linux常用命令（问的时候都会给具体某一个场景）

Linux内核相关（select、poll、epoll）

1. 进程和线程以及它们的区别

2. 线程同步的方式

3. 进程的通信方式

4. 进程同步有哪几种机制。

5. 什么是缓冲区溢出？有什么危害？其原因是什么？

6. 什么是死锁？死锁产生的条件？

7. 说一说死锁的处理基本策略和常用方法。

8. 进程有哪几种状态？

9. 分页和分段有什么区别？

10. 操作系统中进程调度策略有哪几种？

11. 内存之页面置换算法

12. 孤儿进程和僵尸进程

13. 父子进程

14. 什么是虚拟内存

15. 虚拟地址、逻辑地址、线性地址和物理地址的区别

1. 请分别简单说一说进程和线程以及它们的区别。

(1) 进程是具有一定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调度和分配的一个独立单位。

(2) 线程是进程的实体，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。

(3) 一个进程可以有多个线程，多个线程也可以并发执行。

2. 线程同步的方式有哪些？

(1) 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被多个线程同时访问。

(2) 信号量：它允许同一时刻多个线程访问同一资源，但是需要控制同一时刻访问此资源的最大线程数量。

(3) 事件（信号）：通过通知操作的方式来保持多线程同步，还可以方便的实现多线程优先级的比较操作。

3. 进程间通信方式有哪些？

(1) 主要分为：管道、系统IPC（包括消息队列、信号量、共享存储）、SOCKET

(2) 管道主要分为：普通管道PIPE 、流管道（s\_pipe）、命名管道（name\_pipe）

① 管道是一种半双工的通信方式，数据只能单项流动，并且只能在具有亲缘关系的进程间流动，进程的亲缘关系通常是父子进程

② 命名管道也是半双工的通信方式，它允许无亲缘关系的进程间进行通信

③ 信号量是一个计数器，用来控制多个进程对资源的访问，它通常作为一种锁机制。

④ 消息队列是消息的链表，存放在内核中并由消息队列标识符标识。

⑤ 信号是一种比较复杂的通信方式，用于通知接收进程某个事件已经发生。

⑥ 共享内存就是映射一段能被其它进程访问的内存，这段共享内存由一个进程创建，但是多个进程可以访问。

⑦ 套接字( socket ) ： 套解口也是一种进程间通信机制，与其他通信机制不同的是，它可用于不同主机间的进程通信。

4. 说一说进程同步有哪几种机制。

(1) 原子操作、信号量机制

PV操作能够实现对临界区的管理要求；实现简单；允许使用它的代码休眠，持有锁的时间可相对较长。一个信号量只能置一次初值，以后只能对之进行p操作或v操作。

(2) 自旋锁

旋锁是为了保护共享资源提出的一种锁机制； 调用者申请的资源如果被占用，即自旋锁被已经被别的执行单元保持，则调用者一直循环在那里看是否该自旋锁的保持着已经释放了锁； 低开销；安全和高效；但是可能会引起死锁和过多占用CPU。

(3) 管程

集中式同步进程——管程。其基本思想是将共享变量和对它们的操作集中在一个模块中，操作系统或并发程序就由这样的模块构成。这样模块之间联系清晰，便于维护和修改，易于保证正确性。

(4) 会合：进程直接进行相互作用

(5) 分布式系统：消息和rpc。由于在分布式操作系统中没有公共内存，因此参数全为值参，而且不可为指针。

5. 什么是缓冲区溢出？有什么危害？其原因是什么？

(1) 缓冲区溢出是指当计算机向缓冲区填充数据时超出了缓冲区本身的容量，溢出的数据覆盖在合法数据上。

(2) 危害有以下两点：

① 程序崩溃，导致拒绝服务

② 跳转并且执行一段恶意代码

(3) 造成缓冲区溢出的主要原因是程序中没有仔细检查用户输入。

6. 什么是死锁？死锁产生的条件？

在两个或者多个并发进程中，如果每个进程持有某种资源而又等待其它进程释放它或它们现在保持着的资源，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。通俗的讲就是两个或多个进程无限期的阻塞、相互等待的一种状态。

