1. 锁的类型：
   1. 用户模式锁
      1. Volatile
      2. Spinlock自旋锁
   2. 内核模式锁
   3. 互斥锁
   4. 读写锁
   5. 动态计数锁
   6. 监视锁
2. 锁常见的问题：
   1. 死锁：两个线程各占用着对方即将要执行的锁（代码），谁也访问不了。
   2. 线程安全：多线程中，共享变量不是预期的值，比如秒杀的超买超卖问题。
3. 用户模式锁
   1. Volatile关键字
      1. c#编译器会优化代码，比如count=10\*10-100 优化后就是count=0;
      2. 在变量上加上Volatile关键字后，就会取消这种优化
   2. Sponlock旋转锁
4. 读写锁（读多个，写一个）（自旋锁）如果被阻塞，那么就一直循环查看是否释放
   1. 特点
      1. 读的时候，其他线程可以读，但是不能写
      2. 写的时候，其他线程不能写也不能读
   2. **ReaderWriterLock**:定义支持单个写线程和多个读线程的锁。
   3. **ReaderWriterLockSlim**类:表示用于管理资源访问的状态，可实现多线程读取或进行独占式写入访问
   4. 两者的API十分相近，而且Slim相对与前者来说 更加安全
   5. 两者都是实现多个线程可同时读取、只允许一个线程写入的类
   6. ReaderWriterLockSlim的常用方法
5. 互斥锁
   1. Lock(：同一时间只能有统一线程访问)如果阻塞，那么就睡眠
      1. 使用lock后，能使该代码块按照指定的顺序执行，被lock这块代码已经被其中一个线程访问了，那么另外一个线程只能等待。
   2. Lock等同于Monitor
      1. Monitor.Enter(obj)
      2. Monitor.Exit(obj)
   3. Mutex
      1. Mutex.WaitOne();
      2. Mutex.ReleaseMutex();