参考文档：

<https://blog.csdn.net/u011109589/article/details/80242931>

AbstractQueuedSynchronizer

CAS

**Lock VS Synchronized**

AbstractQueuedSynchronizer通过构造一个**基于阻塞的CLH队列**容纳所有的阻塞线程，而对该队列的操作均通过Lock-Free（CAS）操作，但对已经获得锁的线程而言，ReentrantLock实现了偏向锁的功能。

synchronized的底层也是一个基于CAS操作的等待队列，但JVM实现的更精细，把等待队列分为ContentionList和EntryList，目的是为了降低线程的出列速度；当然也实现了偏向锁，从数据结构来说二者设计没有本质区别。但synchronized还实现了自旋锁，并针对不同的系统和硬件体系进行了优化，而Lock则完全依靠系统阻塞挂起等待线程。

当然Lock比synchronized更适合在应用层扩展，可以继承AbstractQueuedSynchronizer定义各种实现，比如实现读写锁（ReadWriteLock），公平或不公平锁；同时，Lock对应的Condition也比wait/notify要方便的多、灵活的多。