iOS GCD中级篇 - dispatch_semaphore（信号量）的理解及使用

理解这个概念之前，先抛出一个问题

问题描述：

假设现在系统有两个空闲资源可以被利用，但同一时间却有三个线程要进行访问，这种情况下，该如何处理呢？

或者

我们要下载很多图片，并发异步进行，每个下载都会开辟一个新线程，可是我们又担心太多线程肯定cpu吃不消，那么我们这里也可以用信号量控制一下最大开辟线程数。

定义：

1、信号量：就是一种可用来控制访问资源的数量的标识，设定了一个信号量，在线程访问之前，加上信号量的处理，则可告知系统按照我们指定的信号量数量来执行多个线程。

其实，这有点类似锁机制了，只不过信号量都是系统帮助我们处理了，我们只需要在执行线程之前，设定一个信号量值，并且在使用时，加上信号量处理方法就行了。

2、信号量主要有3个函数，分别是：

```
1
2
3 dispatch_semaphore_create（信号量值）
4 dispatch_semaphore_wait（信号量，等待时间）
5 dispatch_semaphore_signal(信号量)
```

注意，正常的使用顺序是**先降低**然后**再提高**，这两个函数通常成对使用。　（具体可参考下面的代码示例）

3、那么就开头提的问题，我们用代码来解决

```
-(void)dispatchSignal{

    dispatch_semaphore_t semaphore = dispatch_semaphore_create(2);
    dispatch_queue_t quene = dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT, 0);

    dispatch_async(quene, ^{
        dispatch_semaphore_wait(semaphore, DISPATCH_TIME_FOREVER);
        NSLog(@"run task 1");
        sleep(1);
        NSLog(@"complete task 1");
        dispatch_semaphore_signal(semaphore);
    });<br>

    dispatch_async(quene, ^{
        dispatch_semaphore_wait(semaphore, DISPATCH_TIME_FOREVER);
        NSLog(@"run task 2");
        sleep(1);
        NSLog(@"complete task 2");
```

```
        dispatch_semaphore_signal(semaphore);
    });<br>


    dispatch_async(quene, ^{
        dispatch_semaphore_wait(semaphore, DISPATCH_TIME_FOREVER);
        NSLog(@"run task 3");
        sleep(1);
        NSLog(@"complete task 3");
        dispatch_semaphore_signal(semaphore);
    });
}


```

执行结果：

```
2017-01-11 16:42:29.904 TestOC[23473:1138509] run task 1
2017-01-11 16:42:29.904 TestOC[23473:1138522] run task 2
2017-01-11 16:42:30.977 TestOC[23473:1138522] complete task 2
2017-01-11 16:42:30.977 TestOC[23473:1138509] complete task 1
2017-01-11 16:42:30.978 TestOC[23473:1138516] run task 3
2017-01-11 16:42:32.051 TestOC[23473:1138516] complete task 3
```

总结：由于设定的信号值为2，先执行两个线程，等执行完一个，才会继续执行下一个，保证同一时间执行的线程数不超过2。

这里我们扩展一下，假设我们设定信号值=1

```
dispatch_semaphore_create(1)<br><br>
```

那么结果就是：

```
2017-01-11 16:41:00.074 TestOC[23411:1136853] run task 1
2017-01-11 16:41:01.148 TestOC[23411:1136853] complete task 1
2017-01-11 16:41:01.149 TestOC[23411:1136861] run task 2
2017-01-11 16:41:02.224 TestOC[23411:1136861] complete task 2
2017-01-11 16:41:02.224 TestOC[23411:1136860] run task 3
2017-01-11 16:41:03.298 TestOC[23411:1136860] complete task 3
```

如果设定信号值=3

```
1 dispatch_semaphore_create(3)<br><br>
```

那么结果就是：

```
2017-01-11 16:42:58.571 TestOC[23507:1139337] run task 3
2017-01-11 16:42:58.571 TestOC[23507:1139330] run task 2
2017-01-11 16:42:58.571 TestOC[23507:1139324] run task 1
2017-01-11 16:42:59.644 TestOC[23507:1139337] complete task 3
2017-01-11 16:42:59.644 TestOC[23507:1139324] complete task 1
2017-01-11 16:42:59.644 TestOC[23507:1139330] complete task 2
```

其实设定为3，就是不限制线程执行了，因为一共才只有3个线程。

以上只是举的比较简单的例子，在一些特殊场景下，合理利用信号量去控制，能够方便的解决我们的难题哦