

进程 (Process): 系统中正在运行的应用程序

进程特性:

- 每个进程间独立存在
- 每个进程运行在其专用且受保护的内存空间内

内存

Xcode进程

Chrome进程

Keynote进程

Safari进程

QQ音乐进程

.....

线程 (Thread): 程序的一段执行序列, 是进程的基本执行单元

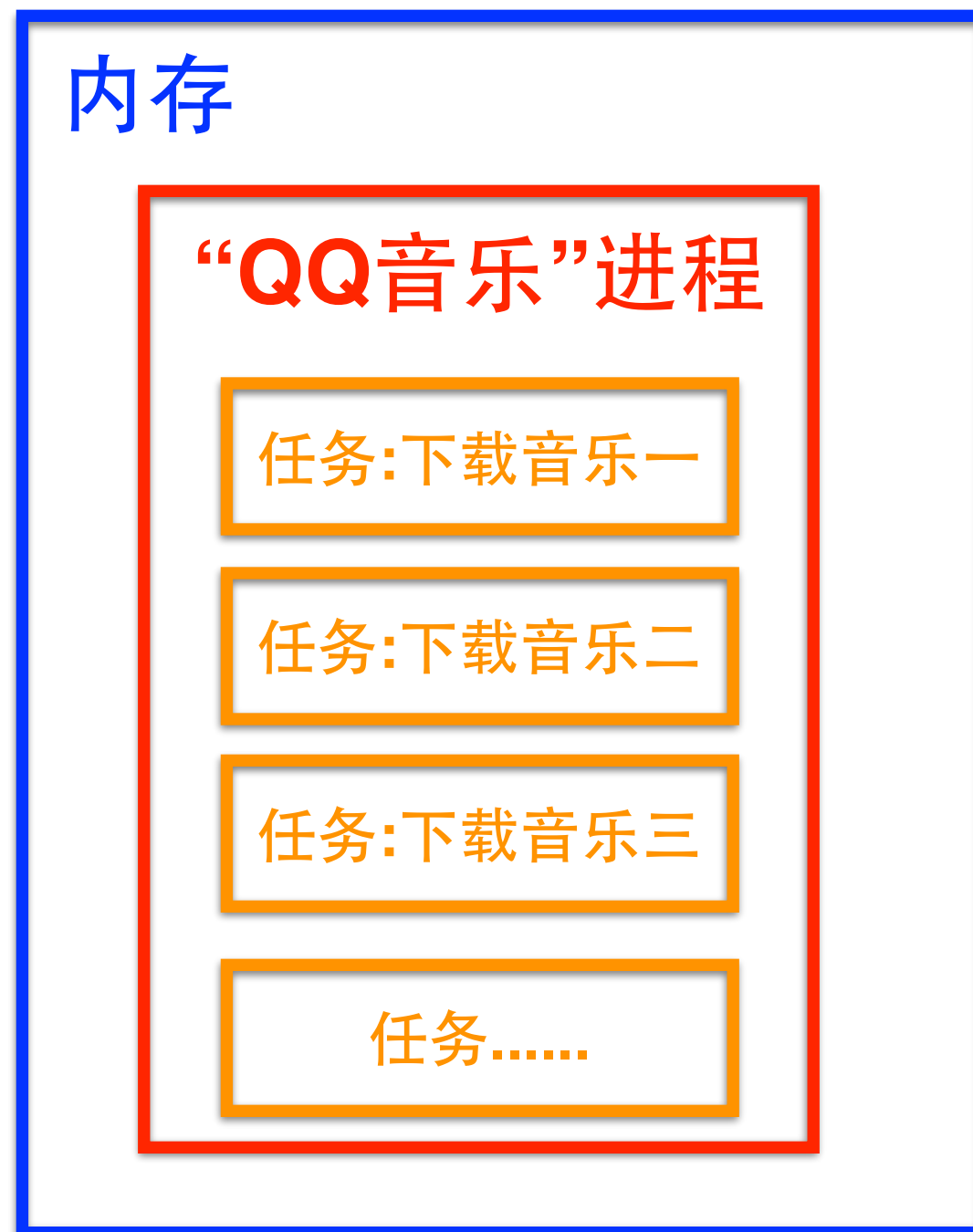
进程/线程关系:

- 进程本身不会执行任务, 只有线程有执行任务的功能



情况一： 进程只有一个线程 (主线程)

——> 所有该应用程序对应的任务只能一个一个顺序执行



情况二：进程有多线程

一个进程中可以同时运行多条线程，每条线程同时(并行)执行不同的任务

为什么需要多线程？

- 适当提高程序的执行效率
- 适当提高资源利用率 (cpu、内存利用率)



内存

“QQ音乐”进程

任务:下载音乐一

任务:下载音乐二

任务.....