(1) 死锁产生的四个条件（有一个条件不成立，则不会产生死锁）（重要）

① 互斥条件：一个资源一次只能被一个进程使用

② 请求与保持条件：一个进程因请求资源而阻塞时，对已获得资源保持不放

③ 不剥夺条件：进程获得的资源，在未完全使用完之前，不能强行剥夺

④ 循环等待条件：若干进程之间形成一种头尾相接的环形等待资源关系

7. 说一说死锁的处理基本策略和常用方法。

(1) 解决死锁的基本方法如下：

预防死锁、避免死锁、检测死锁、解除死锁

(2) 解决死锁的常用策略如下：

鸵鸟策略、预防策略、避免策略、检测与解除死锁

8. 进程有哪几种状态？（）

(1) 就绪（可运行）状态：进程已获得除处理机以外的所需资源，等待分配处理机资源

(2) 运行状态：占用处理机资源运行，处于此状态的进程数小于等于CPU数

(3) 阻塞状态： 进程等待某种条件，在条件满足之前无法执行

9. 分页和分段有什么区别？

① 段是信息的逻辑单位，它是根据用户的需要划分的，因此段对用户是可见的 ；页是信息的物理单位，是为了管理主存的方便而划分的，对用户是透明的。

② 段的大小不固定，有它所完成的功能决定；页的大小固定，由系统决定

③ 段向用户提供二维地址空间；页向用户提供的是一维地址空间

④ 段是信息的逻辑单位，便于存储保护和信息的共享，页的保护和共享受到限制。

(1) 分页管理的基本思想

用户程序的地址空间被划分成若干固定大小的区域，称为“页”，相应地，内存空间分成若干个物理块，页和块的大小相等。可将用户程序的任一页放在内存的任一块中，实现了离散分配。

(2) 分段管理的基本思想

① 段是按照程序的自然分界划分的，长度可以动态改变。通常，程序员把子程序、操作数和常数等不同类型的数据划分到不同的段中（写c程序时会用到），并且每个程序可以有多个相同类型的段。

② 段表本身也是一个段，可以存在辅存中，但一般是驻留在主存中。将用户程序地址空间分成若干个大小不等的段，每段可以定义一组相对完整的逻辑信息。存储分配时，以段为单位，段与段在内存中可以不相邻接，也实现了离散分配。

10. 操作系统中进程调度策略有哪几种？

FCFS(先来先服务)，优先级，时间片轮转，多级反馈

(1) 抢占式中断和非抢占式中断

非抢占式中断时让一个进程一直运行直至堵塞或者自动释放CPU，该进程不会被强制挂起。其结果是在中断产生时，如果没有更高优先级的进程等待到时，该进程会一直运行下去。相反，抢占式中断会挑选一个进程，并让该进程运行某一个固定的最大值，如果在该时间段内进程仍然在运行，则该进程被挂起，调度程序选择另一个进程进行执行。

(2) FCFS(先来先服务)

先来先服务：最简单的非抢占式算法，基本上维持一个就绪进程的单一队列，有新作业到来时会插入到队列尾部。每次系统调用时从队列头部取作业进行运行，指导该作业完成。

(3) 优先级调度

优先级调度：轮转调度隐含的假设是所有进程同等重要，也就是所有进程的优先级相同。然后对于操作系统和用户会认为某些进程要比其他进程更为重要。因此调度算法调度时需要考虑一些外部因素。优先级调度的基本思想是为每一个进程赋予一个优先级，允许最高优先级的可运行程先运行。

(4) 时间片轮转

轮转调度：轮转调度是一种最古老，最简单且最公平使用最广的算法。它为每一个进程分配一个时间段成为时间片。如果在该时间片结束时该进程还在运行，则将剥夺CPU并分配给另一个进程。如果该进程在时间片内堵塞或者结束，则CPU立即切换。时间片调度容易实现，调度程序只需要维护一张可运行程序的进程列表即可。

(5) 多级反馈（均衡调度）

多级队列：多级队列的思想是将进程分为几个优先级类，当一个进程用完所分配的时间片后将会被移到更高优先级。我们考虑一个需要100个时间片来进行连续计算的计算的进程。假如它最初被分配1个时间片，则时间片用完之后将其移到下一类，同时它将获得2个时间片，依次类推，接下来其将获得8、16、32和64个时间片。这样其通过7次进程切换就可以完成，如果是纯粹的时间片轮转算法则需要100次此进程切换。并且随着优先级的不断下降，其可以将CPU让给运行时间较短的进程。

