1、**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.

**2、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.

3、**线程和进程的关系：**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.

　　相对进程而言，线程是一个更加接近于执行体的概念，它可以与同进程中的其他线程共享数据，但拥有自己的栈空间，拥有独立的执行序列。

4、**线程和进程的区别：**

**（1）**线程是进程的一个实体，一个进程可以有多个线程，多个线程也可以并发执行，一个没有线程的进程也可以看做是单线程的，同样线程也经常被看做是一种轻量级的进程。并且进程就可以不依赖于行线程而单独存在，而线程则不然。

**（2）**进程是并发程序在一个数据集合上的一次执行过程，进程是系统进行资源分配和调度的独立单位，线程是进程的实体，他是比进程更小的能够独立运行的基本单元，线程自己不拥有任何系统资源，但是它可以访问进程的全部资源。

**（3）**与进程的控制表PCB相似，线程也有自己的控制表TCB，但是TCB中所保存的线程状态比PCB小的多。

**进程和线程的主要差别**在于它们是不同的操作系统资源管理方式。进程有独立的地址空间，一个进程崩溃后，在保护模式下不会对其它进程产生影响，而线程只是一个进程中的不同执行路径。线程有自己的堆栈和局部变量，但线程之间没有单独的地址空间，一个线程死掉就等于整个进程死掉，所以多进程的程序要比多线程的程序健壮，但在进程切换时，耗费资源较大，效率要差一些。但对于一些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。

简单来记：

1) 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.

2) 线程的划分尺度小于进程，使得多线程程序的并发性高。

3) 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。

4) 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。

5) 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。



5、**进程的作用和意义：**为了提高CPU的执行效率，为了避免因等待而造成CPU空转以及其他计算机资源的浪费而提出的。

6、**线程的引入：**为了减少进程切换和创建的开销，提高执行效率和节省资源。

7、**Windows下的内存是如何管理的？**

（1）虚拟内存： 最适合用来管理大型对象或者结构数组

（2）内存映射文件： 最适合用来管理大型数据流（通常来自文件）以及在单个计算机上运行多个进程之间共享数据

（3）内存堆栈： 最适合用来管理大量的小对象

**8、中断和轮询的特点：**

对I/O设备的程序轮询的方式，是早期的计算机系统对I/O设备的一种管理方式。它定时对各种设备轮流询问一遍有无处理要求。轮流询问之后，有要求的就加以处理。在处理I/O设备的要求之后，处理机返回继续工作。尽管轮询需要时间，但轮询要比I/O设备的速度要快得多，所以一般不会发生不能及时处理的问题。当然，再快的处理机，能处理的输入输出设备的数量也是有一定限度的。而且，程序轮询毕竟占据了CPU相当一部分处理时间，因此程序轮询是一种效率较低的方式，现代计算机系统中已很少应用。

　　轮询效率低，等待时间很长，CPU利用率不高；中断容易遗漏一些问题，CPU利用率高。

9、**什么是临界区、如何解决冲突？**

　　每个进程中访问临界资源的那段程序称为临界区，每次只准许一个进程进入临界区，进入后不允许其他进程进入。如果有若干个进程要求进入空闲的临界区，一次仅允许一个进程进入。任何时候，处于临界区的进程不可多于一个。如已有进程进入自己的临界区，则其他试图进入临界区的进程必须等待。进入临界区的进程要在有限时间内退出，以便其他进程能及时进入自己的临界区。如果不能进入自己的临界区，就应该让出CPU，避免进程出现忙等等现象。

10、**分段和分页的区别？**

　　页是信息的物理单位，分页是为了实现离散分配方式，以减少内存的外零头，提高内存的利用率。分页仅仅是由于系统管理的需要，而不是用户的需要。

　　段是信息的逻辑单位，它含有一组其意义相对完整的信息。分段的目的是为了能更好的满足用户的需要。

　　页的大小固定且由系统确定，把逻辑地址分为页号和页内地址两部分，由机器硬件实现的。因此一个系统只能有一种大小的页面。段的长度却不固定，决定于用户所编写的程序，通常由编写程序在对源代码进行编辑时，根据信息的性质来划分。

