实验5 同步互斥

仔细阅读实验文档lab7同步互斥，完成以下练习（不做实验文档中的题目）。扩展练习选做，有能力者完成。

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？

答：通过定时器，屏蔽使能中断，等待队列来实现的

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。

答：ucore中最重要的信号量操作是P操作函数down和V操作函数up。

P操作函数down：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1，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。

V操作函数up：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1，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。

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。

答：数据结构包括两部分，value值和一个该信号量的等待队列wait\_queue。函数设计方法是通过两级函数实现的，先是up和down函数参数是信号量，之后这两个函数再调用\_up和\_down在这两个函数中才是pv操作的具体实现。

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。

答：semphore\_t mutex 临界区互斥信号量，semphore\_t s[N] 每个哲学家一个信号量。首先判断左右的状态不是“eatting”然后是自己的状态变成“EATING” up(s[i]); down(&mutex)进入临界区；state\_sema[i]=HUNGRY 记录饥饿；试图得到叉子如果得到up(s[i])；up(&mutex)离开临界区；down(&s[i])如果得不到叉子就等待；down（&mutex）进入临界区；state\_sema[i]=THINKING 改变状态为就餐结束；phi\_test\_sema(LEFT)看左邻局能否进餐；phi\_test\_sema(RIGHT)看一下右邻居能否进餐；up(&mutex)离开临界区。

1. 说明ucore中基于管程的哲学家就餐问题的实现机制。

答：数据结构：struct proc\_struct \*philosopher\_proc\_condvar[N]; int state\_condvar[N]; monitor\_t mt, \*mtp=&mt; void phi\_test\_condvar (i)试图得到叉子。down(&(mtp->mutex)); 进入管程的临界区state\_condvar[i] = HUNGRY; 改变状态为 HUNGRY；phi\_test\_condvar(i); 试图拿到叉子；down(&(mtp->mutex));进入临界区；state\_condvar[i] = THINKING; 改变状态为THINKING；phi\_test\_condvar( LEFT); 左边的得到叉子；phi\_test\_condvar(RIGHT); 右边的得到叉子；up(&(mtp->mutex)); 离开临界区。

扩展练习：了解java中同步互斥的实现机制，说明其与操作系统原理课的管程之间的关系，并用其实现写者优先的读者写者问题。