iOS中的多线程

主线程： 程序运行后，默认开启一条线程，称为“主线程”

iOS主线程作用：

- 显示/刷新UI界面
- 处理UI事件(比如点击事件、滚动事件和拖拽事件等)
- 将耗时的任务 (例如:网络图片、视频、音乐等资源的下载) 放在其他线程中执行

iOS多线程技术

1.Pthread

一套通用的多线程API; 难度大; 几乎不用

2.NSThread

直接操作线程; 偶尔使用

3.GCD(Grand Central Dispatch)

替代NSThread; 利用设备多核; 经常使用

4.NSOperation

面向对象; GCD上的OC接口; 经常使用

样例一：

目标：

不使用线程 ——> 主线程阻塞

样例描述：

点击界面打印多条log ——> **UITextView**无法滚动

样例二：

目标：

使用**PThread** ——> 不会阻塞主线程

样例描述：

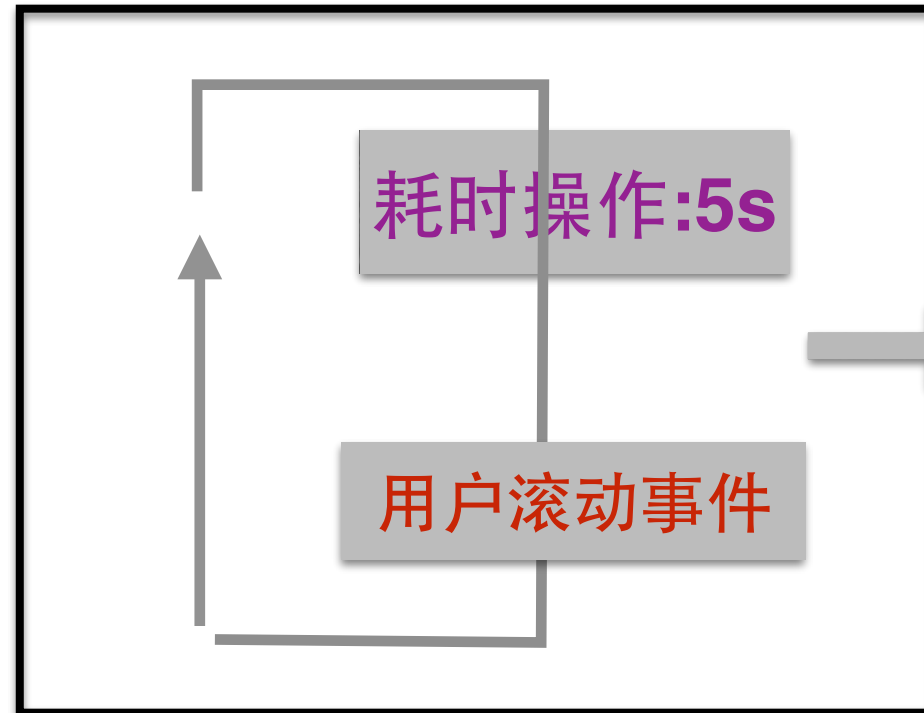
创建**PThread**线程对象 ——> 点击界面打印多条log
——> **UITextView**滚动自如

步骤：

1. 声明**pthread**对象
2. 创建线程对象**pthread_create**
3. 执行线程任务

主线程

不使用子线程



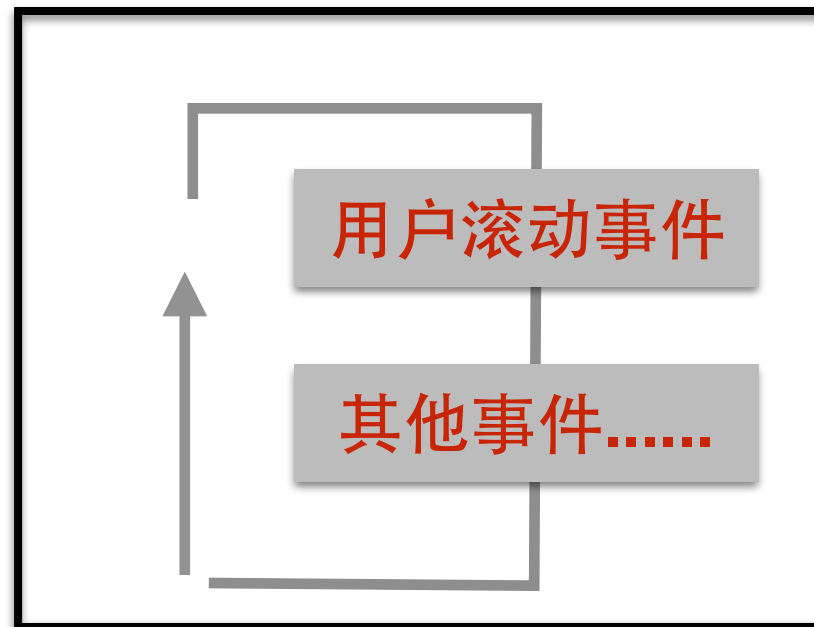
阻塞主线程的执行

使用子线程



主线程

子线程



+



不妨碍主线程执行任务

1. PThread
2. NSThread

PThread

1. pthread_t pthread;
2. pthread_create(&pthread, NULL, task, NULL);
3. void *task(void *data) {
 //执行耗时操作
}

