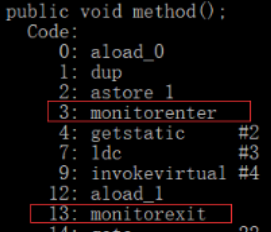
# 实验5 同步互斥

练习1: 了解信号量和管程的实现机制

1. 同步互斥的底层支持是如何实现的？



两条指令组成：

1. monitorenter:

每个对象都有一个监视器锁(monitor)，当monitor被占用时就会处于锁定状态。线程执行monitorenter命令获取monitor锁的过程如下：

1、如果monitor的进入数为0，则线程获取锁，并设置monitor的进入数为1

2、如果该线程已经占有该monitor，则进入数+1

3、如果其他线程占有该monitor，则monitor的进入数不为0，则该线程进入阻塞状态，直到monitor为0，重新获取monitor的所有权

②monitorexit:   
执行monitorexit的线程必须是monitor的所有者。   
当执行该命令时，monitor的进入数-1，当monitor的进入数为0，该线程已经不再是该monitor的所有者，其他被这个monitor阻塞的线程可以尝试获取monitor的所有权。即同步底层是通过一个monitor的对象来完成。

1. 对比原理课上学到的信号量和p，v操作，说明Ucore中信号量机制的实现。

 \_\_up: 对应到了原理课中提及到的V操作，表示释放了一个该信号量对应的资源，如果有等待在了这个信号量上的进程，则将其唤醒执行；结合函数的具体实现可以看到其采用了禁用中断的方式来保证操作的原子性，函数中操作的具体流程为：

* 查询等待队列是否为空，如果是空的话，给整型变量加1；
* 如果等待队列非空，取出其中的一个进程唤醒；

 \_\_down: 同样对应到了原理课中提及的P操作，表示请求一个该信号量对应的资源，同样采用了禁用中断的方式来保证原子性，具体流程为：

* 查询整型变量来了解是否存在多余的可分配的资源，是的话取出资源（整型变量减1），之后当前进程便可以正常进行；
* 如果没有可用的资源，整型变量不是正数，当前进程的资源需求得不到满足，因此将其状态改为SLEEPING态，然后将其挂到对应信号量的等待队列中，调用schedule函数来让出CPU，在资源得到满足，重新被唤醒之后，将自身从等待队列上删除掉；

当多个（>1）进程可以进行互斥或同步合作时，一个进程会由于无法满足信号量设置的某条件而在某一位置停止，直到它接收到一个特定的信号（表明条件满足了）。为了发信号，需要使用一个称作信号量的特殊变量。为通过信号量s传送信号，信号量的V操作采用进程可执行原语semSignal(s)；为通过信号量s接收信号，信号量的P操作采用进程可执行原语semWait(s)；如果相应的信号仍然没有发送，则进程被阻塞或睡眠，直到发送完为止。

1. Ucore中的信号量是基于信号量和条件变量实现的，请说明其中的数据结构和函数方法的设计。

● \_\_down(semaphore\_t \*sem, uint32\_t wait\_state, timer\_t \*timer)：具体实现信号量的P操作，首先关掉中断，然后判断当前信号量的value是否大于0。如果是>0，则表明可以获得信号量，故让value减一，并打开中断返回即可；如果不是>0，则表明无法获得信号量，故需要将当前的进程加入到等待队列中，并打开中断，然后运行调度器选择另外一个进程执行。如果被V操作唤醒，则把自身关联的wait从等待队列中删除（此过程需要先关中断，完成后开中断）。具体实现如下所示：

●\_\_up(semaphore\_t \*sem, uint32\_t wait\_state)：具体实现信号量的V操作，首先关中断，如果信号量对应的wait queue中没有进程在等待，直接把信号量的value加一，然后开中断返回；如果有进程在等待且进程等待的原因是semophore设置的，则调用wakeup\_wait函数将waitqueue中等待的第一个wait删除，且把此wait关联的进程唤醒，最后开中断返回。具体实现如下所示：



练习2: 了解基于信号量和管程的哲学家就餐问题

1. 说明ucore中基于信号量的哲学家就餐问题的实现机制。

基于信号量的哲学家问题，首先实现初始化了一个互斥信号量，然后创建了对应5个哲学家行为的5个信号量，并创建5个内核线程代表5个哲学家，每个内核线程完成了基于信号量的哲学家吃饭睡觉思考行为实现，其中核心就是phi\_take\_forks\_sema和phi\_put\_forks\_sema两个函数。接着是信号量的核心部分，\_\_up函数和\_\_down函数，而这两个函数分别对应着信号量的V，P操作。首先是\_up函数，通过local\_intr\_save函数关闭中断，如果信号量对应的wait queue中没有进程在等待，直接把信号量的value加一，然后通过local\_intr\_restore函数开中断返回。如果有进程在等待且进程等待的原因是semophore设置的，则调用wakeup\_wait函数将wait\_queue中等待的第一个wait删除，且把此wait关联的进程唤醒，最后通过local\_intr\_restore函数开中断返回。

2)说明ucore中基于管程的哲学家就餐问题的实现机制。

基于管程的哲学家问题，ucore中的管程机制是基于信号量和条件变量来实现的。管程中的条件变量cv通过执行wait\_cv，会使得等待某个条件C为真的进程能够离开管程并睡眠，且让其他进程进入管程继续执行；而进入管程的某进程设置条件C为真并执行signal\_cv时，能够让等待某个条件C为真的睡眠进程被唤醒，从而继续进入管程中执行。发出signal\_cv的进程A会唤醒睡眠进程B，进程B执行会导致进程A睡眠，直到进程B离开管程，进程A才能继续执行，这个同步过程是通过信号量next完成的；而next\_count表示了由于发出singal\_cv而睡眠的进程个数。