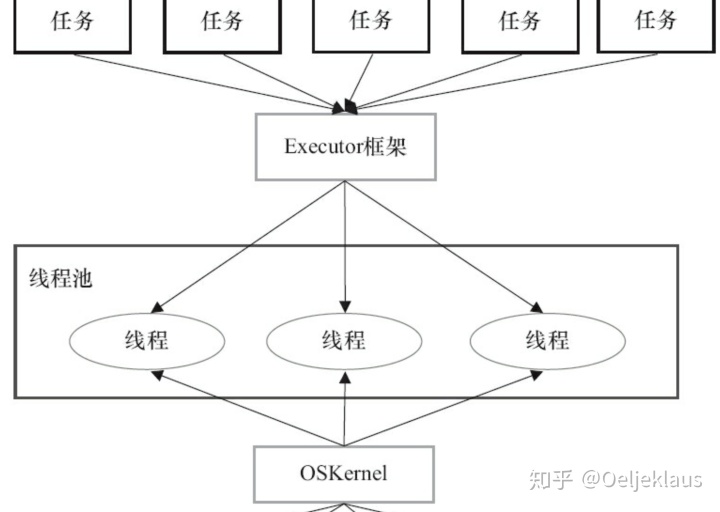
### 1.线程的两级调度机制



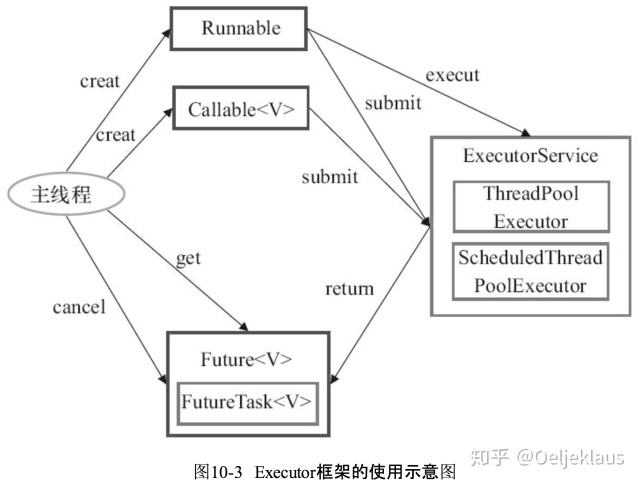
🡺 在上层，Java多线程程序通常把应用分解成为若干任务，然后使用用户级的调度器(Executor框架)将这些任务映射成为固定数量的线程；

🡺 在底层，操作系统内核将这些线程映射到硬件处理器上。

应用程序通过Executor框架控制上层的调度；

而下层的调度由操作系统内核控制，下层的调度不受应用程序的控制。

### 2.Executor框架的结构和成员



#### 主要分为三部分：

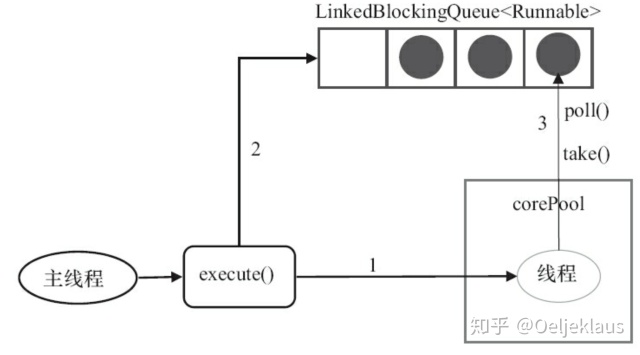
1. 任务：
   * 被执行的任务需要实现Runnable接口和 Callable 接口
2. 任务执行：
   * 任务执行的核心接口Exucutor 以及继承自Executor 的 ExecutorService 接口。
   * Executor框架有两个关键类实现了ExecutorService接口(ThreadPoolExecutor和ScheduThreadPoolExecutor)。
   * ThreadPoolExecutor
     1. SingleThreadExecutor: 创建单个线程的SingleThreadExector，适用于保证顺序的执行各个任务
     2. FixedThreadPool：创建使用固定线程数，使用当前线程数量的应用场景，比较适应负载比较重的服务器
     3. CachedThreadPool：创建一个根据需要创建新线程，大小无界的线程池，适用于执行很多的短期异步任务的小程序。
   * ScheduledThreadPoolExecutor，通常使用工厂类Executors创建
     1. ScheduThreadPoolExecutor：包含若干线程，适用于需要多个后台线程执行周期任务，同时为了满足资源管理的需求而限制后台线程的数量的应用场景。​
     2. SingleThreadScheduledExecutor：包含一个线程，适用于需要单个后台线程执行周期任务，同时需要保证顺序的执行各个任务的应用场景。
3. 异步计算结果
   * 包含接口Future和实现Future接口的FutureTask类。