11.内存的页面置换算法

(1) 分页式内存基本介绍

① 为提高内存利用率，创造了分页式内存抽象。同时有一个虚拟内存的概念，是指将内存中暂时不需要的部分写入硬盘，看上去硬盘扩展了内存的容量，所以叫做“虚拟”内存。使用虚拟内存，应用程序可以使用比实际物理内存更大的内存空间。可以认为这个更大的内存空间就在硬盘上，只有将某一部分需要被用到时，才被写入真实内存；当它暂时不再被用到时，又被写回硬盘。

② 分页式内存管理将物理内存分为等大的小块，每块大小通常为1K、2K、4K等，称为页帧；逻辑内存（使用虚拟内存技术扩大的内存，可认为其位于硬盘上）也被分为等大的小块，称为页；且页和页帧的大小一定是一样的，是写入真实内存和写回硬盘最小单位。

(2) 置换算法介绍

① 假设某一时刻内存页帧已经被写满了，但这时又需要将一个页写到物理内存中，就需要将原本在物理内存中的某一页换出来。如果置换不当，就会导致刚刚被换出到硬盘的页又要被写回内存，减慢系统运行的速度。页面置换算法就是考虑将哪一页换出来以获得优良性能的方法。

11.1 Optimal算法（最优算法）

最优算法，是需要知道以后要被用到的页，然后将不会被用到的页换出内存；如果所有页都会被用到，就把需要使用时间离现在最长的页换出，以尽量使不好的情况晚发生。这种方法能使系统获得最佳性能，但是，它是不可能实现的......因为当前无法获知以后哪些页要被用到。不过最优算法还是能够作为其他算法优秀程度的衡量。

11.2 FIFO（First-In First-Out，先进先出）算法

置换出当前已经待在内存里时间最长的那个页。FIFO算法的运行速度很快，不需要考虑其他的因素，需要的开销很少。但是正是由于没有考虑页面的重要性的问题，FIFO算法很容易将重要的页换出内存。

11.3 Second Chance（第二次机会）算法

为了避免FIFO算法将重要的页换出内存，Second Chance算法提供了一些改进。Second Chance算法在将页面换出内存前检查其使用位（使用位前文有介绍），如果其使用位为1，证明此页最近有被使用，猜测它还可能被使用，于是不把它置换出内存，但是把其使用位置为0，随后检查下一个页面，直到发现某页的使用位为0，将此页置换出内存。

11.4 Clock算法（时钟轮转法）

为了节约Second Chance算法一个接着一个检查使用位的开销，时钟轮转法又提出了改进。时钟轮转法将所有的页组成一个圆，圆心的指针指向下一个要被置换的页面，置换前同样检查使用位，如果使用位为1，同样将其使用位置为0，随后将顺指针旋转，检查下一个页面，直到发现某页的使用位为0，将此页置换出内存。

11.5 LRU（Least Recent Used, 最近最少使用）算法

为获得对最优算法的模拟，提出了LRU算法。由于当前时间之后需要用到哪些页无法提前获知，于是记录当前时间之前页面的使用情况，认为之前使用过的页面以后还会被用到。在置换时，将最近使用最少的页面换出内存。此种方法的开销比较大。

11.6 NRU（Not Recent Used， 最近未使用）算法

(1) 前面提到修改位和使用位，NRU算法利用这两个标志位将所有页帧分为4组：

　　第0组：修改位和使用位都为0；

　　第1组：修改位为0，使用位为1；

　　第2组：修改位为1，使用位为0；

　　第3组：修改位和使用位都为1。

(2) NRU算法从组数最小的一组（第0组）中随机选择一个页面将其移出内存。

　　NRU算法不是最好的，但是它使用起来开销很小，用较小的代价就得到了不错的效果，不失为一种不错的算法。

12. 孤儿进程和僵尸进程

(1) 僵尸进程：一个子进程在其父进程还没有调用wait()或waitpid()的情况下退出。这个子进程就是僵尸进程。任何一个子进程(init除外)在exit()之后，并非马上就消失掉，而是留下一个称为僵尸进程(Zombie)的数据结构，等待父进程处理。这是每个子进程在结束时都要经过的阶段。如果子进程在exit()之后，父进程没有来得及处理，那么保留的那段信息就不会释放，其进程号就会一直被占用，但是系统所能使用的进程号是有限的，如果大量的产生僵尸进程，将因为没有可用的进程号而导致系统不能产生新的进程. 此即为僵尸进程的危害，应当避免。

