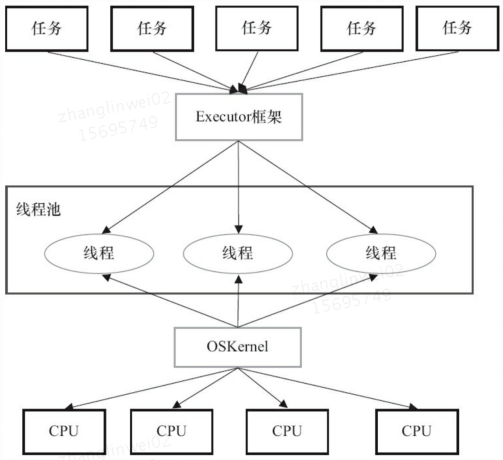
04 Executor框架

C-2 创建: 张林伟, 最后修改: 张林伟 2018-11-01 20:43

Java的线程既是工作单元，也是执行机制。从JDK 5开始，把工作单元和执行机制分离开来。工作单元包括Runnable和Callable，而执行机制由Executor框架提供。

Executor框架简介

Executor框架两级调度模型：

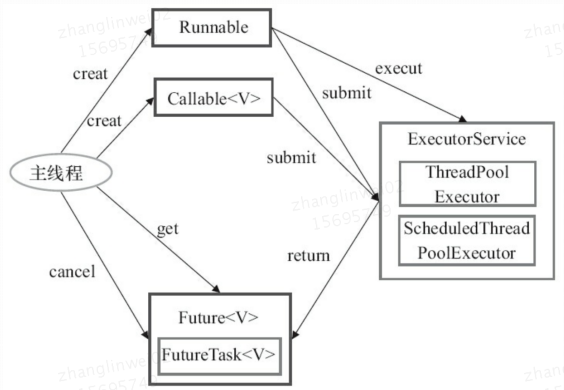


在HotSpot VM的线程模型中，Java线程被一对一映射为本地操作系统线程。

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控制。

Executor框架组成部分

- 任务：Runnable、Callable
- 任务的执行：Executor、ExecutorService
- 异步计算的结果：Future、FutureTask



Executor框架成员

- ThreadPoolExecutor

  通常使用Executors创建

  - FixedThreadPool：固定线程数，适合负载较重服务器
  - SingleThreadExecutor：单个线程，适合顺序执行
  - CachedThreadPool：无界线程池，适合数量大的短期异步任务，或负载较轻服务器
- ScheduledThreadPoolExecutor

  给定延迟后运行或者定期执行命令，通常使用Executors创建

  - ScheduledThreadPoolExecutor：适用于多个后台线程执行周期任务
  - SingleThreadScheduledExecutor：只包含一个线程
- Future
- Runnable和Callable

ScheduledThreadPoolExecutor详解

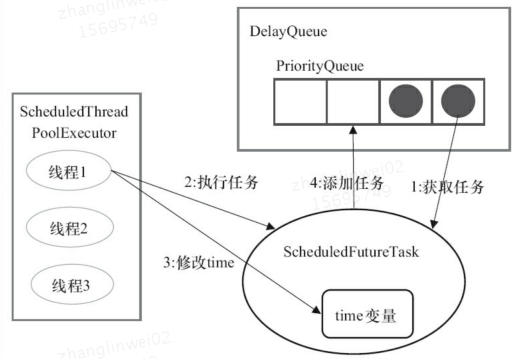
ScheduledThreadPoolExecutor的实现

ScheduledFutureTask（调度任务）

- long time（任务将要被执行的具体时间）
- long sequenceNumber（任务被添加到ScheduledThreadPoolExecutor中的序号）
- long period（任务执行的间隔周期）

ScheduledThreadPoolExecutor会把调度的任务（ScheduledFutureTask）放到一个DelayQueue中。DelayQueue封装了一个PriorityQueue，这个PriorityQueue会对队列中的ScheduledFutureTask进行排序。排序时，time小的排在前面（时间早的任务将先被执行），如果time相同，则比较sequenceNumber（先提交的先执行）。

任务执行步骤

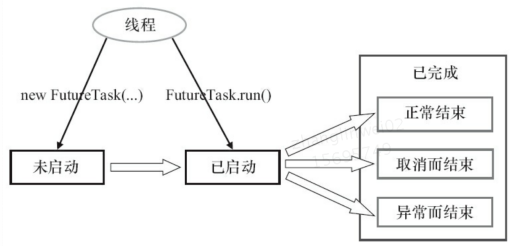


FutureTask详解

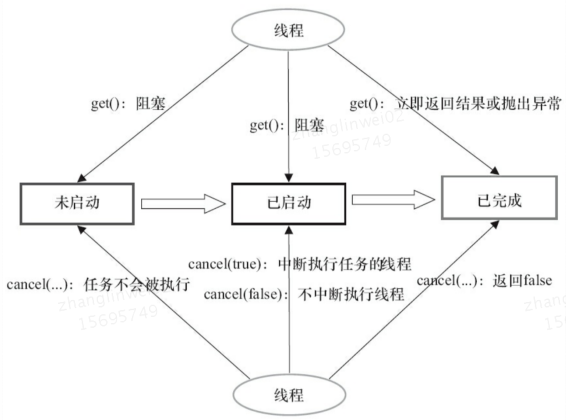
FutureTask状态

- 未启动（创建FutureTask，但未执行run方法）
- 已启动（执行run方法过程）
- 已完成（run方法执行完毕、被cancel方法取消、抛出异常而异常结束）

FutureTask状态迁移图



FutureTask执行get和cancel方法



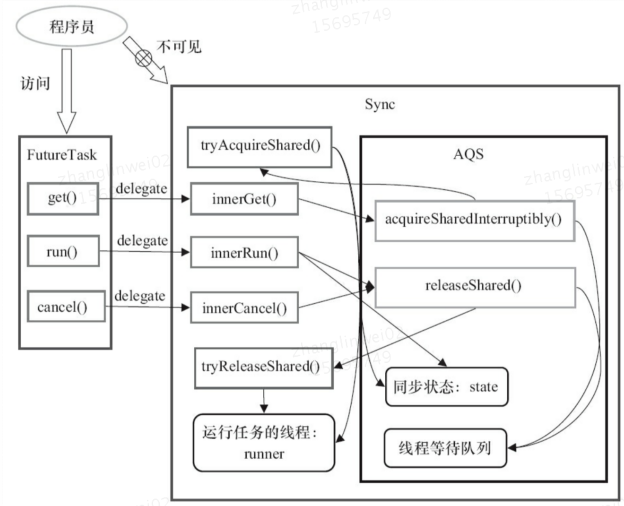
FutureTask实现

FutureTask实现基于AbstractQueuedSynchronizer（AQS）。AQS是一个同步框架，它提供通用机制来原子性管理同步状态、阻塞和唤醒线程，以及维护被阻塞线程的队列。基于AQS实现的同步器包括：ReentrantLock、Semaphore、ReentrantReadWriteLock等。

基于AQS实现的同步器都会包括两种类型操作

- acquire操作。阻塞调用线程，除非/直到AQS的状态允许这个线程继续执行
- release操作。改变AQS状态，改变后的状态可允许一个或多个阻塞线程被解除阻塞

FutureTask设计示意图



Sync是FutureTask的内部私有类，它继承自AQS。