JAVA教程：解析Java的多线程机制 PDF转换可能丢失图片或格式，建议阅读原文
https://www.100test.com/kao_ti2020/145/2021_2022_JAVA_E6_95_99_E7_A8_8B_c104_145480.htm 一、进程与应用程序的区别 进程（Process）是最初定义在Unix等多用户、多任务操作系统环境下用于表示应用程序在内存环境中基本执行单元的概念。以Unix操作系统为例，进程是Unix操作系统环境中的基本成分、是系统资源分配的基本单位。Unix操作系统中完成的几乎所有用户管理和资源分配等工作都是通过操作系统对应用程序进程的控制来实现的。 C、C 、Java等语言编写的源程序经相应的编译器编译成可执行文件后，提交给计算机处理器运行。这时，处在可执行状态中的应用程序称为进程。从用户角度来看，进程是应用程序的一个执行过程。从操作系统核心角度来看，进程代表的是操作系统分配的内存、CPU时间片等资源的基本单位，是为正在运行的程序提供的运行环境。进程与应用程序的区别在于应用程序作为一个静态文件存储在计算机系统的硬盘等存储空间中，而进程则是处于动态条件下由操作系统维护的系统资源管理实体。多任务环境下应用程序进程的主要特点包括： ●进程在执行过程中有内存单元的初始入口点，并且进程存活过程中始终拥有独立的内存地址空间； ●进程的生存期状态包括创建、就绪、运行、阻塞和死亡等类型； ●从应用程序进程在执行过程中向CPU发出的运行指令形式不同，可以将进程的状态分为用户态和核心态。处于用户态下的进程执行的是应用程序指令、处于核心态下的应用程序进程执行的是操作系统指令。 在Unix操作系统启动过程中，系统自动创建swapper、init

等系统进程，用于管理内存资源以及对用户进程进行调度等。在Unix环境下无论是由操作系统创建的进程还要由应用程序执行创建的进程，均拥有唯一的进程标识（PID）。 二、进程与Java线程的区别 应用程序在执行过程中存在一个内存空间的初始入口点地址、一个程序执行过程中的代码执行序列以及用于标识进程结束的内存出口点地址，在进程执行过程中的每一时间点均有唯一的处理器指令与内存单元地址相对应。 Java语言中定义的线程（Thread）同样包括一个内存入口点地址、一个出口点地址以及能够顺序执行的代码序列。但是进程与线程的重要区别在于线程不能够单独执行，它必须运行在处于活动状态的应用程序进程中，因此可以定义线程是程序内部的具有并发性的顺序代码流。 Unix操作系统和Microsoft Windows操作系统支持多用户、多进程的并发执行，而Java语言支持应用程序进程内部的多个执行线程的并发执行。多线程的意义在于一个应用程序的多个逻辑单元可以并发地执行。但是多线程并不意味着多个用户进程在执行，操作系统也不把每个线程作为独立的进程来分配独立的系统资源。进程可以创建其子进程，子进程与父进程拥有不同的可执行代码和数据内存空间。而在用于代表应用程序的进程中多个线程共享数据内存空间，但保持每个线程拥有独立的执行堆栈和程序执行上下文（Context）。 基于上述区别，线程也可以称为轻型进程 (Light Weight Process，LWP)。不同线程间允许任务协作和数据交换，使得在计算机系统资源消耗等方面非常廉价。 线程需要操作系统的支持，不是所有类型的计算机都支持多线程应用程序。Java程序设计语言将线程支持与语言运行环境结合在一起，提供了多任务并发执行的能

力。这就好比一个人在处理家务的过程中，将衣服放到洗衣机中自动洗涤后将大米放在电饭锅里，然后开始做菜。等菜做好了，饭熟了同时衣服也洗好了。 需要注意的是：在应用程序中使用多线程不会增加 CPU 的数据处理能力。只有在多CPU 的计算机或者在网络计算体系结构下，将Java程序划分为多个并发执行线程后，同时启动多个线程运行，使不同的线程运行在基于不同处理器的Java虚拟机中，才能提高应用程序的执行效率。另外，如果应用程序必须等待网络连接或数据库连接等数据吞吐速度相对较慢的资源时，多线程应用程序是非常有利的。基于Internet的应用程序有必要是多线程类型的，例如，当开发要支持大量客户机的服务器端应用程序时，可以将应用程序创建成多线程形式来响应客户端的连接请求，使每个连接用户独占一个客户端连接线程。这样，用户感觉服务器只为连接用户自己服务，从而缩短了服务器的客户端响应时间。 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com