《操作系统》期末速通教程

6. 简答题

6.1 OS 概述

[**OS 的基本特征**]

(1) 并发: 两个或多个事件在同一时间间隔内发生.

(2) 共享.

(3) 虚拟: 将一个物理实体变为若干个逻辑对应物.

(4) 异步: 进程以不可预知的速度向前推进.

[**并发与并行的区别**] 同一时刻为并行, 同一时间间隔为并发.

[**多道批处理系统**]

(1) 优点:

① 资源利用率高, 如 CPU 、内存、I/O 设备等.

② 系统吞吐量大.

(2) 缺点:

① 进程的平均周转时间长.

② 无交互能力.

③ 用户响应时间长.

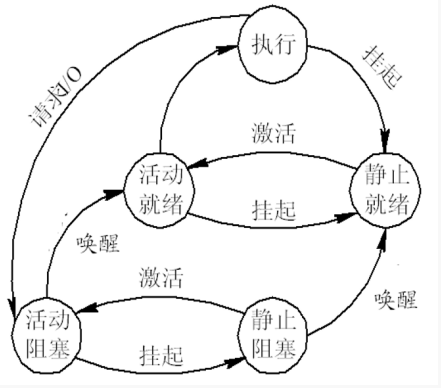
④ 系统开销大.

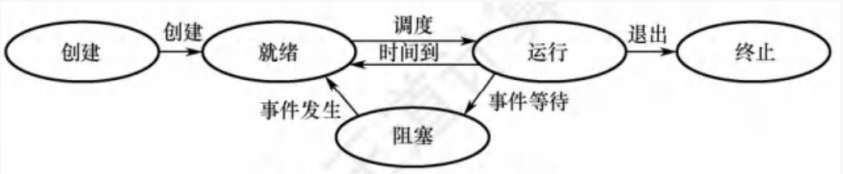
6.2 进程管理

6.2.1 进程概述

[进程的状态]

示意图:





6.2.2 进程控制

[进程创建过程, 创建原语]

(1) 为新进程分配一个唯一的进程标识号, 并申请一个空白的 PCB . 若 PCB 申请失败, 则进程创建失败.

(2) 为进程分配运行所需的资源, 如内存、文件、I/O设备、CPU 时间等. 这些资源或从 OS 获得, 或从父进程获得. 若资源不足, 则处于创建态, 等待内存资源.

(3) 初始化 PCB , 主要为初始化标志信息、CPU 状态信息、CPU 控制信息, 并设置进程的优先级等.

(4) 若进程就绪队列能接纳新进程, 则新进程入队, 等待调度.

6.2.3 线程

[线程相比于进程的优点]

(1) 进程的缺点:

① 实现并发的开销大.

② 系统并发度低.

(2) 线程的优点:

① 减少程序并发时的开销.

② 提高系统并发性.

③ 便于进程通信.

(3) 缺点: 降低进程安全性. 一个进程的各线程共享进程空间, 一个进程出错可能影响整个进程.

6.2.4 死锁

[**死锁的必要条件**]

(1) 互斥条件: 请求的资源是临界资源.

(2) 请求并保持条件: 持有旧资源, 申请新资源.

(3) 不可剥夺条件: 已获得的资源不可被外力剥夺, 只能主动释放.

(4) 循环等待条件.

6.2.5 处理机调度

[**三级调度**]

(1) 高级调度(作业调度).

(2) 中级调度(内存调度).

(3) 低级调度(进程调度).

[**先来先服务算法**, **First Come First Service**, **FCFS**]

(1) 优点:

① 公平.

② 实现简单.

③ 算法开销小.

④ 利于长作业.

⑤ 利于 CPU 繁忙型作业.

(2) 缺点:

① 不利于短作业.

② 不利于 I/O 繁忙型作业.

[**短作业优先算法**, **Shortest Job First**, **SJF**]

(1) 优点:

① 利于短作业.

② 平均等待时间、平均周转时间最短.

(2) 缺点:

① 不利于长作业, 增加周转时间.

