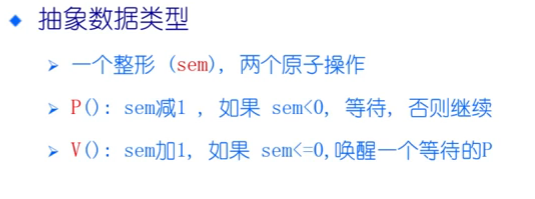
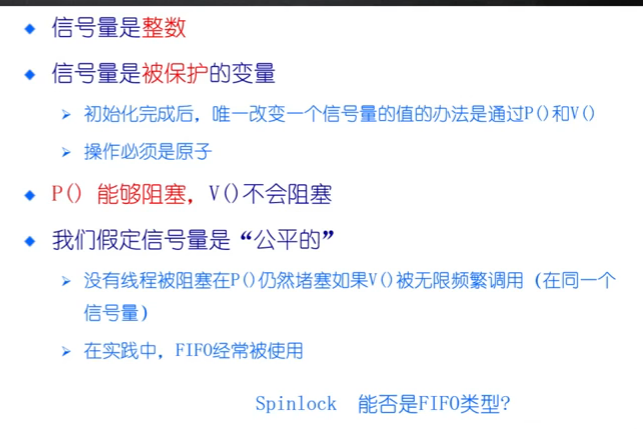
信号量：