NSThread: 创建/启动线程

```
1. NSThread *thread = [[NSThread alloc] initWithTarget:self  
    selector:(task) object:nil];  
2. thread.name = @"线程名字";  
3. [thread start];  
4. - (void)task {  
    //耗时操作  
}
```

其他用法

1. 获得当前线程

```
NSThread *current = [NSThread currentThread];
```

2. 线程的调度优先级: 调度优先级的取值范围是0.0 ~ 1.0, 默认0.5, 值越大, 优先级越高

```
+ (double)threadPriority;
```

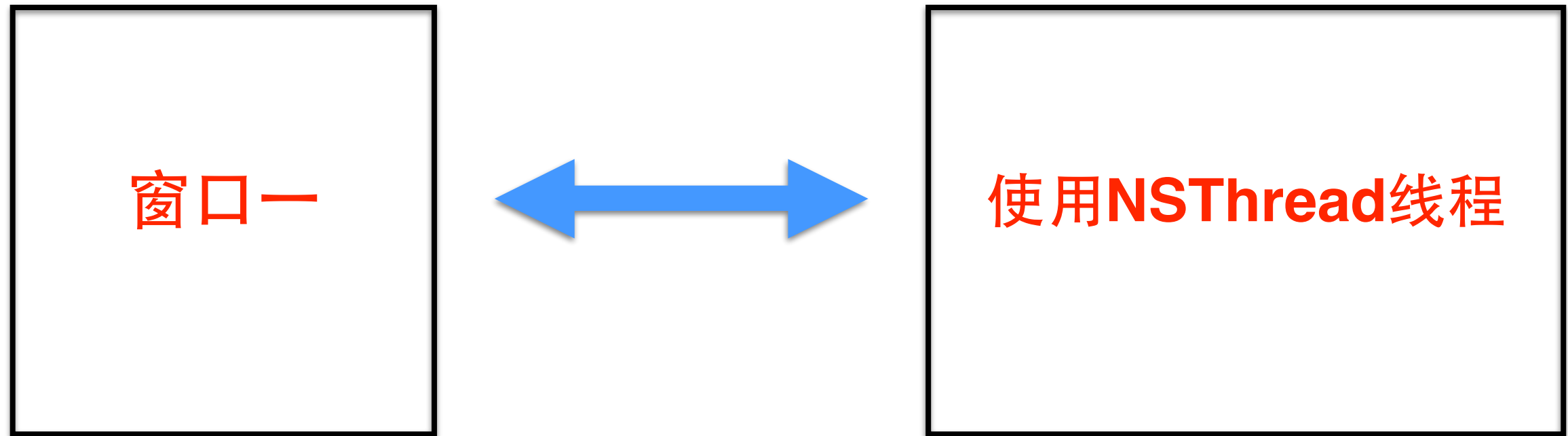
```
+ (BOOL)setThreadPriority:(double)p;
```

3. 设置线程的名字

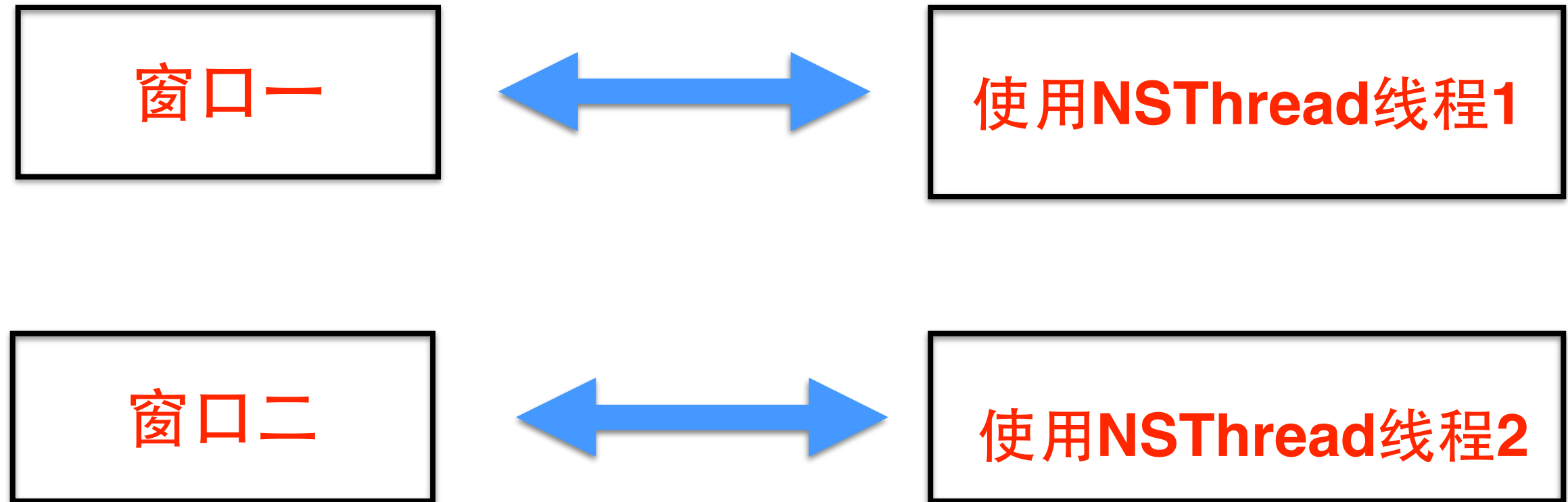
```
- (void)setName:(NSString *)n;
```

```
- (NSString *)name;
```

