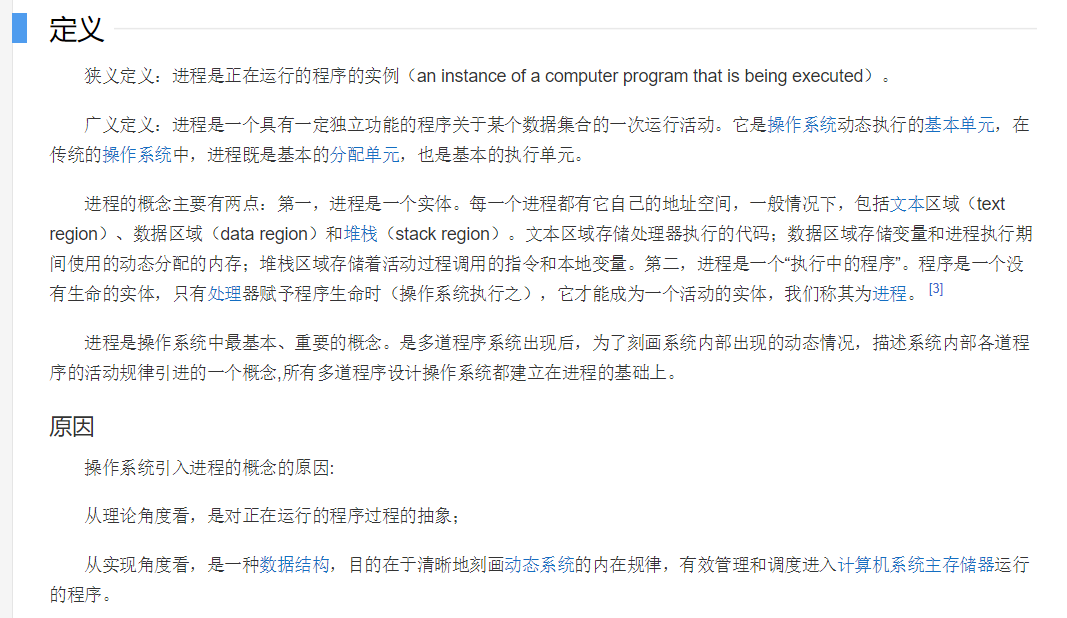
引：

如果对什么是线程，什么是进程



<https://baike.baidu.com/item/%E7%BA%BF%E7%A8%8B/103101?fr=aladdin>

https://baike.baidu.com/item/%E8%BF%9B%E7%A8%8B/382503?fr=Aladdin

用多线程只有一个目的：

那就是更好的利用CPU资源，应为所有的多线程代码都可以用单线程来实现，说这个话其实只有一半对，应为反应“多角色”程序代码最起码每个角色要给他一个线程，否则连实际的场景都无法模拟，当然也没说能用单线程来实现：比如，最常见的“生产者，消费者模型

很多人都对其中的一些概念不够明确：比如同步，并发，

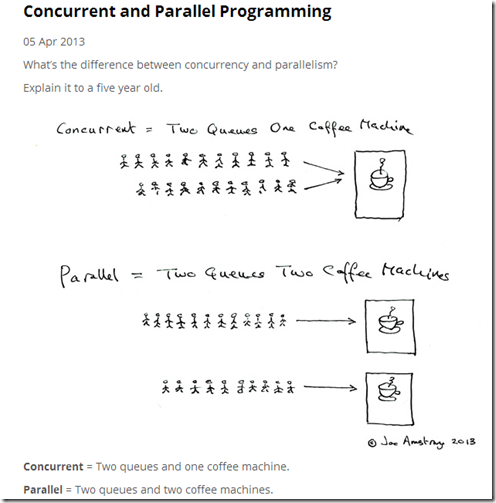
多线程：之的是这个程序（一个进程）运行时产生了不止一个线程。

并行（Parallel）与并发（concurrent）：

并行。多个cpu实例或者多台机器同时执行一段处理逻辑，是真正的同时。

并发：通过CPU调度算法：

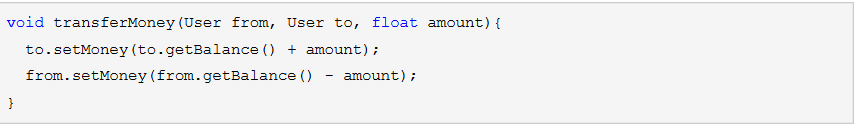
让用户看上去同时执行，实际上从CPU操作层面不是真正的同时，并发往往在场景中有公用的资源，那么针对这个公用的资源玩玩产生瓶颈，我们会用TPS或者QPS来反应这个系统的处理能力。



并发与并行

线程安全：经常用来描绘一段代码，指在并发的情况下，该代码经过多线程使用，现场的调度顺序不影响任何结果。这个时候使用多线程，我们只要关注系统的内存，CPU是不是够用即可，

反过来，线程不安全就意味着现场的调度顺序会影响最终结果。如不加事物的转账代码：



同步：Java中的同步至的通过认为的控制和调度，保证共享资源的多线程访问成为现场安全的，保证结果正确，如上面代码简单加入@ synchronized 关键字。在保证结果准确的同时，提高性能，才是优秀的程序，现场安全性的优先级高于性能。

