多线程概念:

进程:正在进行的程序被称为进程,负责程序运行的内存分配,每个进程都有自己的独立虚拟内存空间.

**线程**

线程是进程中一个独立的执行路径(控制单元)

一个进程中至少包含一条线程，即主线程

可以将耗时的执行路径(如：网络请求)放在其他线程中执行

创建线程的目的就是为了开启一条新的执行路径，运行指定的代码，与主线程中的代码实现同时运行

**iOS的三种多线程技术**

**NSThread**

* 使用NSThread对象建立一个线程非常方便
* 但是！要使用NSThread管理多个线程非常困难，不推荐使用
* 技巧！使用[NSThread currentThread]跟踪任务所在线程，适用于这三种技术
* **NSOperation/NSOperationQueue**是使用GCD实现的一套Objective-C的API是面向对象的线程技术
* 提供了一些在GCD中不容易实现的特性，如：限制最大并发数量、操作之间的依赖关系

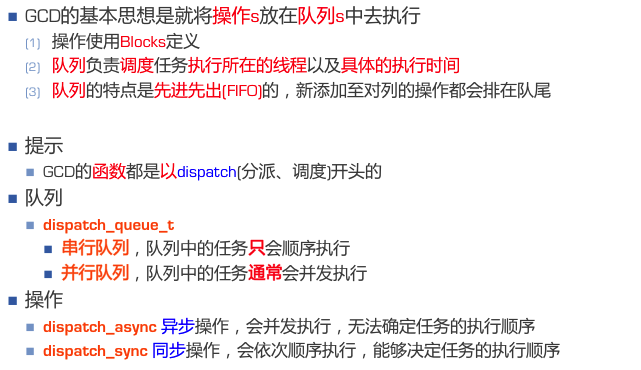
**GCD —— Grand Central Dispatch**

(1)是基于C语言的底层API

(1)用Block定义任务，使用起来非常灵活便捷

(1)提供了更多的控制能力以及操作队列中所不能使用的底层函数

**提示：**iOS的开发者，需要了解三种多线程技术的基本使用，因为在实际开发中会根据实际情况选择不同的多线程技术



串行队列:

#pragma mark  - 串行队列 同步任务和异步任务

- (void) GcdTest{

    //串行队列按顺序执行队列中的任务

    //新建一个串行队列

    dispatch\_queue\_t queue = dispatch\_queue\_create("serialQueue", DISPATCH\_QUEUE\_SERIAL);

    //调用同步方法

    for (int i=0; i<10; i++) {

        dispatch\_sync(queue, ^{

            NSLog(@"srialQueue--sync--%@----%d",[NSThread currentThread],i);

            //同步任务中包含同步会造成线程阻塞

//            dispatch\_sync(queue, ^{

//                NSLog(@"srialQueue--sync111--%@",[NSThread currentThread]);

//

//            });

        });

    }

    //调用异步方法

    for (int i = 0; i<10; i++) {

        dispatch\_async(queue, ^{

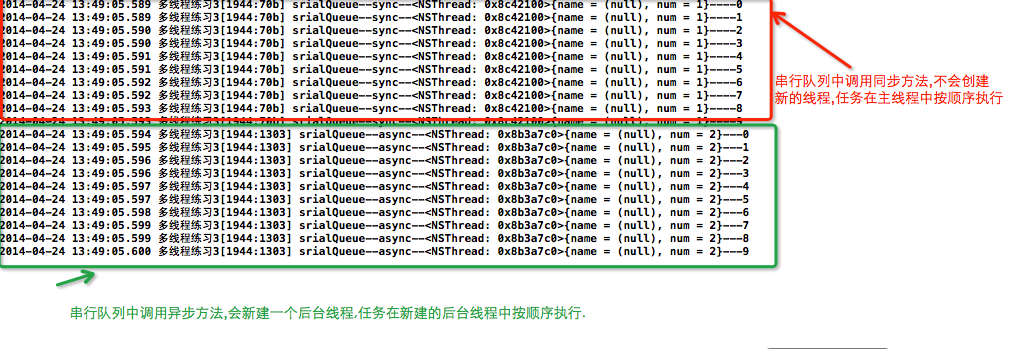
            NSLog(@"srialQueue--async--%@---%d",[NSThread currentThread],i);

        });

    }

}

执行详解:



总结:

同步操作不会新建线程、操作顺序执行（没用！）同步任务中包含同步会造成线程阻塞

异步操作会新建线程、操作顺序执行（非常有用！）

场景：既不影响主线程，又需要顺序执行的操作！