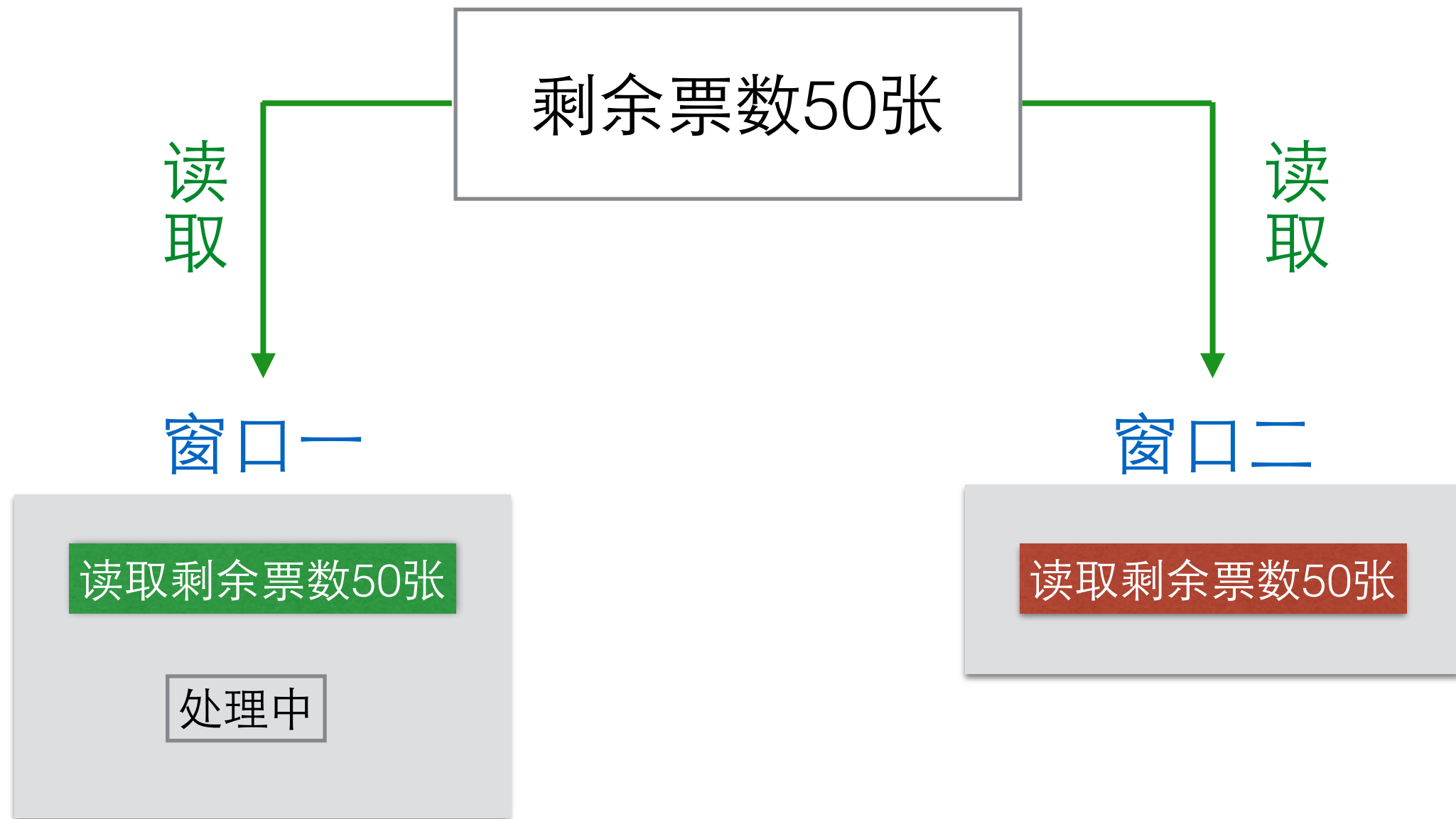
使用NSThread模拟一个窗口的卖票样例一



使用NSThread模拟两个窗口的卖票样例二

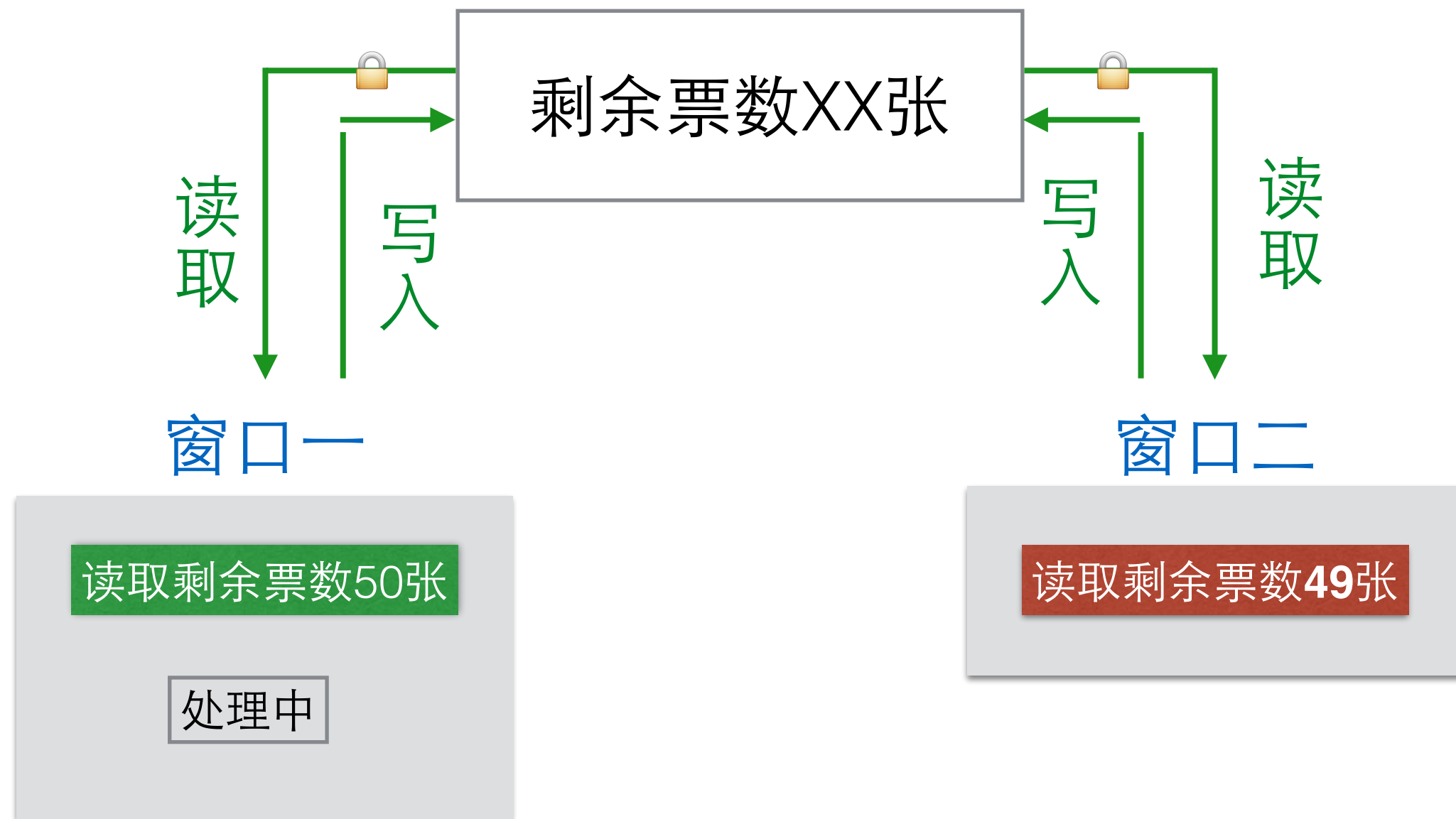


两个窗口同时卖票的流程



