5月15日操作系统实验

2017141463062

曹志铭

3.1

答：通过禁用中断实现，在Semaphore::P() 和 Semaphore::V() 中，对于信号量的更改都需要为原子操作。Nachos的实现方法为设置interrupt的level为IntOff，即无法产生中断，系统也就无法将指令执行的handler控制权移交给其他进程，从而实现了原子操作。

3.2

答：Lock类中核心的方法为Lock::Acquire()和Lock::Release()，其中Lock::Acquire()想要实现的功能是请求一个lock资源，也可以理解为等待lock的状态为可用状态，如果可用那么将lock资源占用并设置占用线程为当前线程，其代码只有两行，第一行为将lock的semaphore进行P()操作，即信号量减一，利用信号量的原子性和P()中的逻辑实现了等待锁资源的释放，并由自己占用。第二行将占用的线程设置为自身，为lock的释放提供ASSERT的前置条件。

Lock::Release()需要提前假设当前release的线程为lock占用线程，如果满足就将占用人设空，同时将lock的semaphore执行V()操作，利用semaphore的逻辑V()操作会释放一个阻塞的P()操作，以此实现lock释放交由阻塞线程执行的操作。

Condition类中核心方法为Condition::Wait(Lock\* conditionLock), Condition::Signal(Lock\* conditionLock), Condition::Broadcast(Lock\* conditionLock)。三个方法共同作用实现Condition类中对于线程请求资源，互斥问题和广播资源可用等同步问题。Condition::Wait(Lock\* conditionLock)中对等待队列Append一个value为0的Semaphore，再对这个Semaphore执行P()使其阻塞，conditionLock保证这一过程的原子性。

Condition:Signal(Lock\* conditionLock)对于队首元素进行V()操作，利用Semaphore的特点实现对与条件变量释放的操作，同时将这个信号Remove出队列。

Condition:Broadcast(Lock\* cinditionLock)，对于当前等待队列的执行一次全体SIgnal()，直到队列为空。以此实现广播的效果。

3.3

答：对于生产者每次循环中，先对full这个信号量进行P()操作，以此实现当full条件满足时的阻塞效果。如果不进行阻塞，则先使用ssmutex对之后的临界区代码进行保护，之后在临界区中对buf进行生产而后释放ssmutex互斥锁，由于生产了一个新的产品empty进行V()操作解锁，以便如果有阻塞的消费者进行消费。

对于消费者的每次循环中，先对empty这个信号量尽心V()操作，以实现当空条件满足时的阻塞效果，之后对生产队列进行消费，同样操作临界区代码使用ssmutex进行保护。而后对full信号量进行V()操作以便有阻塞的生产者进行再生产。

./nacos -K

