**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：

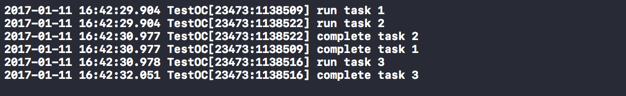
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | //创建信号量，参数：信号量的初值，如果小于0则会返回NULL  dispatch\_semaphore\_create（信号量值）    //等待降低信号量  dispatch\_semaphore\_wait（信号量，等待时间）    //提高信号量  dispatch\_semaphore\_signal(信号量) |

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

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30 | -(void)dispatchSignal{      //crate的value表示，最多几个资源可访问      dispatch\_semaphore\_t semaphore = dispatch\_semaphore\_create(2);      dispatch\_queue\_t quene = dispatch\_get\_global\_queue(DISPATCH\_QUEUE\_PRIORITY\_DEFAULT, 0);        //任务1      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>      //任务2      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>      //任务3      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：

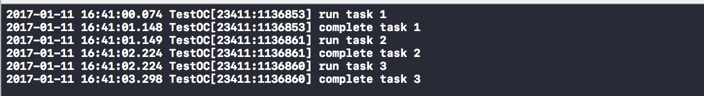


总结：由于设定的信号值为2，先执行两个线程，等执行完一个，才会继续执行下一个，保证同一时间执行的线程数不超过2。

这里我们扩展一下，假设我们设定信号值=1

|  |  |
| --- | --- |
| 1 | dispatch\_semaphore\_create(1)<br><br> |

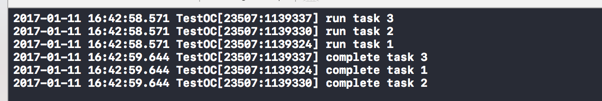
那么结果就是：



如果设定信号值=3

|  |  |
| --- | --- |
| 1 | dispatch\_semaphore\_create(3)<br><br> |

那么结果就是：



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。