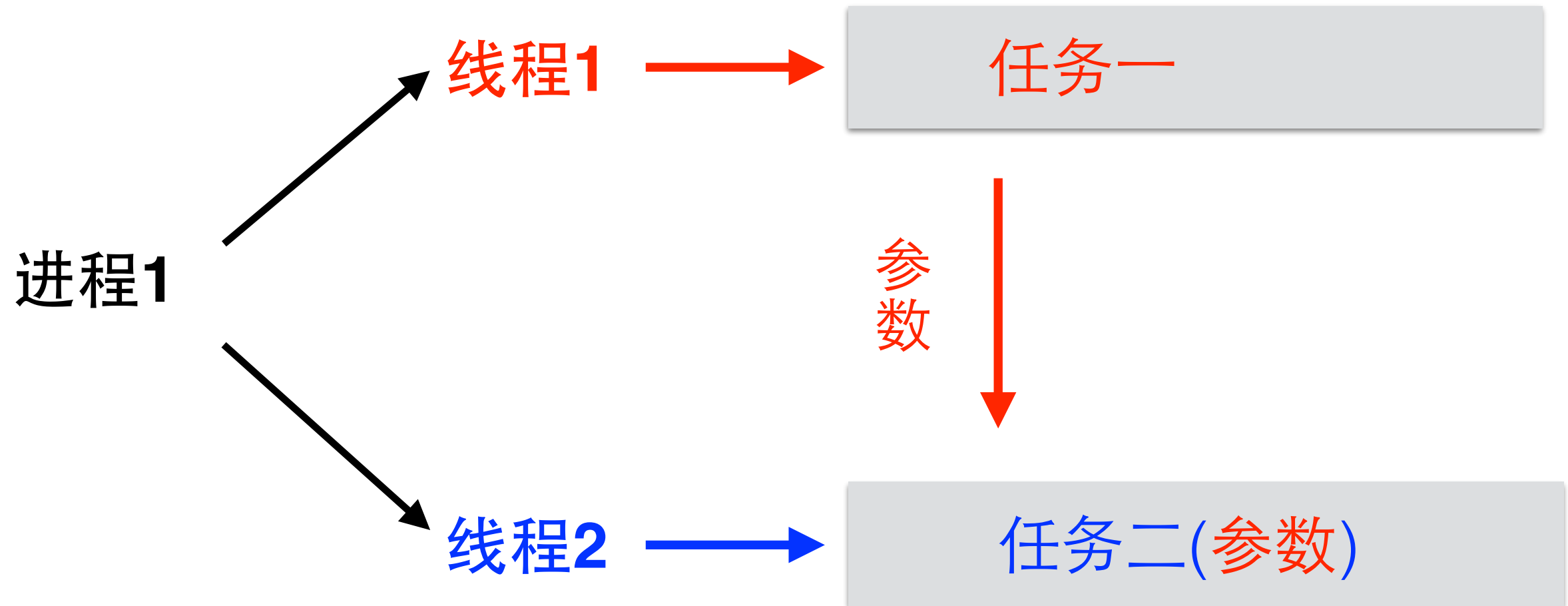
结论：多个线程同时访问同一个数据，会造成数据不一致问题

两个窗口使用加锁/解锁方式卖票的流程

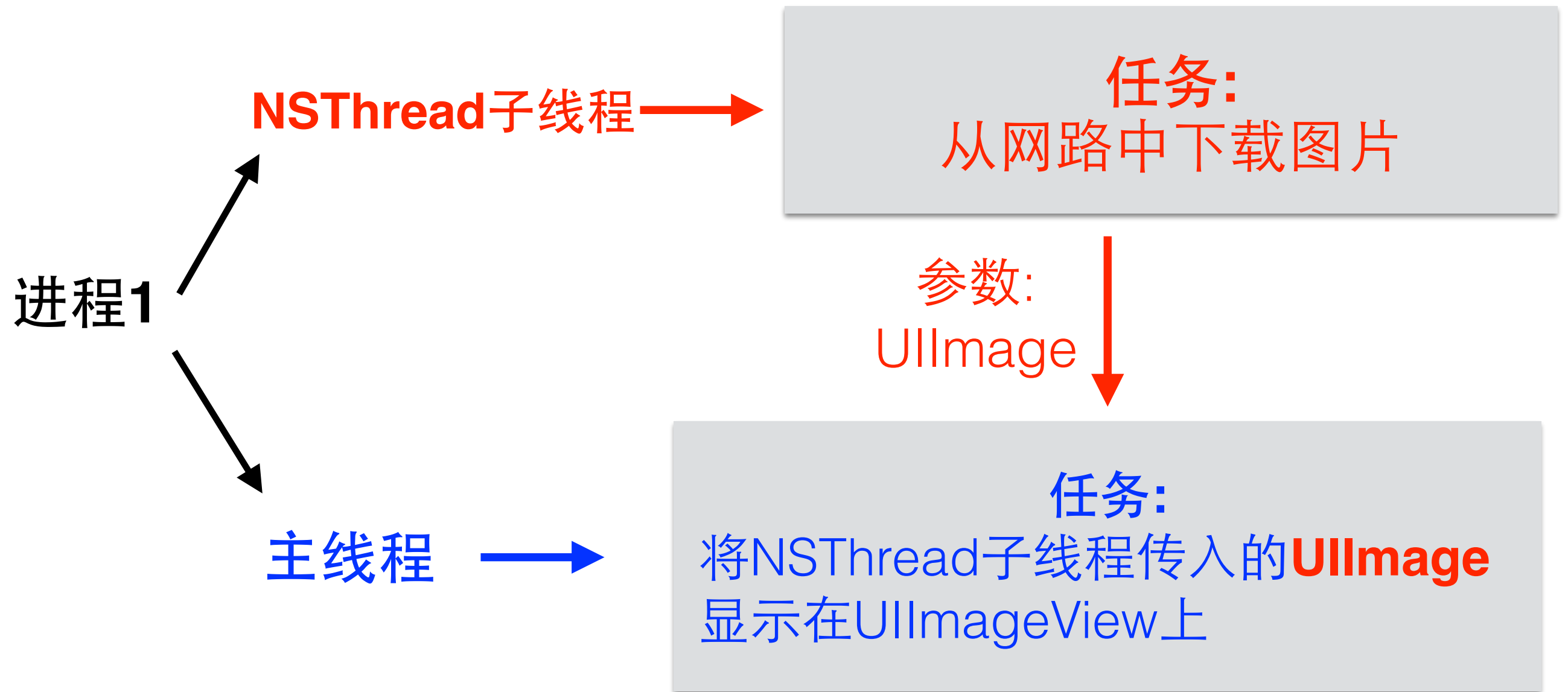


结论： 只用使用了加锁/解锁的机制，才可以保证数据一致。

线程间通讯



子线程和主线程间通讯



