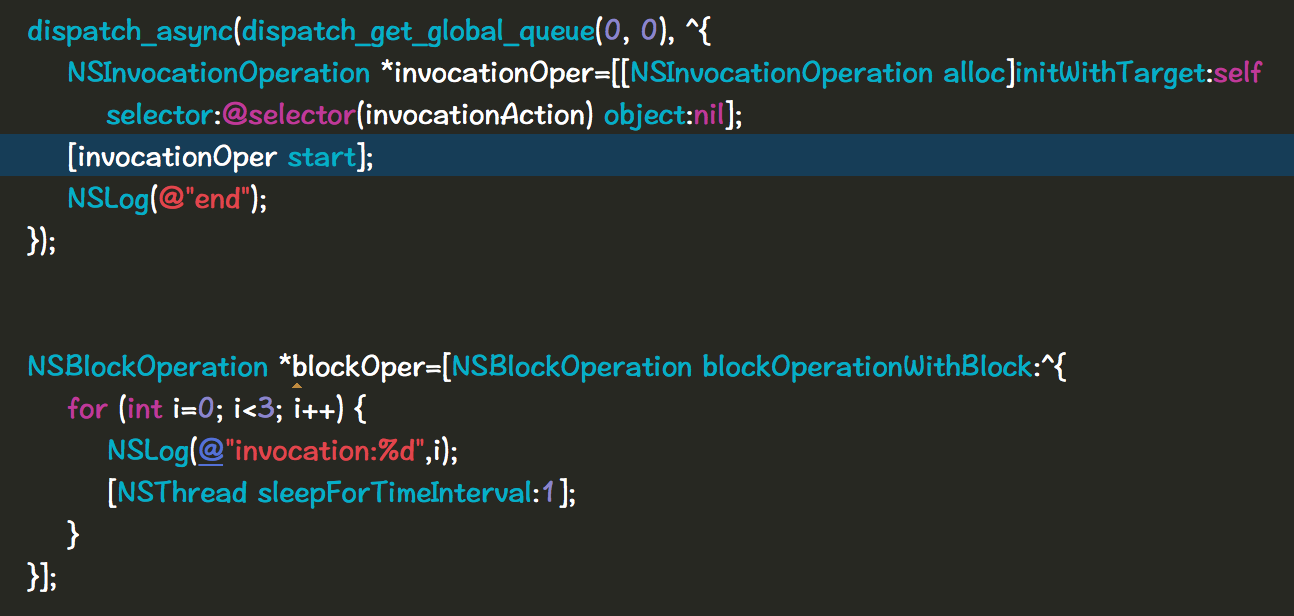
1. **NSOperationQueue**
2. **addOperation**
3. **setMaxConcurrentOperationCount**
4. **状态**

**ready、cancelled、executing、finished、asynchronous**

1. **依赖——addDependency**

**其中，系统分配给我们两个NSOperation的子类：NSInvocationOperation与NSBlockOperation**

**该两种方式的初始化方法：**

****

**自定义NSOperation的子类**

**CustomOperation.h：**

**-(instancetype)initWithName:(NSString \*)name;//名字属性**

**CustomOperation.m：**

**-(instancetype)initWithName:(NSString \*)name{**

**if (self=[super init]) {**

**self.operName=name;**

**}**

**return self;**

**}**

**-(void)main{**

**for (int i=0; i<3; i++) {**

**NSLog(@"%@  %d",self.operName,i);**

**[NSThread sleepForTimeInterval:1];**

**}}**

**然后使用时，导入该头文件。**

**创建一个属性：**

****

****

**NSOperation相关链接：**[**http://www.jianshu.com/p/4b1d77054b35**](http://www.jianshu.com/p/4b1d77054b35)