#### ThreadPoolExecutor 介绍

**1).SingleThreadExecutor：**

使用单个worker线程的Executor

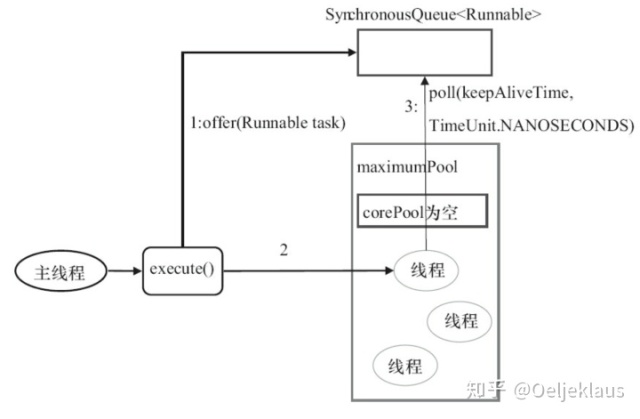
核心线程数和最大线程数设置为1，使用无界队列作为组塞队列



**2).**CachedThreadPool**：**

根据需求创建新线程的线程池。

核心线程池数量为0，最大为整型最大值，KeepAliveTime为60秒，使用SynchronousQueue作为线程池的工作队列

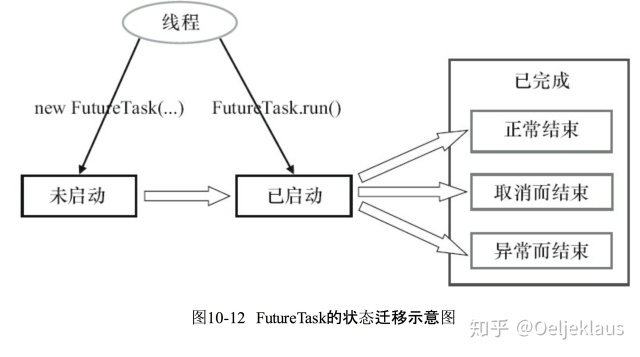


#### FutureTask 介绍

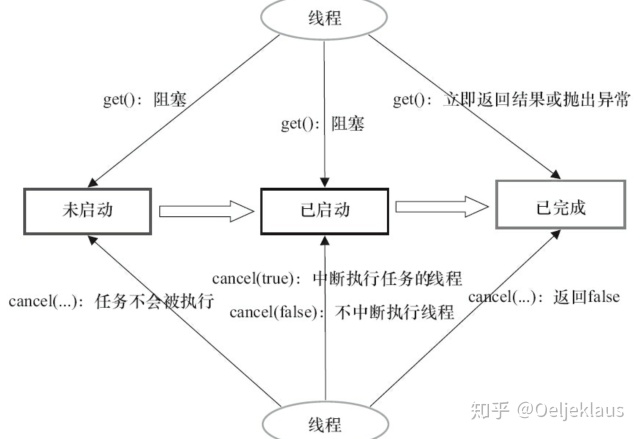
1. **介绍**

FutureTask.run() 方法执行的时机；FutureTask处于下面3种状态

1. 未启动。没有执行run方法之前，这个FutureTask处于未启动状态
2. 已启动。run方法被执行的过程，处于已启动状态
3. 已完成，执行完成



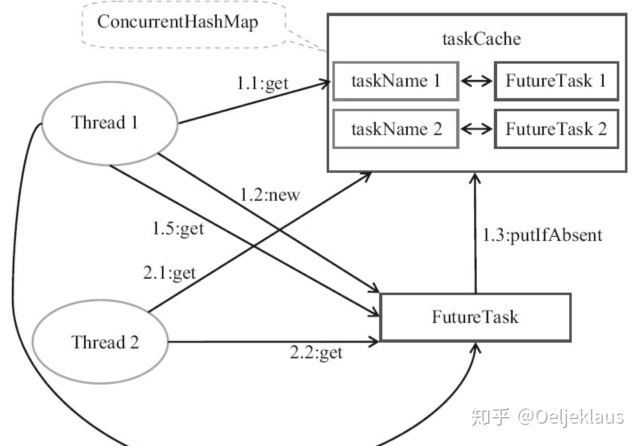
1. **FutureTask的get方法和cancel方法执行示意图**



1. **FutureTask 的使用**

当一个线程等待另一个线程把某个任务执行完成后他才能继续执行，此时可以使用FutureTask。

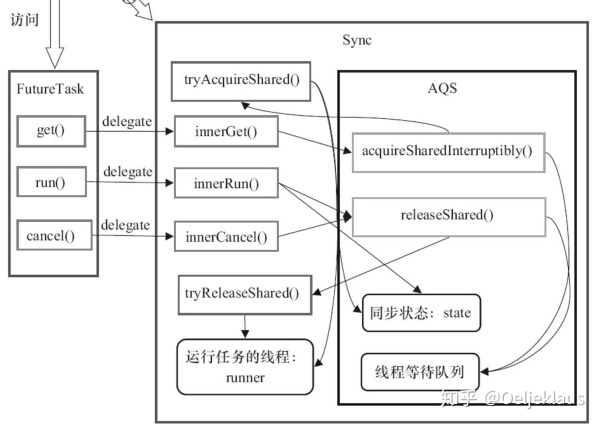
​当多个线程试图同时执行同一个任务时，只允许一个线程执行任务，其他线程需要等待这个任务执行完成在能继续执行。



1. **FutureTask 的实现**

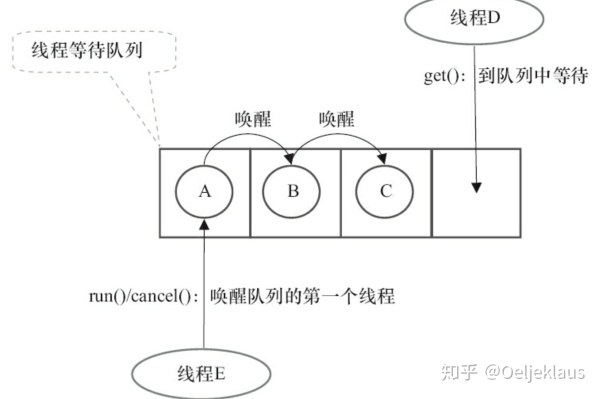
FutureTask的实现基于AQS。

下面是FutureTask的示意图



Sync是FutureTask的内部私有类，继承自AQS。创建FutureTask时会创建内部私有的成员对象Sync，FutureTask所有的公有方法都直接委托给内部私有的Sync。

线程级联唤醒



### 3.Executor框架UML 图

