1. **什么是虚拟内存？**

**首先，我们知道每一个进程都有它自己独立的地址空间，然后这些空间呢，通过分页技术又被划分成多个大小相等的页，但是呢，并不是所有的页都要加载到物理内存中去才能够运行程序，另一方面也是由于物理内存的不足的原因，现在物理内存普遍才4G,8G,全部加载进去肯定是不够用的，因此通过虚拟内存技术，让不需要的页暂时存放在硬盘时。当程序请求对应的地址空间的时候，会由操作系统进行映射，如果映射的物理内存不存在，就会去硬盘中读取并重新执行失败的指令。这样，在逻辑上程序似乎有很大的逻辑空间，实际上只对应了一部分的物理内存，还有一部分在硬盘上。**

**因此呢，虚拟内存技术主要适用于大的对象和数据，好处就是解决了内存不足，减少了内存的占用。坏处就是频繁的硬盘读取，可能会降低系统的速度。**

1. 分段和分页的区别？

段式存储管理：将程序分为若干段，比如说数据段，代码段，堆栈段等。他的好处就是没有内碎片，因为**段的大小的可以变化**的，你需要多少就分配多少。坏处就是有外碎片。

页式存储管理：将程序的逻辑地址分为大**小相等的页**，物理地址划分为大小相等的块，由**页表建立起页号到物理块号的映射**。在程序加载的时候，可以将任一页放入任一帧中，这些帧不必连续，从而实现了离散分离。优点就是没有外碎片，但是有内碎片，因为分配的大小不变，但是程序可能用不完。

不同点：

1. 内存碎片不同，段式产生外碎片，页式残生内碎片
2. 分段是满足用户的需要，由用户进行划分，分页是满足系统的需要，由操作系统划分，更加方便管理
3. 大小不同，段是根据数据动态增长的，而页的大小只有一种，是不会变化的。
4. 地址不同，页的地址空间是一维的，而段的地址空间是二维的（段号，段内偏移）。
5. 页面置换算法

在虚拟内存技术中，当硬盘上的数据加载到内存中时，如果物理内存不足，则需要进行页面置换，将不需要的页换出来，从而能将数据加载到内存中去。常见的页面置换算法有：

* + - 1. FIFO（先进先出）:把在页面中停留最久的置换出去。
      2. LRU（最近最久未使用）：把在最近且最久没使用的置换出去
      3. OPT（最佳置换算法）：选择未来长时间不被访问或者永远不会被访问的置换出去。是一种理想的算法，无法实现，因为无法进行预知。

1. 逻辑地址，物理地址，虚拟地址

逻辑地址：是程序中的段地址。由两部份组成，段标识符和段内[偏移量](https://baike.baidu.com/item/%E5%81%8F%E7%A7%BB%E9%87%8F)。

物理地址：实际的内存的地址。

虚拟地址。虚拟内存管理下的一个地址。

1. 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

1. 内碎片和外碎片

内碎片是已经被分配出去，但是没有被充分利用的空间，就是分配多了。

外碎片是没有被分配出去，但是由于太小了，无法分配给申请内存空间的进程的空闲内存。

1. 中断和轮询的特点
2. 进程调度算法
3. FCFS:先来先服务，按照作业进入就绪队列的次序进行调度
4. SJF：短作业优先，运行时间最短的进入
5. 时间片轮转调度算法：给每个队首的进程分配一个时间片，用完了就等待下一次时间片再进行。
6. 高响应比优先：响应比：（已等待时间+作业时间）/要求运行时间。
7. 优先权调度：根据进程的优先权来进行调度。
8. 多级队列调度算法：根据作业的性质不同，分成多个就绪队列，不同的队列使用不同的算法。
9. 什么是临界区、如何解决冲突？

临界区：**每个进程访问临界资源的那段代码**。每次只能有一个进程进入临界区。如果已经有进程进去临界区，其他尝试进入临界区的进程必须等待。已经进入的进程要在有限的时间内退出，已让其他进程能够进入自己的临界区。

临界资源：**一次只允许一个进程访问的资源**。比如变量，数组，缓冲区等。

1. 进程有哪几种状态

就绪状态：进程已经获得CPU以外的资源，等待CPU分配资源

运行状态：占用CPU资源运行

阻塞状态：进程等待某种条件，在满足条件之前无法运行

1. 进程程间的通信机制
   1. 管道：
   2. 信号
   3. 消息队列
   4. 共享内存
   5. 信号量
   6. 套接字
2. 线程同步(通信)的方式
   1. 互斥量（锁机制）：只有拥有锁的线程才能访问资源，一个资源只能有一个线程进行访问。
   2. 信号量：同一时刻有多个线程进行访问，但是有信号量控制着最大的访问数量
   3. 事件（wait/notify）:等待通知机制。
3. 用户态，内核态

在进程在执行运行自己代码的时候，就处于用户态，进程在大部分的情况下都是处于用户态的，但当需要操作系统帮助的时候，就需要切换到内核态。他们之间最大的区别就是权限不一样，用户态特权等级为3级，最低，内核态用户等级为0级，最高。有3种方式会使用户态切换到内核态：

1. 系统调用：比如文件操作，网络数据的发送等。
2. 异常：当CPU执行时发生了比如缺页异常等会切换到异常的内核相关进程种。
3. 外围设备中断：
4. 僵尸进程和孤儿进程

僵尸进程：即子进程先于父进程退出后，子进程的PCB需要其父进程释放，但是父进程并没有释放子进程的PCB。僵尸进程实际上是一个已经死掉的进程。

一个父进程退出，而它的一个或多个子进程还在运行，那么那些子进程将成为孤儿进程。孤儿进程将被init进程(进程号为1)所收养，并由init进程对它们完成状态收集工作。

危害：孤儿进程有init(1号进程)为他善后，因此是没有太大危害的。但是僵尸进程虽然实体已经没有了，但是还是在系统中占用着记录，这是对资源的一个浪费。

僵尸进程处理：杀死父进程，让他变为孤儿进程；重启系统；通过信号机制，让子进程退出时通知父进程，然后让父进程调用wait()方法清理子进程的信息。