**操作系统**

1. **进程线程区别和联系，多进程和多线程的区别和联系**

**进程**是一个**执行中程序的实例**，它可以抽象为**一个独立的逻辑控制流和一个私有的地址空间，而**多个进程实际上是轮流使用处理器的，每个进程的指令在物理（CPU）上是周期性的停顿的，在逻辑上可以看做连续的。每一个进程都有一个自己的地址空间，即进程空间或（虚空间）。进程空间的大小只与处理机的位数有关。内核为每个进程维持一个**上下文**，它是内核重新启动一个被抢占的进程所需要的状态，主要包括寄存器、程序计数器、用户站、内核栈、和各种内核数据结构，比如**页表（描述地址空间）、进程表（包含有关当前进程信息）、包含已打开文件信息的文件表。**

**线程**是运行在进程上下文中的逻辑流，是程序执行流的最小单元。线程由内核自动调度。每个线程都有他自己的**线程上下文，**包括一个唯一的线程ID（TID）、栈、栈指针、程序计数器、通用目的寄存器和条件码。所有运行在一个进程里的线程共享该进程的整个虚拟内存空间，包括进程的代码、数据、堆、共享库和打开的文件。

**进程和线程的区别和联系：**

1. 多线程的执行模型和多进程是相似的，线程上下文切换和进程上下文切换的三个逻辑过程是一致的。线程的上下文比进程的上下文小得多，所以线程的上下文切换比进程快得多；
2. 进程是按照父子层次组织，线程和同一个进程中的相关线程组成一个**对等线程池**，独立于其他线程创建的线程。（一个线程可以杀死它的任何对等线程，或者等待它的对等线程终止，每个对等线程都能读写相同的共享数据）。
3. 线程的改变只代表了 CPU 执行过程的改变，而没有发生进程所拥有的资源变化。除了 CPU 之外，计算机内的软硬件资源的分配与线程无关，线程只能共享它所属进程的资源。
4. 进程一般有三个状态：就绪状态、执行状态和等待状态（或称阻塞状态），线程也有就绪、阻塞和运行三种基本状态。
5. 在多处理机系统中，对于传统的进程，即单线程进程，不管有多少处理机，该进程只能运行在一个处理机上。但对于多线程进程，就可以将一个进程中的多个线程分配到多个处理机上，使它们并行执行，这无疑将加速进程的完成。因此，现代处理机OS都无一例外地引入了多线程。
6. 在传统的操作系统中，CPU调度和分派的基本单位是进程。而在引入线程的操作系统中，则把线程作为CPU调度和分派的基本单位，进程则作为资源拥有的基本单位，从而使传统进程的两个属性分开，线程编程轻装运行，这样可以显著地提高系统的并发性。同一进程中线程的切换不会引起进程切换，从而避免了昂贵的系统调用，但是在由一个进程中的线程切换到另一进程中的线程，依然会引起进程切换。

**2.一个进程可以创建多少线程，和什么有关**

进程创建的线程数和线程自身的空间（包括栈、线程局部存储和寄存器）、进程的其他空间（进程的堆、代码段、共享库等）有关，默认线程分配栈空间1MB（可以修改），而每个进程的用户进程空间固定（Linux是3GB,Windows是2GB），所以一个进程创建的线程数=（用户进程空间-进程的其他空间（进程的堆、代码段、共享库等））/线程的栈空间。

**3.堆和栈的区别**

堆：是进程中每个线程共有的空间，分全局堆和局部堆。全局堆就是所有没有分配的空间，局部堆就是用户分配的空间。堆在操作系统对进程初始化的时候分配，运行过程中也可以向系统要额外的堆，但是记得用完了要还给操作系统，要不然就是内存泄漏。

栈：是每个线程独有的，保存其运行状态和局部自动变量的。栈在线程开始的时候初始化，每个线程的栈互相独立，因此，栈是thread safe的。每个Ｃ＋＋对象的数据成员也存在在栈中，每个函数都有自己的栈，栈被用来在函数之间传递参数。操作系统在切换线程的时候会自动的切换栈，就是切换ＳＳ／ＥＳＰ寄存器。栈空间不需要在高级语言里面显式的分配和释放。

1. **一个程序从开始运行到结束的完整过程（四个过程）**

**预编译->编译->汇编->链接，具体过程详见深入理解计算机系统和自我修养**

**5.**