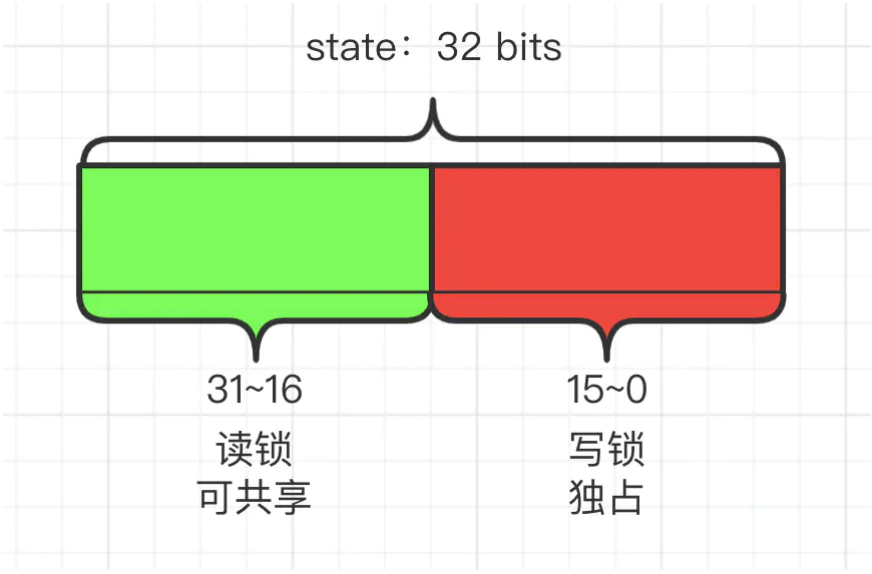
ReentrantReadWriteLock(读写锁)



ReentrantReadWriteLock实现了共享锁的功能，解决读多写少场景下的同步问题，将state变量分为高、低16位，低16位表示写锁、高16位表示读锁(AQS也有Long类型state实现AQLS)，这样就可以实现同时表示读写锁的信息；包含ReadLock、WriteLock成员变量分别实现了Lock接口，提供读写锁功能，默认以非公平方式进行锁竞争.

读锁

- tryAcquireShared

  首先获取同步状态state，先判断写锁是否>0(低16位)，如果有写锁并且不是当前线程，则获取失败直接返回；无写锁或者是当前线程占用的写锁，则判断当前线程是否应该阻塞，非阻塞则获取/重入读锁，利用threadlocal记录重入读锁的次数.

  - nonfairSyn

  非公平的获取读锁，只会在同步队列中第二个等待节点为写锁等待时才会阻塞，否则不会阻塞.

  - fairSyn

  公平的获取读锁，只要同步队列中有有效的等待节点即阻塞.

- tryReleaseShared

  获取(重入)读锁的次数，当前读锁次数减1，cas修改当前state，如果state=0，即当前**读写锁都释放完**了，则唤醒下一个等待线程，否则直接退出.

写锁

- `tryAcquire`

  首先获取同步状态，判断是否有读锁，有读锁则不能再加写锁，直接失败返回；如果没有锁或者是当前线程占用的写锁，则判断是否阻塞，非阻塞则获取/重入写锁.

  - `nonfairSyn`

  非公平的获取读锁，直接不阻塞.

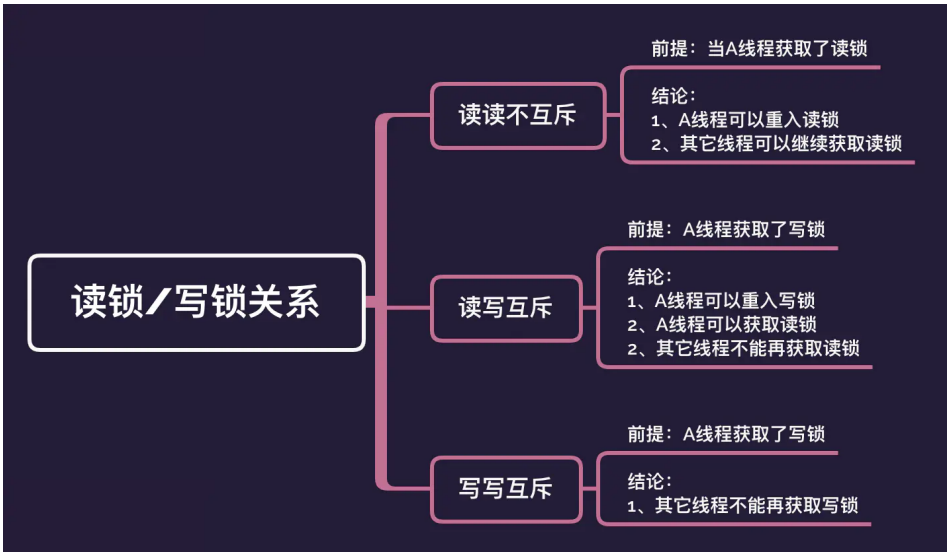
  - `fairSyn`

  公平的获取读锁，只要同步队列中有有效的等待节点即阻塞.

- `tryRelease`

  释放一次写锁，设置`state`，当写锁全部释放时则返回`true`唤醒等待线程，否则返回`false`.

读写锁关系



- A读-AB读
- A写-A读写

参考资料

- [Java 并发之 ReentrantReadWriteLock 深入分析]