(2) 孤儿进程：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。孤儿进程是没有父进程的进程，孤儿进程这个重任就落到了init进程身上，init进程就好像是一个民政局，专门负责处理孤儿进程的善后工作。每当出现一个孤儿进程的时候，内核就把孤 儿进程的父进程设置为init，而init进程会循环地wait()它的已经退出的子进程。这样，当一个孤儿进程凄凉地结束了其生命周期的时候，init进程就会代表党和政府出面处理它的一切善后工作。因此孤儿进程并不会有什么危害。

13. 父子进程

(1) 一个进程通过fork（）函数创建一个新的进程，两者关系为父子关系，子进程拥有和父进程几乎但又不完全一样。两者拥有相同但又相互独立的地址空间，是可以并发运行的独立进程，子进程共享父进程的正文段和已经打开的文件，两者最大的差别就是进程ID不一样。

14. 什么是虚拟内存

为提高内存利用率，创造了分页式内存抽象。同时有一个虚拟内存的概念，是指将内存中暂时不需要的部分写入硬盘，看上去硬盘扩展了内存的容量，所以叫做“虚拟”内存。使用虚拟内存，应用程序可以使 用比实际物理内存更大的内存空间。可以认为这个更大的内存空间就在硬盘上，只有将某一部分需要被用到时，才被写入真实内存；当它暂时不再被用到时，又被写回硬盘。

15. 虚拟地址、逻辑地址、线性地址和物理地址的区别

(1) 物理地址：放在寻址总线上的地址，用于内存芯片级内存单元寻址。放在寻址总线上，如果是读，电路根据这个地址每位的值就将相应地址的物理内存中的数据放到数据总线中传输。如果是写，电路根据这个地址每位的值就将相应地址的物理内存中放入数据总线上的内容。物理内存是以字节(8位)为单位编址的，是地址变换的最终结果地址，物理地址由32位或36位无符号整数表示。

(2) 逻辑地址：是指由程序产生的与段相关的偏移地址部分，每一个逻辑地址都由一个段和偏移量组成。在进行C语言指针编程中，可以读取指针变量本身值(&操作)，实际上这个值就是逻辑地址，它是相对于你当前进程数据段的地址，不和绝对物理地址相干。只有在Intel实模式下，逻辑地址才和物理地址相等（因为实模式没有分段或分页机制,Cpu不进行自动地址转换）；逻辑也就是在Intel 保护模式下程序执行代码段限长内的偏移地址（假定代码段、数据段如果完全一样）。

(3) 线性地址：是逻辑地址到物理地址变换之间的中间层。程序代码会产生逻辑地址，或者说是段中的偏移地址，加上相应段的基地址就生成了一个线性地址，是一个32位无符号整数，可以用来表示高达4GB的地址，也就是说，高达4294967296个内存单元，以十六进制表示，0x00000000到oxffffffff。如果启用了分页机制，那么线性地址可以再经变换以产生一个物理地址。若没有启用分页机制，那么线性地址直接就是物理地址。Intel 80386的线性地址空间容量为4G（2的32次方即32根地址总线寻址）。

(4) 虚拟内存：现在操作系统中，都使用了MMU的存储管理技术，而MMU管理的地址是虚拟地址，虚拟内存是计算机系统内存管理的一种技术。它使得应用程序认为它拥有连续的可用的内存（一个连续完整的地址空间），而实际上，它通常是被分隔成多个物理内存碎片，还有部分暂时存储在外部磁盘存储器上，在需要时进行数据交换。与没有使用虚拟内存技术的系统相比，使用这种技术的系统使得大型程序的编写变得更容易，对真正的物理内存（例如RAM）的使用也更有效率。有时我们也把逻辑地址称为虚拟地址。因为和虚拟内存空间的概念类似，逻辑地址也是和实际物理内存容量无关的。