② 长作业可能饥饿.

③ 不能保证实时性.

④ 作业的执行时间基于用户估算, 准确性不足.

[**优先级调度算法**] 缺点: 可能饥饿.

[**高响应比优先调度算法**]

(1) 优点:

① 兼顾长短作业:

(i) 等待时间相同时, 要求服务时间越短, 响应比越高, 利于短作业. 类似于 SJF .

(ii) 要求服务时间相同时, 等待时间越长, 响应比越高. 类似于 FCFS .

② 长作业的响应比随等待时间的增加而上升, 等待时间足够长时能保证获得 CPU , 不饥饿.

(2) 缺点: 每次调度前都需计算响应比, 系统开销大.

[**时间片轮转**, **Round Robin**, **RR**]

(1) 优点:

① 兼顾长短作业.

② 响应时间短.

(2) 缺点:

① 系统吞吐量和平均周转时间不如批处理系统.

② 上下文切换开销大.

[**多级反馈队列调度算法**]

(1) 优点:

① 兼顾长短作业.

② 响应时间短.

③ 可行性高.

(2) 缺点: 实现复杂.

6.2.6 进程同步与互斥

[**同步机制应遵循的规则**]

(1) 空闲让进.

(2) 忙则等待.

(3) 有限等待.

(4) 让权等待.

[**记录型信号量**]

(1) 整型信号量的缺点: 忙等, 未遵循 "让权等待" .

(2) 记录型信号量的优点: 不忙等, 遵循 "让权等待" .

6.3 内存管理

6.3.1 程序的装入与链接

[**程序进入内存的步骤**]

(1) 编译.

(2) 链接.

(3) 装入.

[**装入方式**]

(1) **绝对装入**的缺点:

① 只适用于单道程序环境.

② 需知道程序要放到内存中的物理地址, 编程不便.

(2) **可重定位装入** (**静态重定位**) 的缺点:

① 程序装入内存时需为其分配要求的全部内存空间, 若内存不足, 则无法装入.

② 程序进入内存后, 整个运行期间不可在内存中移动, 也不可再申请内存空间.

(3) **动态运行时装入** (**动态重定位**) 的优点:

① 可将程序分配到不连续的存储区.

② 程序运行前仅需装入部分代码即可投入运行, 程序运行期间可根据需要动态申请内存.

③ 便于共享程序段.

[**链接方式**]

(1) **静态链接**.

(2) **装入时动态链接**的优点:

① 便于程序的修改和更新.

② 便于共享目标模块.

(3) **运行时动态装入**的优点:

① 加速程序的装入过程.

② 节省内存空间.

6.3.2 连续分配

[单一连续分配]

(1) 优点:

① 实现简单.

② 无外部碎片.

③ 无需内存保护.

(2) 缺点:

① 只适用于单用户、单任务 OS .

② 有内部碎片.

③ 存储器利用效率极低.

[固定分区分配]

(1) 优点:

① 实现多道程序存储管理最简单.

② 无外部碎片.

(2) 缺点:

① 程序太大无法装入.

② 程序太小浪费空间, 即有内部碎片.

③ 无法实现多进程共享同一主存区.

④ 内存利用率低.

[**分配算法**]

(1) **首次适应**:

① 优点:

(i) 保留高地址的大空闲分区, 利于大作业装入.

(ii) 开销小.

(iii) 性能最好.

(iv) 回收分区时无需对空闲分区排序.

② 缺点:

(i) 低地址存在很多碎片.

(ii) 查找时间长.

(2) **循环首次适应** (**邻近适应**) :

① 优点: 内存低、高地址的空闲分区被等概率分配.

② 缺点: 高地址无大空闲分区可用.

(3) **最佳适应**:

① 优点:

(i) 内存利用率高.

(ii) 保留大空闲分区.

② 缺点:

(i) 性能很差.

(ii) 每次分配都会留下越来越小的难以利用的内存块, 产生外部碎片最多.

(4) **最坏适应**:

① 优点: 不容易产生外部碎片.