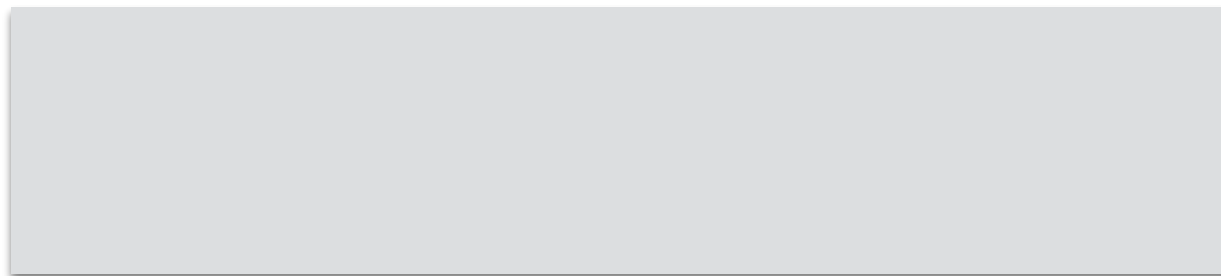
从子线程回到主线程方式:

1. 使用NSObject的方法: `performSelectorOnMainThread`
2. 使用NSObject的方法:
`performSelector:onThread:[NSThread mainThread]`