　　分页的作业地址空间是一维的，即单一的线性空间。

　　分段的作业地址空间是二维的，程序员在标识一个地址时，既需要给出段名，又需要给出段内地址。

11、**进程间通信有哪些方式？它们的区别？**

**IPC方式：7种**

1.**管道（pipe）**：管道是一种半双工的通信方式，数据只能单向流动，而且只能在有血缘关系的进程间使用，进程的血缘关系通常是指父子进程关系。

**2.命名管道**（named pipe）：也是半双工的通信方式，但是它允许无亲缘关系关系进程间通信。

**3.信号**（signal）：是一种比较复杂的通信方式，用于通知接收进程某一事件已经发生。

**4.信号量（**semophere）：信号量是一个计数器，可用来控制多个进程对共享资源的访问。它通常作为一种锁机制，防止某进程正在访问共享资源时，其他进程也访问该资源。因此，主要作为进程间以及同一进程内不同线程之间的同步手段。

**5.消息队列**（message queue）:消息队列是由消息组成的链表，存放在内核中，并由消息队列标识符标识。消息队列克服了信号传递消息少，管道只能承载无格式字节流以及缓冲区大小受限等缺点。

**6.共享内存**（shared memory）:就是映射一段能被其他进程所访问的内存，这段共享内存由一个进程创建，但多个进程都可以访问，共享内存是最快的IPC方式，它是针对其他进程间的通信方式运行效率低而专门设计的。它往往与其他通信机制，如信号量等配合使用，来实现进程间的同步和通信。

**7.套接字**（socket）：套接口也是进程间的通信机制，与其他通信机制不同的是它可用于不同及其间的进程通信。

**几种方式的比较：**

管道：速度慢、容量有限

消息队列：容量收到系统限制，且要注意第一次读的时候，要考虑上一次没有读完数据的问题。

信号量：不能传递复杂信息，只能用来同步。

共享内存：能够很容易控制容量，速度快，但要保持同步，比如一个进程在写的时候，另一个进程要注意读写的问题，相当于线程中的线程安全。

12、**线程间的通信机制**

**1.锁机制：互斥锁、条件变量、读写锁**

　互斥锁提供了以排他方式防止数据结构被并发修改的方法。

　读写锁允许多个线程同时读共享数据，而对写操作是互斥的。

　条件变量可以以原子的方式进行阻塞进程，直到某个特定条件为真为止。对条件的测试是在互斥锁的保护下进行的。条件变量始终与互斥锁一起使用。

**２.信号量机制：**包括无名信号量和命名线程信号量

**３.信号机制：**类似进程间的信号处理

线程间的通信目的主要是用于线程同步，所以线程没有像进程通信中的用于数据交换的通信机制。

**13、什么是死锁？产生条件？如何避免死锁**

**死锁的概念**：在2个或多个并发进程中，如果每个进程持有某有资源而又都等待别的进程释放它或他们现在保持的资源，在未改变这种状态之前都不能向前推进，称这一组进程产生了死锁。通俗地讲，就是2个或多个进程被无限期地阻塞、相互等待的一种状态。

**死锁产生的原因**：系统资源不足，进程推进顺序非法

**产生死锁的必要条件**：

**1.互斥条件**：一个资源每次只能被一个进程使用

**2.不可剥夺条件：**进程已获得资源，在未使用完之前，不能被其他进程强行剥夺，只能主动释放

**3.请求和保持条件：**进程已经保持了至少一个资源，但又提出了新的资源请求，而该资源已被其他进程占有，此时请求进程被阻塞，但对自己已获得的资源保持不放。

**4.循环等待条件：**即进程集合{p0,p1,p2,p3…..pn};p0正在等待p1占用的资源，p1正在等待p2占用的资源，pn正在等待p0占用的资源。

　只要上述一个条件不成立，就不会发生死锁。

**死锁的解除和预防**：理解了死锁的原因，以及产生死锁的四个必要条件，就可以最大可能地避免和预防和解锁死锁。所以在系统设计、进程调度等方面注意如何不让这四个必要条件成立，如何确定资源的合理分配算法，避免进程永久占据系统资源。对资源的分配要给予合理规划

