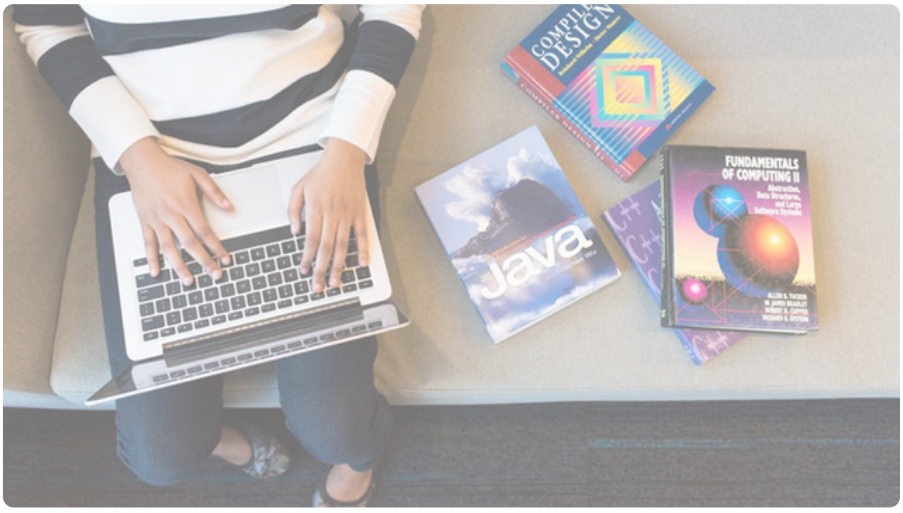
目录

38 线程池源码面试题

更新时间：2019-11-20 10:00:25



> 与有肝胆人共事，从无字句处读书。
>
> ——周恩来

果断更，请联系QQ/微信6426006

引导语

线程池在日常面试中占比很大，主要是因为线程池内容涉及的知识点较广，比如涉及到队列、线程、锁等等，所以很多面试官喜欢把线程池作为问题的起点，然后延伸到其它内容，由于我们专栏已经说过队列、线程、锁面试题了，所以本章面试题还是以线程池为主。

1：说说你对线程池的理解？

答：答题思路从大到小，从全面到局部，总的可以这么说，线程池结合了锁、线程、队列等元素，在请求量较大的环境下，可以多线程的处理请求，充分的利用了系统的资源，提高了处理请求的速度，细节可以从以下几个方面阐述：

1. ThreadPoolExecutor 类结构；
2. ThreadPoolExecutor coreSize、maxSize 等重要属性；
3. Worker 的重要作用；
4. submit 的整个过程。

通过以上总分的描述，应该可以说清楚对线程池的理解了，如果是面对面面试的话，可以边说边画出线程池的整体架构图（见《ThreadPoolExecutor 源码解析》）。

2：ThreadPoolExecutor、Executor、ExecutorService、Runnable、Callable、FutureTask 之间的关系？

答：以上 6 个类可以分成两大类：一种是定义任务类，一种是执行任务类。

一，并增加了对任务的管理功能；

2. 执行任务类：ThreadPoolExecutor、Executor、ExecutorService。Executor 定义最基本的运行接口，ExecutorService 是对其功能的补充，ThreadPoolExecutor 提供真正可运行的线程池类，三个类定义了任务的运行机制。

日常的做法都是先根据定义任务类定义出任务来，然后丢给执行任务类去执行。

3：说一说队列在线程池中起的作用？

答：作用如下：

1. 当请求数大于 coreSize 时，可以让任务在队列中排队，让线程池中的线程慢慢的消费请求，实际工作中，实际线程数不可能等于请求数，队列提供了一种机制让任务可排队，起一个缓冲区的作用；
2. 当线程消费完所有的线程后，会阻塞的从队列中拿数据，通过队列阻塞的功能，使线程不消亡，一旦队列中有数据产生后，可立马被消费。

4：结合请求不断增加时，说一说线程池构造器参数的含义和表现？

答：线程池构造器各个参数的含义如下：

1. coreSize 核心线程数；
2. maxSize 最大线程数；
3. KeepAliveTime 线程空闲的最大时间；
4. queue 有多种队列可供选择，比如：1：SynchronousQueue，为了避免任务被拒绝，要求线程池的 maxSize 无界，缺点是当任务提交的速度超过消费的速度时，可能出现无限制的线程增长；2：LinkedBlockingQueue，无界队列，未消费的任务可以在队列中等待；3：ArrayBlockingQueue，有界队列，可以防止资源被耗尽；
5. 线程新建的 ThreadFactory 可以自定义，也可以使用默认的 DefaultThreadFactory，DefaultThreadFactory 创建线程时，优先级会被限制成 NORM_PRIORITY，默认会被设置成非守护线程；
6. 在 Executor 已经关闭或对最大线程和最大队列都使用饱和时，可以使用 RejectedExecutionHandler 类进行异常捕捉，有如下四种处理策略：ThreadPoolExecutor.AbortPolicy、ThreadPoolExecutor.DiscardPolicy、ThreadPoolExecutor.CallerRunsPolicy、ThreadPoolExecutor.DiscardOldestPolicy。

当请求不断增加时，各个参数起的作用如下：

1. 请求数 < coreSize：创建新的线程来处理任务；
2. coreSize <= 请求数 && 能够成功入队列：任务进入到队列中等待被消费；
3. 队列已满 && 请求数 < maxSize：创建新的线程来处理任务；
4. 队列已满 && 请求数 >= maxSize：使用 RejectedExecutionHandler 类拒绝请求。

