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？

通过借助定时器，屏蔽使能中断，等待队列来实现的，主要是为了实现一些硬件无法达到的原子操作。

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。

Ucore中的信号量实现机制down是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,就value减一,并打开中断让后;如果不是>0,也就是无法获得信号量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。

对于V操作，首先关中断,如果信号量对应的等待队列（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（通过设立flag实现）,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。

数据结构包括两部分，value值和一个该信号量的等待队列。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。

用信号量实现时，有两个信号量，一个是mutex和是s[i]，前者用于实现临界区的互斥访问，s[i]是用来实现哲学家的信号量。在实现中有三个个关键函数，phi\_take\_forks\_sema（i）函数，流程为：先进入临界区，记录第i个哲学家为饥饿状态，然后试图得到两把叉子，之后离开临界区，如果得不到叉子就阻塞；phi\_put\_forks\_sema(i)流程为：进入临界区，哲学家进餐结束，看左右邻居是否能就餐，离开临界区。phi\_test\_sema(i)该函数又来实现判断第i个哲学家是否是饥饿状态以及是否能够进餐，如果能就把他状态变为进餐，然后对哲学家的信号量s[i]进行v操作。

1. 说明ucore中基于管程的哲学家就餐问题的实现机制。

通过管程实现，定义两个数据结构，一个是monitor，一个是condvar\_t（条件变量）。其中monitor中的mutex是一个0-1变量，用来允许每次只有一个进程进入管程，实现互斥。条件变量数据结构中，由信号量sem用来实现某进程是否需要等待。

整个函数的实现过程：

关于使用条件变量来完成哲学家就餐问题的实现中，总共有两个关键函数，以及使用到了N（哲学家数量）个条件变量，在管程中，还包括了一个限制管程访问的锁还有N个用于描述哲学家状态的变量（总共有EATING, THINKING, HUNGER）三种状态； 首先是phi\_take\_forks\_condvar函数，该函数表示指定的哲学家尝试获得自己所需要进餐的两把叉子，如果不能获得则阻塞，具体实现流程为： 给管程上锁；将哲学家的状态修改为HUNGER；判断当前哲学家是否有足够的资源进行就餐（相邻的哲学家是否正在进餐）；如果能够进餐，将自己的状态修改成EATING，然后释放锁，离开管程即可；如果不能进餐，等待在自己对应的条件变量上，等待相邻的哲学家释放资源的时候将自己唤醒。

而phi\_put\_forks\_condvar函数则是释放当前哲学家占用的叉子，并且唤醒相邻的因为得不到资源而进入等待的哲学家：首先获取管程的锁；将自己的状态修改成THINKING；检查相邻的哲学家是否在自己释放了叉子的占用之后满足了进餐的条件，如果满足，将其从等待中唤醒（使用cond\_signal）；释放锁，离开管程；由于限制了管程中在访问共享变量的时候处于RUNNABLE的进程只有一个，因此对进程的访问是互斥的；并且由于每个哲学家只可能占有所有需要的资源（叉子）或者干脆不占用资源，因此不会出现部分占有资源的现象，从而避免了死锁的产生；

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。