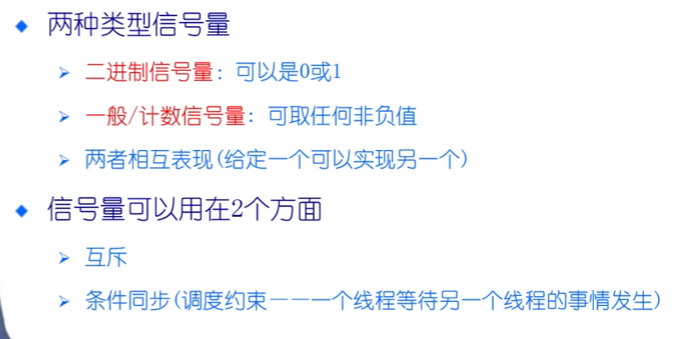


在进入临界区之前执行p操作，离开临界区的时候会执行v操作

信号量一开始一般会被设置为一个大于0的数



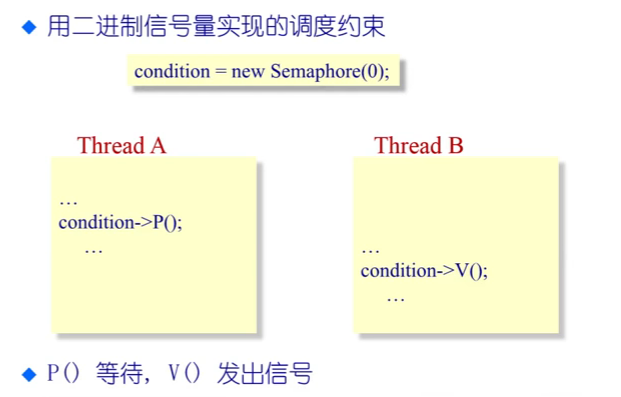
v操作唤醒等待的p堵塞中的进程时，选择的是FIFO，先来先唤醒



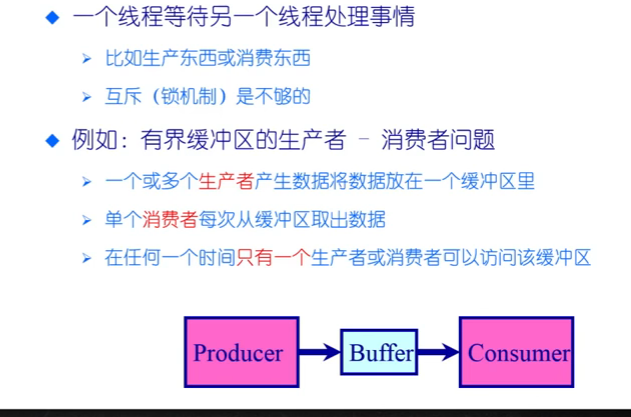


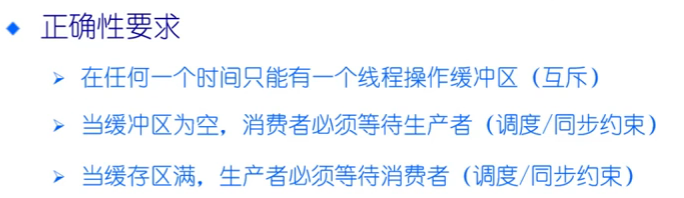
初值为1，实现lock锁操作，互斥

完成同步操作，初值设置为0。 线程a必须要等到线程b执行到某一个地方之后才能执行

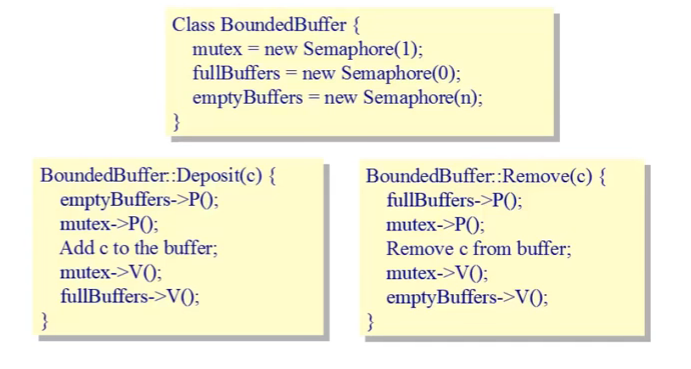


更复杂的同步和互斥：







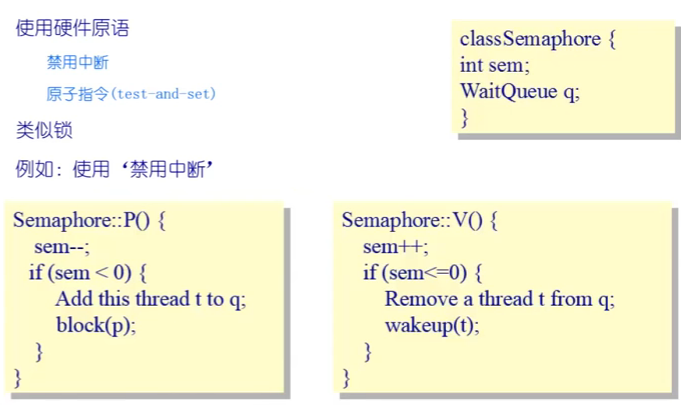


生产者 消费者

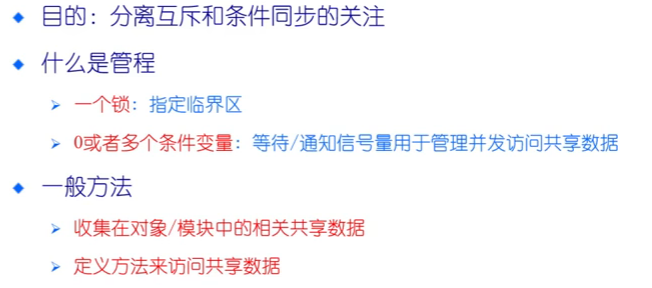
在一个Buffer中的两个v操作的顺序的可以交换，因为v操作只是通知，不会产生阻塞，唤醒需要的进程，所以可以互换。但是p操作不能交换，因为p操作可能会产生阻塞。

