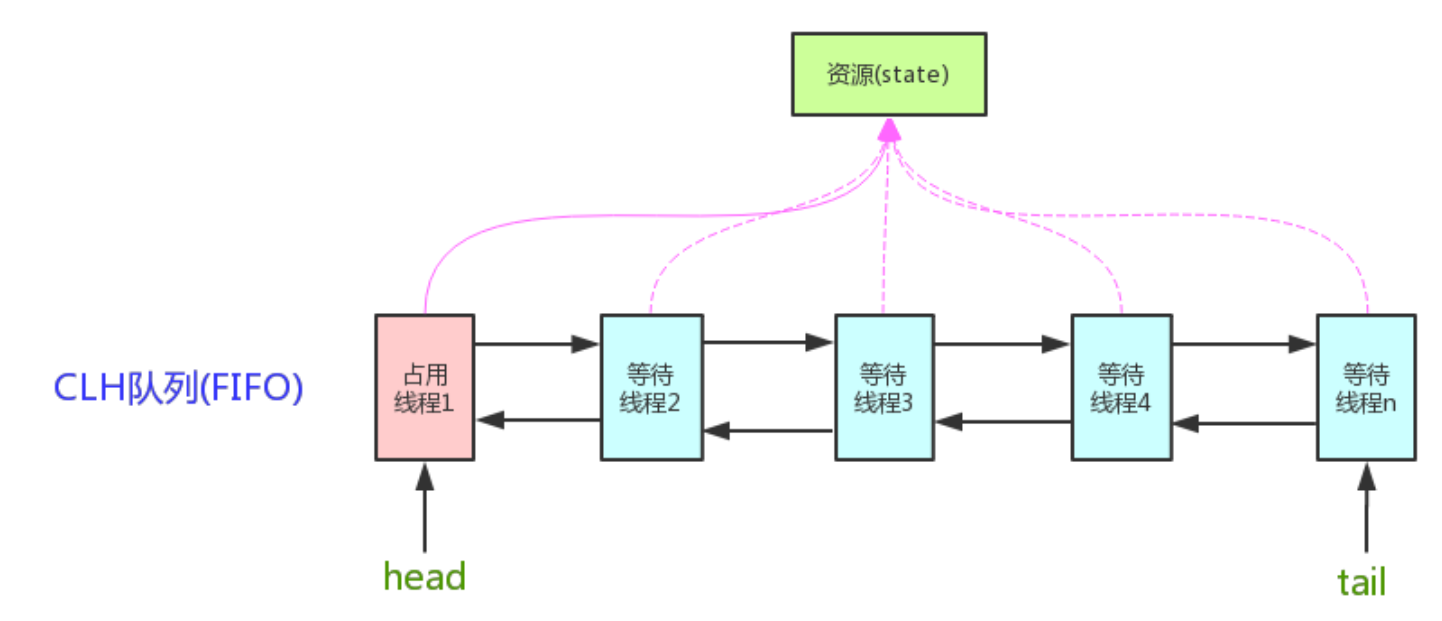
**Java 同步器AQS**，在java.util.concurrent包中很多依赖状态，比如常用的**ReentrantLock**，**ThreadPoolExecutor**等。

**AQS是一个简单的框架，这个框架为同步状态的原子性管理，线程的阻塞和非阻塞以及排队提供了一种通用机制。**表现为一个同步器，主要支持获取锁和释放锁。获取锁的时候如果是独占锁就可能阻塞，如果是共享锁就可能失败。如果是阻塞，线程就要进去阻塞队列，当状态变为可获得锁就修改状态，已进入阻塞队列要从阻塞队列中移出来，释放锁的时候修改状态位以及唤醒阻塞的线程。

**AQS本质是采用了CHL模型完成了一个先进先出的队列模型。对于入队，采用CAS操作，每次比较结尾节点是否一致，然后插入到尾节点，对于出队列，因为每个节点缓存了一个状态位，不满足条件时自旋等待，直到满足条件时将头节点设置为下一个节点。**

CLH(Craig,Landin,and Hagersten)队列是一个虚拟的双向队列（虚拟的双向队列即不存在队列实例，仅存在结点之间的关联关系）。AQS是将每条请求共享资源的线程封装成一个CLH锁队列的一个结点（Node）来实现锁的分配

AQS框架



**它维护了一个volatile int state（代表共享资源）和一个FIFO线程等待队列（多线程竞争资源会进入这个队列）。AQS使用CAS对该同步状态进行原子操作实现对值的修改。**

State的访问方式有三种：

Getstate（）

Setstate（）

compareAndSetState()

AQS定义了两种资源共享方式：Exclusive(**独占**只有一个线程能执行)和share（**共享**，多个线程可以同时执行）

不同的自定义同步器争用共享资源的方式也不同。自定义同步器在实现时只需要实现共享资源state的获取与释放方式即可，自定义同步器实现时主要实现以下几种方法：

IsHeldExclusively（）：该线程是否正在独占资源

tryAcquire（int）：读占方式。尝试获取资源，成功则返回true，失败返回false

**tryAcquireShared(int)：**共享方式。尝试获取资源。负数表示失败；0表示成功，但没有剩余可用资源；正数表示成功，且有剩余资源。

**tryReleaseShared(int)：**共享方式。尝试释放资源，如果释放后允许唤醒后续等待结点返回true，否则返回false。

**AQS对资源的共享方式：（两种）**

* Exclusive（读占）：只有一个线程能执行，如reentrantlock。又可分为公平锁和非公平锁。

公平锁：按照线程在队列中的排队顺序，先到者先拿到锁。

非公平锁：当前线程要获取锁时，无视队列的顺序直接去抢锁。

Share（共享）：多个线程可同时执行，如Semaphore/CountDownLatch。Semaphore、CountDownLatCh、 CyclicBarrier、ReadWriteLock 我们都会在后面讲到。

**Reentrantlock**:(可重入独占锁)：state初试状态为0，表示未锁定状态。A线程lock（）时，会调用tryAcquire（）独占锁并将state + 1，之后其余线程在使用tryAcquire（）时候就会失败。直到unlock（）到state = 0为止，其他线程才有机会获得锁。。A释放锁之前，自己也是可以重复获取此锁（state累加），这就是可重入的概念。

以**CountDownLatch**为例，任务分N个子线程去执行，state就初始化 为N，N个线程并行执行，每个线程执行完之后countDown（）一次，state就会CAS减一。当N子线程全部执行完毕，state=0，会unpark()主调用线程，主调用线程就会从await()函数返回，继续之后的动作。

**Mutex（互斥锁）**

Mutex是一个不可重入的互斥锁。锁资源只有两种状态：0-未锁定 1-表示锁定。