操作系统知识点整理

**一、操作系统概念**

1. 内核态与用户态的区别？

* 处于用户态执行时，进程所能访问的内存空间和对象受到限制，其所占有的处理器是可被抢占的。
* 处于内核态执行时，则能访问所有的内存空间和对象，且所占有的处理器是不允许被抢占的。
* 用户态下运行用户程序，而内核态下运行操作系统程序、操作硬件。

1. 什么情况下会导致用户态到内核态的切换？

* 系统调用
* 异常（如缺页异常）
* 外围设备中断

**二、进程管理**

1. 进程与线程的区别？
2. 线程之间哪些是独立的，哪些是共享的？
3. 进程由哪些部分组成？
4. 进程的状态转换图？
5. 进程在运行的过程中可能会发生哪些情况？
6. 上下文切换的过程？
7. Fork()与Exec()的区别？

* 进程调用Fork()函数时，子进程（逻辑意义上地）复制了父进程的数据段和堆栈段，但它们之间共用同一个代码段。逻辑意义指实际执行Fork()时，物理空间上两个进程的数据段和堆栈段都还是共享的，当有一个进程写了某个数据时，这时两个进程之间的数据才有了区别（也称写时复制）。
* 进程一旦调用Exec()函数，它本身就已经死亡了，系统把代码段替换成新的程序的代码，废弃原有的数据段和堆栈段，并为新程序分配新的数据段与堆栈段，唯一留下的，就是进程号。
* 调用Fork()时，只有当前线程会在子进程中保留。
* 调用Exec()时，除了当前线程，其他全部都会消失。

1. IPC(Interprocess Communication)的5种方式？

* 共享内存
* 消息队列
* 信号量
* 管道
* 套接字

1. 进程调度的几种方法？

* 先来先服务（FCFS）：有利于长进程（或CPU Bound），而不利于短进程（或I/O Bound），平均等待时间往往很长。
* 最短作业优先（SJF）：对预计执行时间短的作业（进程）优先分派处理器，是最优的。最短剩余时间优先（SRTF）是基于抢占的SJF算法。非抢占式SJF会导致饥饿现象。
* 优先级调度（Priority Scheduling）：存在无穷阻塞或饥饿的问题，但是高响应比优先不会导致饥饿。
* 时间片轮转（RR）：就绪进程按照FCFS原则，排成队列，每次调度时将CPU分派给队首进程，让其执行一个时间片（time slice）。其不可能导致饥饿现象。
* 多级队列调度：每个作业归入一个队列，不同队列可有不同的优先级、时间片长度、调度策略等。多级反馈队列算法允许进程在队列中迁移，把占用CPU多的进程移动到低优先级的队列，能够阻止饥饿的发生。

1. 进程同步与进程通信的区别？同步与互斥的区别？

* 进程同步：多个进程按一定顺序执行
* 进程通信：进程间传递信息
* 互斥：两个进程不能够同时访问临界资源

1. 临界区解决方案需要满足哪些条件？

* 互斥
* 空闲让进
* 有限等待

1. 什么是死锁？死锁的必要条件？
2. 死锁预防策略和死锁避免策略？
3. 如何打破死锁？

**三、内存管理**

1. 线性地址，逻辑地址和物理地址的区别？

* 逻辑地址：程序在编译或汇编后，每个目标代码（模块）都从0开始编址，该相对地址称为逻辑地址。
* 线性地址：逻辑地址加上相应段的基地址得到线性地址，如果没有启动分页机制，则线性地址就是物理地址，如果启用了分页机制，还需要再变换一次才能够得到物理地址。
* 物理地址：内存中存储单元的地址。

1. 分段和分页的区别？

* 分页对程序员是透明的（由操作系统自动完成），而分段需要程序员显式划分。
* 分页的地址空间是一维的（只需要根据线性地址就能够得出页号和页偏移），而分段的地址空间是二维的（需要根据段号和段偏移才能够得出地址）。
* 页的大小不可改变而段的大小可以改变。
* 分段的出现主要是使程序和数据可以被划分为逻辑上独立的地址空间，有助于共享和保护；而分页主要是为了实现虚拟内存，获得更大的地址空间。

1. 常见的页面置换算法？

* FIFO：维护一个队列，每次替换最早进来的页，会出现Belady现象（页大小增大反而增加了缺页次数）
* OPT（最佳页面置换算法）：选择离当前最远位置上出现的页置换
* LRU（最近最少使用）：选择页面中最久没有引用的页面置换，需要通过链表实现，开销非常大
* 附加引用位算法：增加一个字节作为引用位，被访问时将最高位置1，定期右移并将最高位补0，寄存器数值最小的就是最久没有使用的位。
* CLOCK算法（二次机会算法）：使用环形链表将页面连起来，维护一个指针，当指针指向的页面引用位为1时，将其置0并寻找下一个页面，否则将其置换。
* 增强型CLOCK算法，使用引用位和修改位，按照(0, 0)，(0, 1)，(1, 0)，(1, 1)的顺序进行淘汰。

**四、文件管理**

1. 硬链接和符号链接的区别？

硬链接(hard link)：A是B的硬链接（A和B都是文件名），则A的目录项中的inode节点号与B的目录项中的inode节点号相同，即一个inode节点对应两个不同的文件名，两个文件名指向同一个文件，A和B对文件系统来说是完全平等的。如果删除了其中一个，对另外一个没有影响。每增加一个文件名，inode节点上的链接数增加一，每删除一个对应的文件名，inode节点上的链接数减一，直到为0，inode节点和对应的数据块被回收。注：文件和文件名是不同的东西，rm A删除的只是A这个文件名，而A对应的数据块（文件）只有在inode节点链接数减少为0的时候才会被系统回收。

软链接(soft link)：A是B的软链接（A和B都是文件名），A的目录项中的inode节点号与B的目录项中的inode节点号不相同，A和B指向的是两个不同的inode，继而指向两块不同的数据块。但是A的数据块中存放的只是B的路径名（可以根据这个找到B的目录项）。A和B之间是“主从”关系，如果B被删除了，A仍然存在（因为两个是不同的文件），但指向的是一个无效的链接。

1. 常见的磁盘调度算法有哪些？

* FCFS（不会饥饿）
* SSTF（最短寻道时间优先）：处理距离当前磁头位置最短寻道时间的请求
* SCAN：从磁臂的一端移动到另一端，到达另一端后反转
* C-SCAN：到达另一端后直接返回，不处理回程请求
* LOOK：只移动到一个方向上的最远请求为止