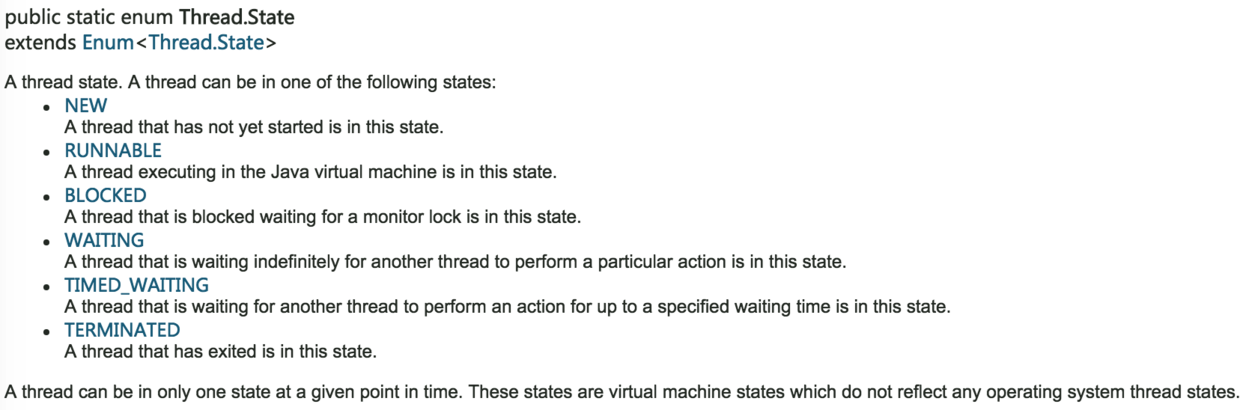
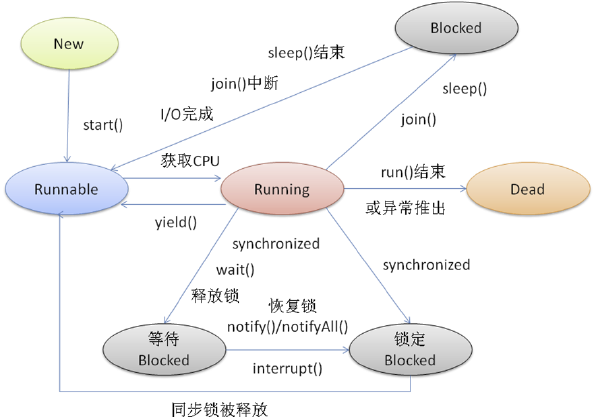
1：扎好马步：线程的状态

2：内功心法：每个对象都有的方法（机制）

3：太祖长拳：基本线程类

4：九阴真经：高级多线程控制类

扎好马步：现场的状态：

线程状态转换：

线程在Running的过程中可能会遇到阻塞（Blocked）情况：

1：调用join()和sleep()方法，sleep()时间结束或被打断，join()中断，IO完成都会回到Runnable状态：等待JVM的调度。

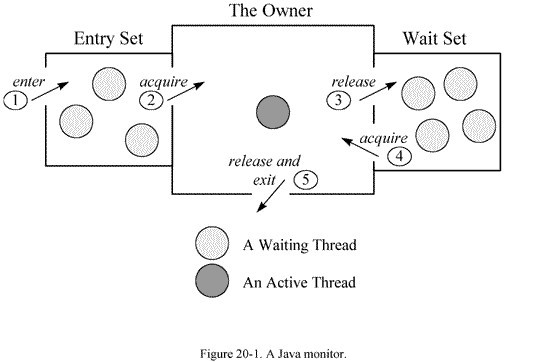
2：调用wait(),使该线程处于等待池（wait blocked pool），直到notify()/nofityAll(),线程非唤醒放到锁定池（lock blocked pool）,释放同步锁使用线程回到可运行状态（Runnable）

3:对Running 状态的线程加 同步锁 Synchronized 使其进入锁定池（lock blocked pool）

同步锁被释放进入可运行状态（Runnable）；

此外：在runnable 状态的线程处于被调度的线程，此时的调度顺序不一定的，Thread类中的yield方法可以让一个running状态转入Runnable状态。

内功心法：每个对象都有的方法：（机制）

Synchronized,wait,notify 是任何对象都具有的同步工具，让我们来了解他们：

他们是应用于同步问题的人工线程调度工具。讲其本质，首先就要明确monitor的概念，Java中的每个对象都有一个监视器，来监测并发代码的重入。在非多线程编码时该监视器不发挥作用，反之如果在synchronized 范围内，监视器发挥作用。

wait/notify必须存在于synchronized块中。并且，这三个关键字针对的是同一个监视器（某对象的监视器）。这意味着wait之后，其他线程可以进入同步块执行。

当某代码并不持有监视器的使用权时（如图中5的状态，即脱离同步块）去wait或notify，会抛出java.lang.IllegalMonitorStateException。也包括在synchronized块中去调用另一个对象的wait/notify，因为不同对象的监视器不同，同样会抛出此异常。