互斥锁

解决临界区最简单的工具就是锁（mutex lock）。一个进程在进入临界区时应获得锁；在退出临界区时释放锁。函数acquire()获得锁，而函数release()释放锁。

每个互斥锁有一个布尔变量available，表示锁是否可用。如果锁是可用的，调用acquire()会成功，且锁不再可用。当一个进程试图获取不可用的锁时，会被阻塞，直到锁被释放。

```
acquire()
{
    while(!available);//忙等待
    available = false;//获得锁
}
release()
{
    available = true;//释放锁
}
```

acquire()或release()的执行必须是原子操作，因此互斥锁通常采用硬件机制来实现。

互斥锁的主要缺点是忙等待，当有一个进程在临界区中，任何其他进程在进入临界区时必须连续循环调用acquire()。当多个进程共享同一CPU时，就浪费了CPU周期。因此，互斥锁通常用于多处理器系统，一个线程可以在一个处理器上等待，不影响其他线程的执行。

需要连续循环忙等的互斥锁，都可称为自旋锁（spin lock），如TSL指令、swap指令、单标志法

特性：

- 需忙等，进程时间片用完才下处理机，违反“让权等待”
- 优点：等待期间不用切换进程上下文，多处理器系统中，若上锁的时间短，则等待代价很低
- 常用于多处理器系统，忙等的过程中不可能解锁

```
do{
    entry section;//进入区 acquire()
    critical section;//临界区
    exit section;//退出区 release()
    remainder section;//剩余区
}while(true)
```