一：进程

1、进程管理

进程是正在运行的程序实体，并且包括这个运行的程序中占据的所有系统资源，比如说CPU（寄存器），IO,内存，网络资源等。

进程管理是操作系统的职能之一，主要是对处理机进行管理。为了提高 CPU 的利用率而采用多道程序技术。通过进程管理来协调多道程序之间的关系，使 CPU 得到充分的利用。

2、进程的状态有哪些

进程主要有以下三种状态：

- 就绪(Ready)状态

当进程已分配到除 CPU 以外的所有必要的资源，只要获得处理机便可立即执行，这时的进程状态称为就绪状态。

- 执行（Running）状态

当进程已获得处理机，其程序正在处理机上执行，此时的进程状态称为执行状态。

- 阻塞(Blocked)状态

正在执行的进程，由于等待某个事件发生而无法执行时，便放弃处理机而处于阻塞状态。引起进程阻塞的事件可有多种，例如，等待 I/O 完成、申请缓冲区不能满足、等待信件(信号)等。

3、进程三种状态间的转换：

一个进程在运行期间，不断地从一种状态转换到另一种状态，它可以多次处于就绪状态和执行状态，也可以多次处于阻塞状态。

(1) 就绪→执行

处于就绪状态的进程，当进程调度程序为之分配了处理机后，该进程便由就绪状态转变成执行状态。

(2) 执行→就绪

处于执行状态的进程在其执行过程中，因分配给它的一个时间片已用完而不得不让出处理机，于是进程从执行状态转变成就绪状态。

(3) 执行→阻塞

正在执行的进程因等待某种事件发生而无法继续执行时，便从执行状态变成阻塞状态。

(4) 阻塞→就绪

处于阻塞状态的进程，若其等待的事件已经发生，于是进程由阻塞状态转变为就绪状态。

二：线程

1、线程的状态有哪些

1）新建状态(new)：

创建线程对象，但是没有交给 CPU，也就是没有调用 start()方法，只有一个 Thread 对象，还没有一个真正的线程，每个线程只存在一次新建状态；

2）运行状态(runnable)：

调用 start()方法，启动线程，start()方法只能调用一次，该状态的线程随时被 CPU 执行，因为控制权在 CPU（这部分也可以称为可运行状态，Java 线程中将可运行和运行中两种状态合并称为“运行”）。当线程被 CPU 执行时，线程就处于运行状态，其目标方法 run()方法将会被执行；

3）死亡状态(terminated)：

当 run()方法结束，线程就死去，线程一旦死亡，就不能复生（不能调用 start()方法）；

4）阻塞/等待/睡眠状态(blocked)：

线程是活的，但没有条件运行（例如：对象锁被占用时，线程处于阻塞状态）；

5）有期限等待(waiting)：

sleep(long)：线程睡眠方法，在指定时间之后醒来，转成可运行状态；

wait(long)：线程等待方法，在指定时间之后醒来，转成可运行状态；

join(long)：某个线程等待其他线程，在指定时间之后不再等待，转成可运行状态。

6）无限期等待(timed-waiting)：

sleep()、wait()、join()、yield()，可以通过被其他线程通知执行 notify()、notifyAll()；yield()会主动让出 CPU 资源给其他线程，但有可能又被 CPU 拿来执行。

三：进程和线程的区别

进程：是并发执行的程序在执行过程中分配和管理资源的基本单位，是一个动态概念，竞争计算机系统资源的基本单位。

线程：是进程的一个执行单元，是进程内科调度实体。比进程更小的独立运行的基本单位。线程也被称为轻量级进程。

一个程序至少一个进程，一个进程至少一个线程。

进程线程的区别：

地址空间：同一进程的线程共享本进程的地址空间，而进程之间则是独立的地址空间。

资源拥有：同一进程内的线程共享本进程的资源如内存、I/O、cpu 等，但是进程之间的资源是独立的。

一个进程崩溃后，在保护模式下不会对其他进程产生影响，但是一个线程崩溃整个进程都死掉。所以多进程要比多线程健壮。

进程切换时，消耗的资源大，效率高。所以涉及到频繁的切换时，使用线程要好于进程。同样如果要求同时进行并且又要共享某些变量的并发操作，只能用线程不能用进程

执行过程：每个独立的进程程有一个程序运行的入口、顺序执行序列和程序入口。但是线程不能独立执行，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。

线程是处理器调度的基本单位，但是进程不是。

两者均可并发执行。

优缺点：

线程执行开销小，但是不利于资源的管理和保护。线程适合在 SMP 机器（双 CPU 系统）上运行。

进程执行开销大，但是能够很好的进行资源管理和保护。进程可以跨机器前移。

何时使用多进程，何时使用多线程？

对资源的管理和保护要求高，不限制开销和效率时，使用多进程。

要求效率高，频繁切换时，资源的保护管理要求不是很高时，使用多线程。