GCD 一般工作原理

1. 创建空队列

空dispatch queue



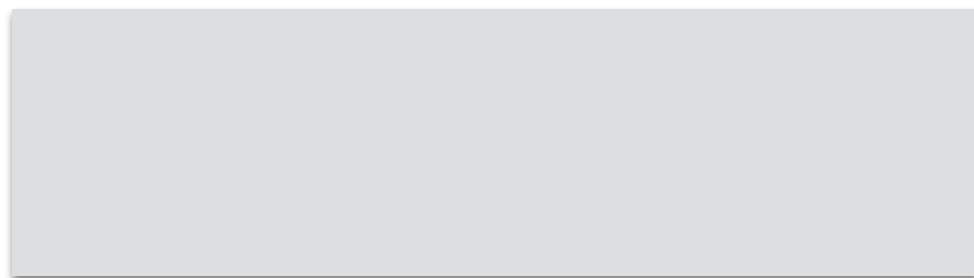
2. 把任务添加队列中

dispatch queue

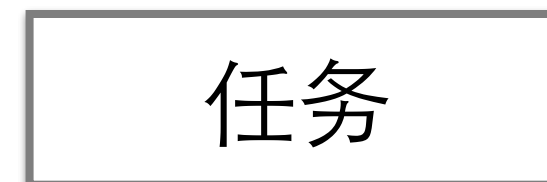


3. 执行队列中的任务

执行



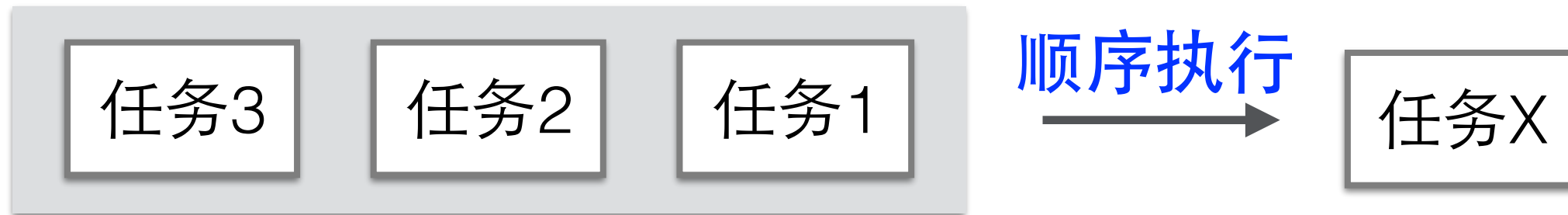
处理



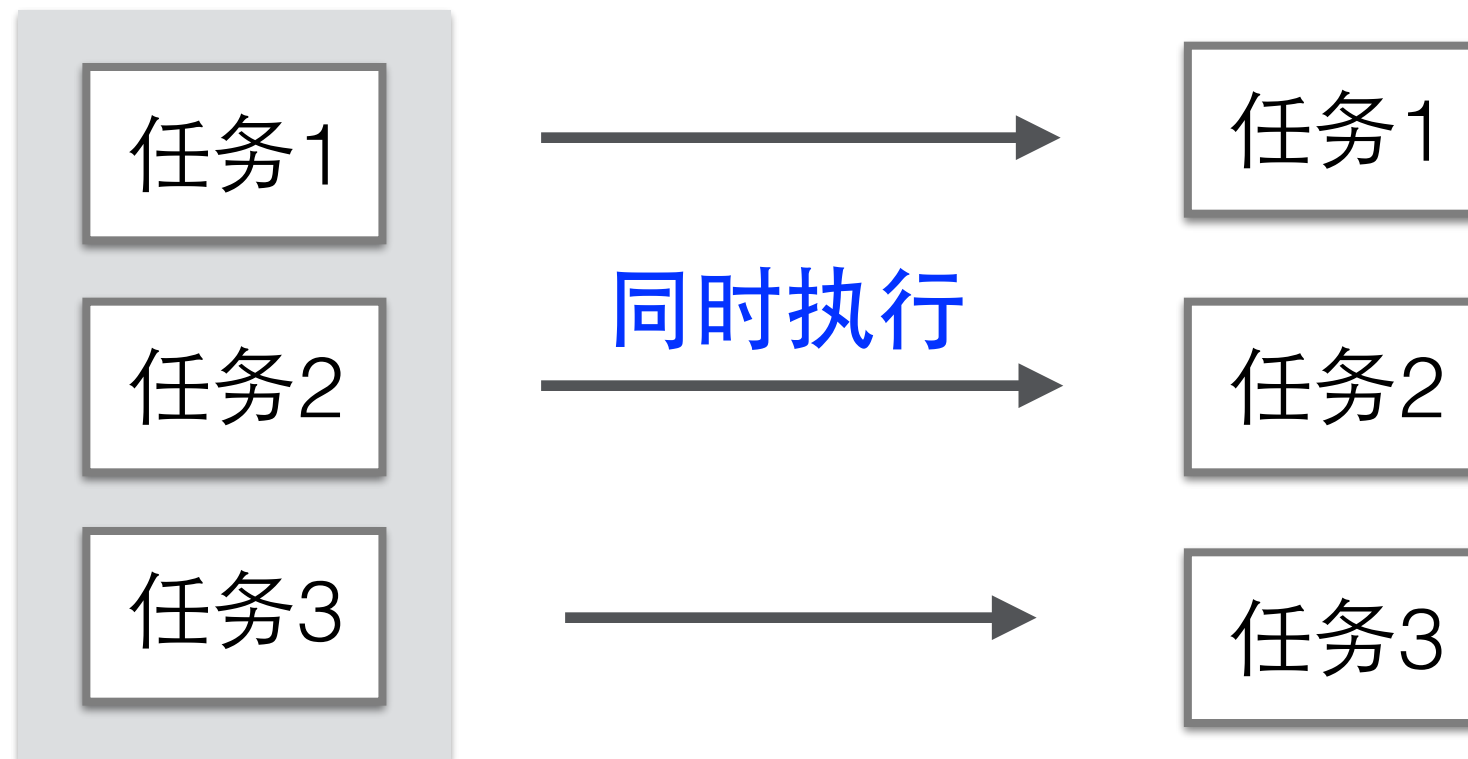
GCD 常用术语一

GCD队列类型:

1. 串行队列(Serial Dispatch Queue): 一个一个顺序地执行



2. 并行队列(Concurrent Dispatch Queue): 同时执行多个任务



GCD 常用术语二

执行任务方式：

1. 同步执行(**Synchronous Dispatch**): 队列中的任务在**当前线程**中执行
(不会启动新的线程)
2. 异步执行(**Asynchronous Dispatch**): 队列中的任务在**子线程**中执行
(启动新的线程)

GCD 队列类型和执行方式

队列类型 执行方式	串行队列	并行队列
同步执行	串行同步	并行同步
异步执行	串行异步	并行异步

GCD 四种组合

结论：

- 1.串行同步：队列中的任务**顺序**执行；在**当前**线程中执行；
- 2.串行异步：队列中的任务**顺序**执行；在**子**线程中执行；主线程继续执行，不会等待子线程执行完毕
- 3.并行同步(一般不用)：队列中的任务**顺序**执行；在**当前**线程中执行；
- 4.并行异步：队列中的任务**同时**执行；在**子**线程中执行；主线程继续执行，不会等待子线程执行完毕

GCD 中两种系统提供的队列

全局队列(Global Queue): 是全局的并行队列

主队列(Main Queue): 是在主线程中执行的队列(串行)

队列类型 执行方式	串行队列	并行队列/全局队列	主队列
同步执行	串行同步	并行同步	主队列同步
异步执行	串行异步	并行异步	主队列异步