Java开发中线程是经常用到的技术，那么让我们来回顾一下一些线程中经常考虑的问题吧

1、线程分为用户线程和守护线程，有什么区别2、线程有哪些运行状态

**一、什么是线程**

线程，一个执行实体，正在执行的程序，担当分配系统资源（CPU、内存）的实体。一个完整的线程包括，需要运行的逻辑和需要运行需要的资源。

**二、线程的状态**

线程有哪些状态呢，在Thread的代码中的State枚举已经很清楚了

public enum State {NEW,RUNNABLE,BLOCKED,WAITING,TIMED\_WAITING,TERMINATED;}

线程的状态实际是7种，**新建、就绪、运行、阻塞、等待、超时等待、终止。**不过一般将就绪和运行都算作运行中

**新建：新建的线程对象，还未调用start方法**

**就绪：已经具备了运行条件，等待CPU服务，这个状态下，线程在就绪队列中等待**

**运行：线程具备运行条件，并获得了CPU资源**

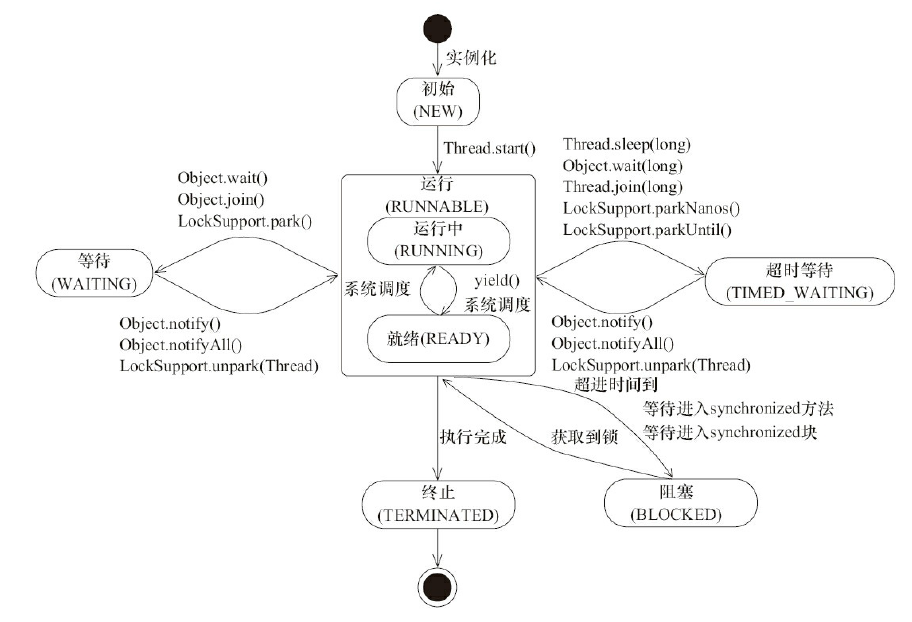
**阻塞：线程被挂起，一般是加锁获取同步状态被阻塞，放到阻塞队列中，阻塞状态消除后，会进入就绪状态**

**等待：等待状态表示当前线程需要等待其他线程做出一些特定动作（等待通知机制）**

**超时等待：在等待的基础上，加上一定时间后返回就绪状态**

**终止：线程已经运行完毕**

那么这些状态是如何切换的呢，借用java并发编程艺术中的一张图



start方法调用后进入运行状态，wait、sleep等方法进入等待状态或，yeild方法进入就绪状态，lock和synchronized进入阻塞状态

**三、线程的分类**

线程可以分为守护线程和用户线程两种，那么什么是用户线程，什么是守护线程呢？

**1、用户线程**

用户线程就是我们平时使用的用来处理逻辑的线程。

**2、守护线程**

**守护线程，是服务线程，程序运行时在后台提供的一种通用服务的线程。**最常见的就是jvm的垃圾回收线程。

那么守护线程相对普通的线程有什么不同，是用来做什么的呢？

1）守护线程主要为其他线程提供服务的2）在jvm中所有用户线程停止运行后，只剩守护线程了,那么jvm也会退出

了解了守护线程的特点了，那么用户线程和守护线程运行的实际效果呢？

首先看下Thread类型中的setDaemon方法，通过这个方法来设置是守护线程和用户线程，默认是用户线程

首先，先创建一个用户线程，代码如下

public static void main(String[] args){Thread task = new Thread(new Runnable(){@Overridepublic void run() {System.out.println("start thread ..." +System.currentTimeMillis());try {Thread.sleep(5000);} catch (InterruptedException e){e.printStackTrace();}System.out.println("end thread..."+System.currentTimeMillis());}});task.start();try {Thread.sleep(2000);} catch (InterruptedException e){e.printStackTrace();}System.out.println("end main..."+System.currentTimeMillis());}

运行结果

start thread ...1589291388584end main...1589291390585end thread...1589291393585

很明显的看到在main函数结束以后，等待线程技术了，jvm才停止运行

那么对代码做一下修改，改为守护线程呢，代码如下

public static void main(String[] args){Thread task = new Thread(new Runnable(){@Overridepublic void run(){System.out.println("start thread ..." +System.currentTimeMillis());try {Thread.sleep(5000);} catch (InterruptedException e){e.printStackTrace();}System.out.println("end thread..."+System.currentTimeMillis());}});task.setDaemon(true);task.start();try {Thread.sleep(2000);} catch (InterruptedException e){e.printStackTrace();}System.out.println("end main..."+System.currentTimeMillis());}

运行结果

start thread ...1589291497224end main...1589291499225

在main函数结束以后，jvm就退出来

守护线程的应用场景：

**1、为用户线程提供服务的线程**

**2、在任何情况下都能关闭的线程（文件的读写等不能直接关闭的操作最好不要使用守护线程）**

守护线程的注意事项

**1、守护线程状态的设置必须在线程启动之前**

**2、守护线程中创建的线程也是守护线程**

那么关于线程的知识点先说到这里

上文中说到线程状态的切换了，分享一个经常讨论的问题sleep和wait的区别

常规的答案

**1、sleep没有释放锁，wait释放了锁**

**2、sleep是Thread的静态方法，wait是锁对象的方法**

**3、sleep是休眠一段时间，wait需要notify唤醒**

是否感觉理解的太浅了，咱们一块研究的深入一点

分别说一下sleep和wait从开始休眠到唤醒的状态变化

1、sleep被调用后，释放CPU资源，并在指定时间内不再去争夺CPU资源，相当于告诉操作系统，在N时间内不需要管我了，这个过程中线程会进入等待队列中。当休眠时间结束以后，线程进入就绪状态，并进入就绪队列，等待分配系统资源。

这里经常会有一个误区，就是认为在线程等到时间结束后，会直接进入运行状态。线程只是进入就绪状态。

现在了解了sleep调用过程中发生了什么，那么可以考虑下，在线程代码中会经常发现有Thread.sleep（0）这种休眠0毫秒操作，这是为了什么呢？

2、wait在调用的时候，首先释放锁状态，然后线程进入到等待状态，当其他线程调用notify时，会随机唤醒一个等到线程，个人理解是把线程移入锁对象的阻塞队列当中，然后等到获取锁状态，进入运行状态。如果调用notifyAll方法会把所有的线程都唤醒，去抢占锁，在抢占结束以后，没有获取到锁的线程进入阻塞状态。