操作系统简答题

1.进程和线程的区别联系

联系：

（1）一个进程可以有多个线程，但至少有一个线程；而一个线程只能在一个进程的地址空间内活动。

（2）资源分配给进程，同一个进程的所有线程共享该进程所有资源。

（3）CPU分配给线程，即真正在处理器运行的是线程。

（4）线程在执行过程中需要协作同步，不同进程的线程间要利用消息通信的办法实现同步。

区别：

（1）调度：线程作为调度和分配的基本单位，进程作为拥有资源的基本单位

（2）并发性：不仅进程之间可以并发执行，同一个进程的多个线程之间也可并发执行

（3）拥有资源：进程是拥有资源的一个独立单位，线程不拥有系统资源，但可以访问隶属于进程的资源.

（4）系统开销：在创建或撤消进程时，由于系统都要为之分配和回收资源，导致系统的开销明显大于创建或撤消线程时的开销。

2.如果一个程序需要多个用户操作，使用多进程还是多线程？

（1）数据共享、同步

多进程：数据共享复杂，需要用IPC；数据是分开的，同步简单

多线程因为共享进程数据，数据共享简单，但也是因为这个原因导致同步复杂

各有优势

（2）内存、CPU

多进程：占用内存多，切换复杂，CPU利用率低

多线程：占用内存少，切换简单，CPU利用率高

线程占优

（3）创建销毁、切换

多进程：创建销毁、切换复杂，速度慢

多线程：创建销毁、切换简单，速度很快

线程占优

（4）编程、调试

多进程：编程简单，调试简单

多线程：编程复杂，调试复杂

进程占优

（5）可靠性

多进程：进程间不会互相影响

多线程：一个线程挂掉将导致整个进程挂掉

进程占优

（6）分布式

多进程：适应于多核、多机分布式；如果一台机器不够，扩展到多台机器比较简单

多线程：适应于多核分布式

进程占优

3.一个进程中能否既有内核级线程又有用户级线程？Java线程属于内核级线程还是用户级线程？

4.连续分配、链接分配、索引分配，优缺点

（1）连续分配

原理：创建文件时，分配一组连续的块；FAT（文档分配表）中每个文件只要一项，说明起始块和文件长度。对于顺序文件有利。

优点：1.简便。适用于一次性写入操作。2.支持顺序存取和随机存取，顺序存取速度快。3.所需的磁盘寻道次数和寻道时间最少。（因为空间的连续性，当访问下一个磁盘块时，一般无需移动磁头，当需要移动磁头时，只需要移动一个磁道。）

缺点：1.文件不能动态增长。（可能文件末尾处的空块已经分配给了别的文件。）2.不利于文件的插入和删除。3.外部碎片问题。（反复增删文件后，很难找到空间大小足够的连续块，需要进行紧缩。）4.在创建文件时需声明文件大小。

（2）链式分配

原理：一个文件的信息存放在若干个不连续的物理块中，各块之间通过指针连接，前一个物理块指向下一个物理块。fat中每个文件同样只需要一项，包括文件名、起始块号和最后块号。任何一个自由块都可以加入到链中。

优点：1.提高磁盘的空间利用率，不存在外部碎片问题。2.有利于文件的插入和删除。3.有利于文件的动态扩充。

缺点：1.存取速度慢，一般只适用于信息的顺序存取，不适于随机存取。2.查找某一块必须从头到尾沿着指针进行。3.可靠性问题，如指针出错。4.更多的寻道次数和寻道时间。5.链接指针占一定的空间，将多个块组成簇，按簇进行分配而不是按块进行分配。（增加了磁盘碎片）

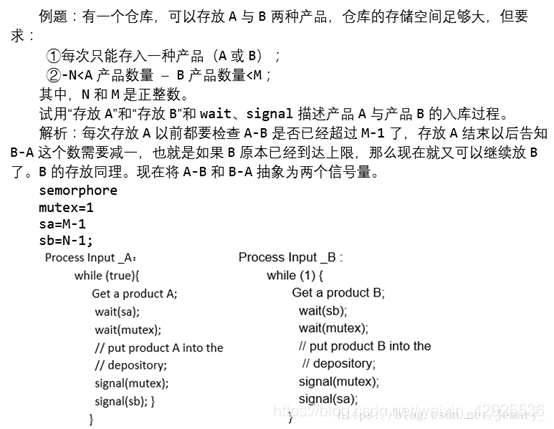
（3）索引分配

原理：每个文件在FAT中有一个一级索引，索引包含分配给文件的每个分区的入口。文件的索引保存在单独的一个块中，FAT中该文件的入口指向这一块。

优点：1.保持了链接结构的优点，又解决了其缺点：按快分配可以消除外部碎片。按大小可改变的分区分配可以提高局部性。索引分配支持顺序访问文件和直接访问文件，是普遍采用的一种方式。2.满足了文件动态增长，插入删除的要求。（只要有空闲块）3.能充分利用外存空间。

缺点：1.较多的寻道次数和寻道空间。2.索引表本身带来了系统开销，如：内外存空间、存取时间。

**5.**



内存保护

内存分配前保护操作系统不受用户进程的影响，同时保护用户进程不受其他用户进程影响。可采取以下两种方法：

1.在cpu中设置一对上下限寄存器和界地址寄存器，每当cpu要访问一个地址，分别和两个寄存器的值相比，判断有无越界。

2.采用重定位寄存器(或基址寄存器)和界地址寄存器(又称限长寄存器)来实现这种保护。重定位寄存器含最小的物理地址值，界地址寄存器含逻辑地址的最大值。内存管理机构动态地将逻辑地址与界地址寄存器进行比较，未发生越界则加上重定位寄存器的值后映射为物理地址，送交内存单元。

死锁，饥饿，阻塞的区别

死锁饥饿区别

相同点：二者都是由于竞争资源而引起的。

不同点：

(1)从进程状态考虑，死锁进程都处于等待状态，忙式等待(处于运行或就绪状态)的进程并非处于等待状态，但却可能被饿死；

(2)死锁进程等待永远不会被释放的资源，饿死进程等待会被释放但却不会分配给自己的资源，表现为等待时限没有上界(排队等待或忙式等待)；

(3)死锁一定发生了循环等待，而饿死则不然。这也表明通过资源分配图可以检测死锁存在与否，但却不能检测是否有进程饿死；

(4)死锁一定涉及多个进程，而饥饿或被饿死的进程可能只有一个。

饥饿和饿死与资源分配策略有关，因而防止饥饿与饿死可从公平性考虑，确保所有进程不被忽视，如FCFS分配算法。

死锁的预防和避免有什么区别

避免是操作系统对进程和进程之间的（对用户程序不加限制）。

预防是操作系统对用户程序限制的（限制其申请资源）。

----------------------------------------------------------------------------------------------------------------------