② 缺点:

(i) 性能很差.

(ii) 划分大空闲分区, 导致无大空闲分区可用.

6.3.3 基本分页存储

[**基本分页存储**]

(1) 优点:

① 内存中支持多道程序.

② 进程可离散分布在内存中.

(2) 缺点:

① 进程的最后一页装不满的部分产生内部碎片, 称**页内碎片**. 但碎片相对于进程很小.

② 每次访存都需进行逻辑地址到物理地址的转换.

③ 页表有内存开销.

④ 页表太大时, 内存利用率降低.

(3) 分页存储与固定分区分配的不同:

① 块大小比分区大小小得多.

② 进程按块划分, 运行时按块申请主存可用空间并运行.

③ 都会产生内部碎片, 但分页存储只在每个进程最后一个装不满的块产生页内碎片.

[**单级页表的缺点**] 页表很大时, 连续存储在内存不切实际.

[**多级页表**]

(1) 优点: 减少页表所占的连续内存空间.

(2) 缺点:

① 地址变换速度慢.

② 缺页中断次数多.

6.3.4 基本分段存储

[**基本分段存储**]

(1) 优点:

① 方便编程.

② 便于信息保护和共享.

③ 便于实现进程的动态增长.

④ 便于实现动态链接, 因为动态链接与进程的逻辑结构有关, 而段按逻辑划分.

(2) 缺点: 段表需连续存放, 易产生碎片.

(3) 分页与分段的异同:

① 同:

(i) 都是非连续分配方式.

(ii) 都需通过地址变换机构实现逻辑地址到物理地址的转换.

(iii) 进行地址变换时, 都需判断页号或段号是否越界.

② 异:

(i) 页是信息的物理单位, 段是信息的逻辑单位.

(ii) 分页是系统管理空间的需要, 对用户不可见; 分段是用户的需要, 对用户可见.

(iii) 分页的目的是提高内存利用率; 分段的目的是满足用户需求.

(iv) 页大小固定, 由系统决定; 段长度不固定, 由用户指定.

(v) 分页管理的地址空间是一维的; 分段管理的地址空间是二维的.

(vi) 分页管理的页内偏移不可能越界; 分段管理的段内偏移可能越界.

6.3.5 虚存管理

[**最佳置换算法**, **OPT**]

(1) 优点: 缺页率最低.

(2) 缺点: 无法实现, 只能作为评价其它算法的参考.

[**先进先出算法**, **FIFO**]

(1) 优点: 实现简单.

(2) 缺点: 未利用局部性原理, 性能较差.

[**最近最久未使用算法, LRU**]

(1) 优点: 性能较好.

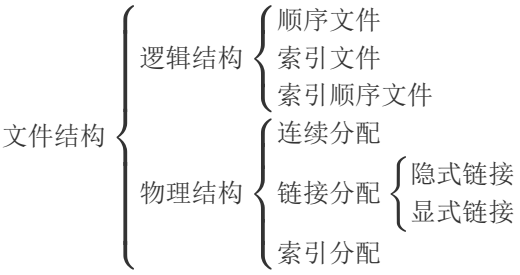
(2) 缺点: 需要硬件支持.

6.4 文件管理

6.4.1 文件与文件系统

[**文件**]

结构:



① 文件的**逻辑结构**: 即文件的组织结构.

② 文件的**物理结构**: 即文件的存储结构, 是文件在外存上的存储和组织形式.

6.4.2 文件的逻辑结构

[**顺序文件**]

(1) 优点:

① 批量操作记录, 即每次读/写大批记录时, 顺序文件效率最高.

② 顺序存储设备 (如磁带) 中, 只有顺序文件能被存储并有效地工作.

(2) 缺点:

① 不方便增删改.

② 文件中的记录变长时, 检索效率低.

改进: 为可变长记录文件建立索引表, 即索引文件.

[**索引文件**]

(1) 优点:

① 方便增删改查.

② 检索速度快.

(2) 缺点: 索引表增加存储开销.

