详细分析操作系统线程&进程 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/259/2021_2022__E8_AF_A6_E7_BB_86_E5_88_86_E6_c100_259124.htm 说法一：进程是具有一定独立功能的程序关于某个数据集合上的一次运行活动,进程是系统进行资源分配和调度的一个独立单位. 线程是进程的一个实体,是CPU调度和分派的基本单位,它是比进程更小的能独立运行的基本单位.线程自己基本上不拥有系统资源,只拥有一点在运行中必不可少的资源(如程序计数器,一组寄存器和栈),但是它可与同属一个进程的其他的线程共享进程所拥有的全部资源. 一个线程可以创建和撤销另一个线程.同一个进程中的多个线程之间可以并发执行 说法二：进程和线程都是由操作系统所体会的程序运行的基本单元，系统利用该基本单元实现系统对应用的并发性。进程和线程的区别在于： 简而言之,一个程序至少有一个进程,一个进程至少有一个线程. 线程的划分尺度小于进程，使得多线程程序的并发性高。 另外，进程在执行过程中拥有独立的内存单元，而多个线程共享内存，从而极大地提高了程序的运行效率。 线程在执行过程中与进程还是有区别的。每个独立的线程有一个程序运行的入口、顺序执行序列和程序的出口。但是线程不能够独立执行，必须依存在应用程序中，由应用程序提供多个线程执行控制。 从逻辑角度来看，多线程的意义在于一个应用程序中，有多个执行部分可以同时执行。但操作系统并没有将多个线程看做多个独立的应用，来实现进程的调度和管理以及资源分配。这就是进程和线程的重要区别。 说法三：多线程共存于应用程序中是现代操作系统中的基本特征和重要标志。

用过UNIX操作系统的读者知道进程，在UNIX操作系统中，每个应用程序的执行都在操作系统内核中登记一个进程标志，操作系统根据分配的标志对应用程序的执行进行调度和系统资源分配，但进程和线程有什么区别呢？ 进程和线程都是由操作系统所体会的程序运行的基本单元，系统利用该基本单元实现系统对应用的并发性。进程和线程的区别在于：线程的划分尺度小于进程，使得多线程程序的并发性搞。 另外，进程在执行过程中拥有独立的内存单元，而多个线程共享内存，从而极大地提高了程序的运行效率。 线程在执行过程中与进程还是有区别的。每个独立的线程有一个程序运行的入口、顺序执行序列和程序的出口。但是线程不能够独立执行，必须依存在应用程序中，由应用程序提供多个线程执行控制。 从逻辑角度来看，多线程的意义在于一个应用程序中，有多个执行部分可以同时执行。但操作系统并没有将多个线程看做多个独立的应用，来实现进程的调度和管理以及资源分配。这就是进程和线程的重要区别。 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）。 说法四：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。 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