5：coreSize 和 maxSize 可以动态设置么，有没有规则限制？

答：一般来说，coreSize 和 maxSize 在线程池初始化时就已经设定了，但我们也可以通过 setCorePoolSize、setMaximumPoolSize 方法动态的修改这两个值。

setCorePoolSize 的限制见如下源码：

果断更，请联系QQ/微信6426006

```
public void setCorePoolSize(int corePoolSize) {
    if (corePoolSize < 0)
        throw new IllegalArgumentException();
    int delta = corePoolSize - this.corePoolSize;
    this.corePoolSize = corePoolSize;
    // 活动的线程大于新设置的核心线程数
    if (workerCountOf(ctl.get()) > corePoolSize)
        // 尝试将可以获得锁的 worker 中断，只会循环一次
        // 最后并不能保证活动的线程数一定小于核心线程数
        interruptIdleWorkers();
    // 设置的核心线程数大于原来的核心线程数
    else if (delta > 0) {
        // 并不清楚应该新增多少线程，取新增核心线程数和等待队列数据的最小值，够用就好
        int k = Math.min(delta, workQueue.size());
        // 新增线程直到k，如果期间等待队列空了也不会再新增
        while (k-- > 0 && addWorker(null, true)) {
            if (workQueue.isEmpty())
                break;
        }
    }
}
```

setMaximumPoolSize 的限制见如下源码：

```
// 如果 maxSize 大于原来的值，直接设置。
// 如果 maxSize 小于原来的值，尝试干掉一些 worker
public void setMaximumPoolSize(int maximumPoolSize) {
    if (maximumPoolSize <= 0 || maximumPoolSize < corePoolSize)
        throw new IllegalArgumentException();
    this.maximumPoolSize = maximumPoolSize;
    if (workerCountOf(ctl.get()) > maximumPoolSize)
        interruptIdleWorkers();
}
```

果断更，请联系QQ/微信6426006

6：说一说对于线程空闲回收的理解，源码中如何体现的？

答：空闲线程回收的时机：如果线程超过 keepAliveTime 时间后，还从阻塞队列中拿不到任务（这种情况我们称为线程空闲），当前线程就会被回收，如果 allowCoreThreadTimeOut 设置成 true，core thread 也会被回收，直到还剩下一个线程为止，如果 allowCoreThreadTimeOut 设置成 false，只会回收非 core thread 的线程。

线程在任务执行完成之后，之所有没有消亡，是因为阻塞的从队列中拿任务，在 keepAliveTime 时间后都没有拿到任务的话，就会打断阻塞，线程直接返回，线程的生命周期就结束了，JVM 会回收掉该线程对象，所以我们说的线程回收源码体现就是让线程不在队列中阻塞，直接返回了，可以见 ThreadPoolExecutor 源码解析章节第三小节的源码解析。

7：如果我想在线程池任务执行之前和之后，做一些资源清理的工作，可以么，如何做？

答：可以的，ThreadPoolExecutor 提供了一些钩子函数，我们只需要继承 ThreadPoolExecutor 并实现这些钩子函数即可。在线程池任务执行之前实现 beforeExecute 方法，执行之后实现 afterExecute 方法。

8：线程池中的线程创建，拒绝请求可以自定义实现么？如何自定义？

目录

口即可；在 ThreadPoolExecutor 初始化时，将两个自定义类作为构造器的入参传递给 ThreadPoolExecutor 即可。

9：说说你对 Worker 的理解?

答：详见《ThreadPoolExecutor 源码解析》中 1.4 小节。

10：说一说 submit 方法执行的过程?

答：详见《ThreadPoolExecutor 源码解析》中 2 小节。

11：说一说线程执行任务之后，都在干啥?

答：线程执行任务完成之后，有两种结果：

1. 线程会阻塞从队列中拿任务，没有任务的话无限阻塞；
2. 线程会阻塞从队列中拿任务，没有任务的话阻塞一段时间后，线程返回，被 JVM 回收。

12：keepAliveTime 设置成负数或者是 0，表示无限阻塞?

答：这种是不对的，如果 keepAliveTime 设置成负数，在线程池初始化时，就会直接报 IllegalArgumentException 的异常，而设置成 0，队列如果是 LinkedBlockingQueue 的话，执行 workQueue.poll (keepAliveTime, TimeUnit.NANOSECONDS) 方法时，如果队列中没有任务，会直接返回 null，导致线程立马返回，不会无限阻塞。

果断更，请联系QQ/微信6426006

如果想无限阻塞的话，可以把 keepAliveTime 设置的很大，把 TimeUnit 也设置的很大，接近于无限阻塞。

13：说一说 Future.get 方法是如何拿到线程的执行结果的?

答：我们需要明确几点：

1. submit 方法的返回结果实际上是 FutureTask，我们平时都是针对接口编程，所以使用的是 Future.get 来拿到线程的执行结果，实际上是 FutureTask.get ，其方法底层是从 FutureTask 的 outcome 属性拿值的；
2. 《ThreadPoolExecutor 源码解析》中 2 小节中详细说明了 submit 方法最终会把线程的执行结果赋值给 outcome。

结合 1、2，当线程执行完成之后，自然就可以从 FutureTask 的 outcome 属性中拿到值。

14：总结

如果我们弄清楚 ThreadPoolExecutor 的原理之后，线程池的面试题都很简单，所以建议大家多看看 《ThreadPoolExecutor 源码解析》这小节。

← 37 ThreadPoolExecutor 源码解析

39 经验总结：不同场景，如何使用线程池 →

精选留言 1

目录

修兮666

你好，不确定ftp的个数，因为是动态配置，想使用ftp连接池，有什么好的思路嘛

0 回复 5天前

千学不如一看，千看不如一练

果断更，请联系QQ/微信6426006