**死锁的处理策略**：鸵鸟策略、预防策略、避免策略、检测与解除死锁

**14、进程间同步与互斥的区别，线程同步的方式？**

**互斥：**指某一个资源同时只允许一个访问者对其进行访问，具有唯一性和排它性。但互斥无法限制访问者对资源的访问顺序，即访问是无序的

同步：是指在互斥的基础上（大多数情况下），通过其它机制实现访问者对资源的有序访问。大多数情况下，同步已经实现了互斥，特别是所有写入资源的情况必定是互斥的。少数情况是指可以允许多个访问者同时访问资源。

同步：体现的是一种协作性。互斥：体现的是排它性。

**进程同步的主要任务：**是对多个相关进程在执行次序上进行协调，以使并发执行的诸进程之间能有效地共享资源和相互合作。从而使程序的执行具有可再现性。

同步机制遵循的原则：

　1.空闲让进；

　2.忙则等待；

　3.有限等待；

　4.让权等待；

**线程同步**是指多个线程同时访问某资源时，采用一系列的机制以保证最多只能一个线程访问该资源。线程同步是多线程中必须考虑和解决的问题，以为很有可能发生多个线程同时访问（主要是写操作）同一资源，如果不进行线程同步，很可能会引起数据混乱，造成线程死锁等问题。

**线程同步的方式：**

　　临界区、互斥量、信号量、事件

　　临界区：通过对多线程的串行化来访问公共资源或者一段代码，速度快，适合控制数据访问。

　　互斥量：采用互斥对象机制，只有拥有互斥对象的线程才有访问公共资源的权限，因为互斥对象只有一个，所以可以保证公共资源不会同时被多个线程访问。

　　信号量：它允许多个线程同一时刻访问同一资源，但是需要限制同一时刻访问此资源的最大线程数目。信号量对象与其他前面几种方法不同，信号允许多个线程同时使用共享资源。

　　事件（信号）：通过通知操作的方式来保持多线程的同步，还可以方便实现多线程的优先级比较操作。



**线程同步不同方式间的总结比较：**

　　互斥量和临界区很相似，但是互斥量是可以命名的，它可以跨越进程使用，所以创建互斥量所需要的资源更多，如果只是为了在进程内部使用使用临界区会带来速度上的优势并能够减少资源占用量。

　　互斥量、信号量、事件都可以被跨越进程使用来进行同步数据操作，而其他的对象与数据同步操作无关，但对于进程和线程来讲，如果进程和线程在运行状态则为无信号状态，所以可以使用WaitForSingleObject来等待进程和线程退出。

**15、进程的调度算法有哪些？**

　　1.**先来先服务（FCFS）**:此算法的原则是按照作业到达后备作业队列（或进程进入就绪队列）的先后次序选择作业（或进程）

　　2.**短作业优先**（SJF:Shortest Process First）：这种算法主要用于作业调度，它从作业后备序列中挑选所需运行时间最短的作业进入主存运行。

　　3.**时间片轮转调度算法**：当某个进程执行的时间片用完时，调度程序便终止该进程的执行，并将它送到就绪队列的末尾，等待分配下一时间片再执行。然后把处理机分配给就绪队列中新的队首进程，同时也让它执行一个时间片。这样就可以保证队列中的所有进程，在已给定的时间内，均能获得一时间片处理机执行时间。

　　4.**高响应比优先**：按照高响应比（已等待时间+要求运行时间）/要求运行时间 优先的原则，在每次选择作业投入运行时，先计算此时后备作业队列中每个作业的响应比RP。选择最大的作业投入运行。

　　5.**优先权调度算法**：按照进程的优先权大小来调度。使高优先权进程得到优先处理的调度策略称为优先权调度算法。注意：优先数越多，优先权越小。

　　6.**多级队列调度算法：**多队列调度是根据作业的性质和类型的不同，将就绪队列再分为若干个队列，所有的作业（进程）按其性质排入相应的队列中，而不同的就绪队列采用不同的调度算法。