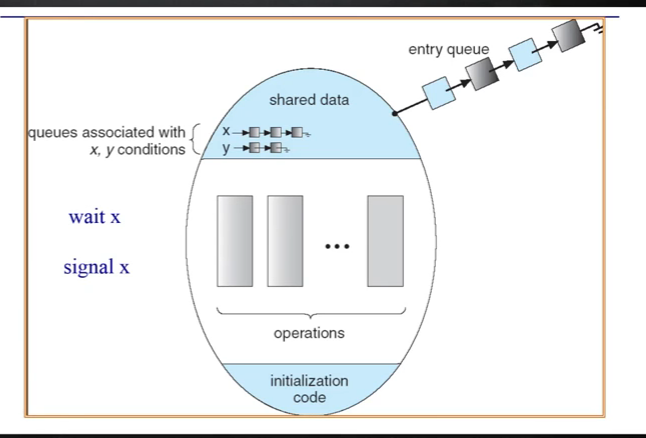
信号量的实现：

信号量包括一个整型以及一个等待队列



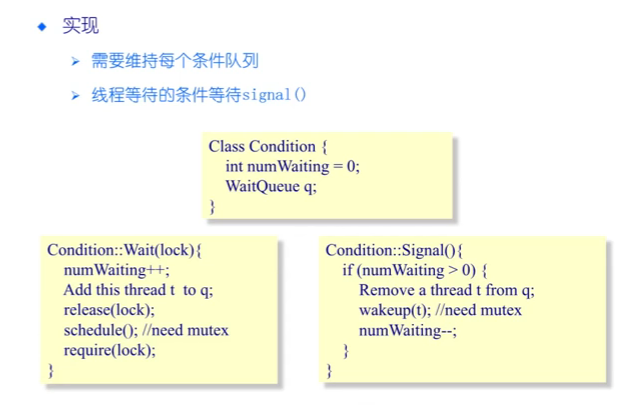
管程：



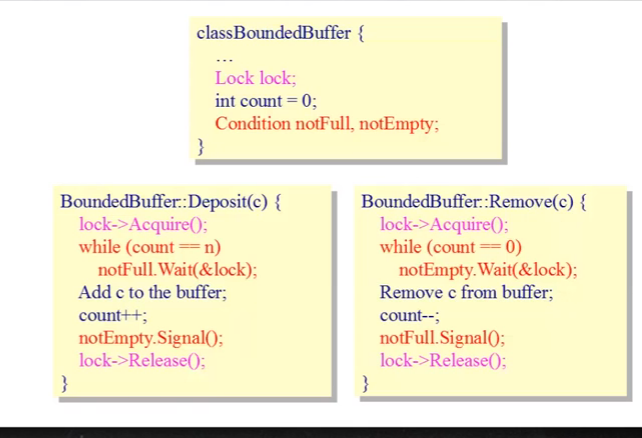




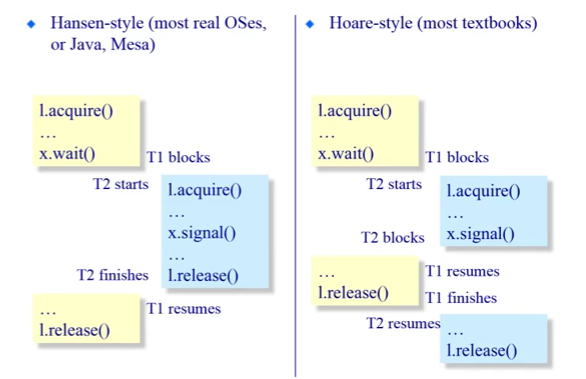
条件变量：



管程解决生产者消费者问题：



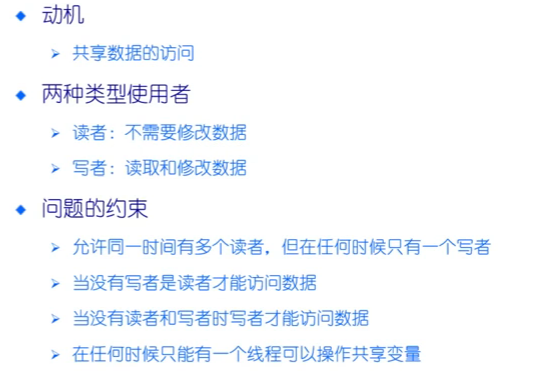
Signal唤醒了一个等待的线程：



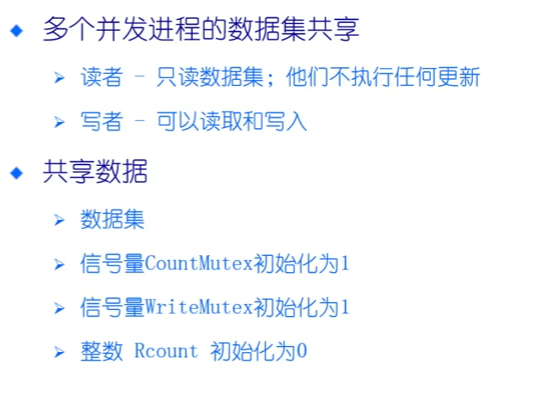
但是唤醒之后是现在线程执行还是被唤醒的线程执行有不同的方式。

经典同步问题：

读者写者问题：

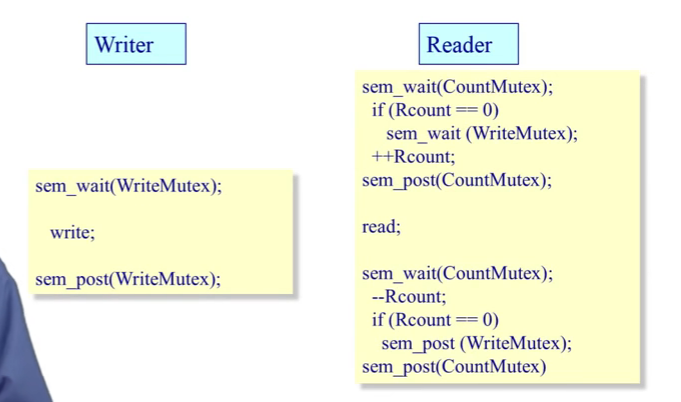


信号量实现：



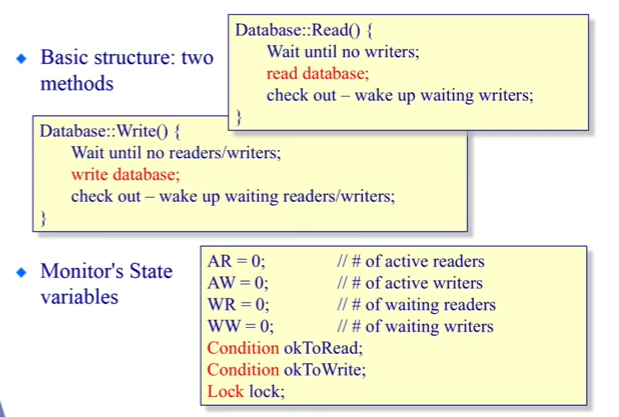
（读者优先？没搞明白）

Rcount记录当前读者的个数



对Rcount进行加减操作时，需要确保多个读者不会同时对其加或者减操作，即互斥

管程实现：



（这里管程的实现太难了，回头再看）

哲学家就餐问题：