[**索引顺序文件**]

(1) 优点:

① 检索速度快.

② 索引表较短, 节约存储空间.

(2) 缺点: 索引表增加存储开销.

6.4.3 文件的物理结构

[**连续分配**]

(1) 优点:

① 支持顺序访问和随机访问.

② 顺序访问简单, 速度快.

原因: 文件占用的盘块位于一条或几条相邻的磁道上, 磁头移动距离最小,

即寻道数和寻道时间都最小.

(2) 缺点:

① 需有连续的存储空间.

② 易产生外部碎片, 磁盘利用率低.

③ 需事先知道文件的长度, 也无法实现文件的动态增长.

④ 为保持文件的有序性, 插入或删除记录时, 需对相邻的记录作物理上的移动.

[**链接分配**]

优点:

① 无需连续的存储空间.

② 无外部碎片, 磁盘利用率高.

③ 无需事先知道文件的长度, 便于动态地为文件分配盘块.

④ 实现便于文件的增删改.

[**隐式链接**]

缺点:

① 只支持顺序访问, 不支持随机访问.

② 可靠性差, 盘块中任一指针出错都会导致文件数据丢失.

③ 指向下一盘块的指针占用数据区, 导致数据区的大小可能非 2 的幂次, 不利于与内存页对应.

[**显式链接**]

(1) 优点:

① 支持顺序访问和随机访问.

② FAT 在系统启动时即读入内存, 检索在内存中进行, 减少访存次数和访盘次数, 效率高.

(2) 缺点:

① FAT 占用额外的内存空间.

② 不支持高速随机存取.

[**索引分配**]

(1) 一级索引:

① 优点:

(i) 支持顺序访问和随机访问.

(ii) 无外部碎片.

② 缺点:

(i) 占用额外的存储空间.

(ii) 大量的小文件使用大量索引, 浪费空间.

(2) 多级索引:

① 优点: 加快大文件的检索速度.

② 缺点: 访问一个盘块时, 启动磁盘的次数随索引级数的增加而增加.

(3) 混合索引:

优点: 照顾到不同大小的文件.

6.4.4 目录管理

[**目录管理**]

工作:

① 实现 "按名存取" .

② 提高目录检索速度.

③ 实现文件共享.

④ 允许文件重名: 允许不同用户对不同文件使用相同的文件名.

[**单级目录**]

(1) 优点:

① 实现简单.

② 实现 "按名存取" .

(2) 缺点:

① 查找速度慢.

② 不允许文件重名.

③ 不便于实现文件共享.

④ 不适用于多用户 OS .

(3) 目录检索的性能的决定因素:

① 目录项数: 影响平均比较次数.

② 目录项大小: 影响占用盘块数, 进而影响磁盘 I/O 时间.

③ 目录项在目录中的位置: 影响检索路径.

[**两级目录**]

(1) 优点:

① 提高检索速度.

② 允许文件重名, 即不同用户目录下可使用相同的文件名.

③ 便于文件共享.

④ 可在目录上实现访问限制.

(2) 缺点:

① 缺乏灵活性.

② 不能对文件分类.

6.4.5 外存管理

[**文件系统**]

提高文件系统性能的方法:

① 目录项分解: 将目录项分为**符号目录项**和**基本目录项**, 查找文件时只需查符号目录项.

② 文件高速缓存.

③ 采用高效的磁盘调度算法.

[**空闲表法**]

优点:

① 分配速度快.

② 减少访问磁盘的 I/O 频率.

③ 适用于空间对换或小文件系统.

[**空闲链表法**]

分类:

① **空闲盘块链**:

(i) 优点: 分配和回收实现简单.

(ii) 缺点:

i) 为一个文件分配盘块时可能需操作多次, 效率低.

ii) 空闲盘块链很长.

② **空闲盘区链**:

(i) 优点:

i) 分配和回收效率高.

ii) 空闲盘区链短.

(ii) 缺点: 分配和回收实现复杂.

[**位示图法**]

(1) 优点:

① 易找到一组相邻接的空闲盘块.

② 位示图小, 可保存在内存, 进而节省启动磁盘的开销.

(2) 缺点: 位示图的大小随磁盘容量的增大而增大, 常用于小型计算机.

[**成组链接法**]

优点:

① 适用于大型文件系统.

② 空闲盘块栈短, 可保存在内存.

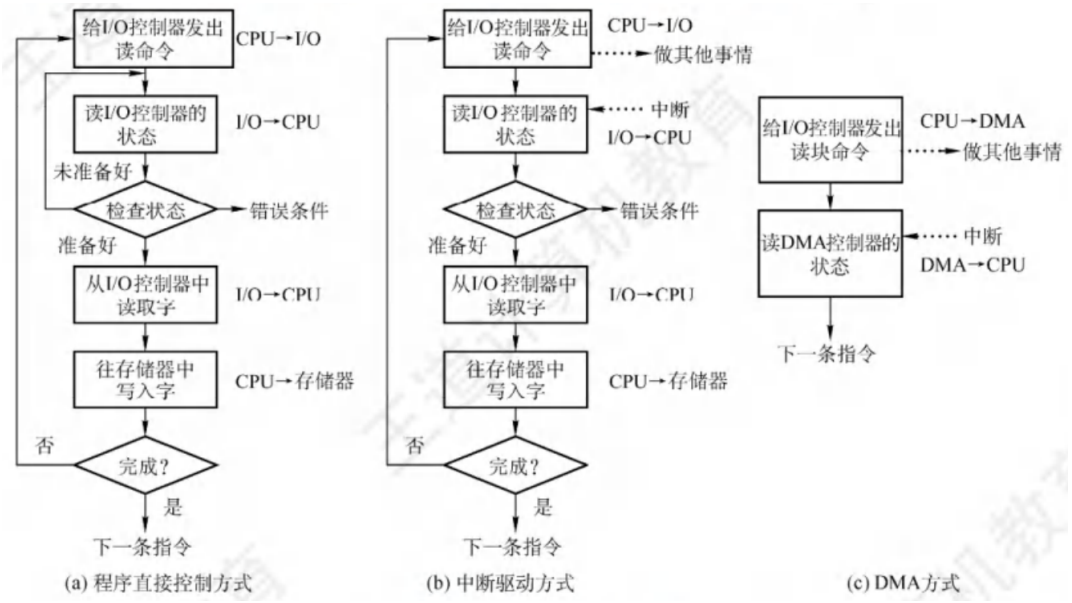
③ 盘块的分配和回收均匀, 减少磁道磨损.

6.5 设备管理

6.5.1 I/O 管理

[I/O 控制方式]

(1) 分类:



① 程序直接控制方式.

② 中断驱动方式.

③ DMA 方式.

(2) **程序直接控制方式 (程序轮询方式)** :

① 优点: 实现简单.

② 缺点:

(i) CPU 绝大部分时间忙等.

(ii) CPU 与 I/O 设备只能串行工作, 资源利用率低.

(3) **中断驱动方式**:

① 优点:

(i) CPU 无需轮询 I/O 设备, 不忙等.

(ii) CPU 与 I/O 设备可并行工作, 资源利用率高.

② 缺点:

(i) 设备与内存的数据交换都需经过 CPU 的寄存器.

(ii) CPU 以字 (或字节) 为单位, 将该方式应用于块设备的 I/O 极其低效.

(4) **直接内存访问方式** (**Direct Memory Access**, **DMA**) :

优点:

(i) 数据传输以块为单位, 效率高.

(ii) CPU 介入频率低.

(iii) 设备与内存的数据交换无需经过 CPU 的寄存器.

(iv) CPU 与 I/O 设备的并行性高.

6.5.2 缓冲管理

[**缓冲管理**]

引入缓冲的目的:

① 缓和 CPU 与 I/O 设备间速度不匹配的矛盾.

② 减少 CPU 中断频率, 放宽 CPU 中断响应的时间限制.

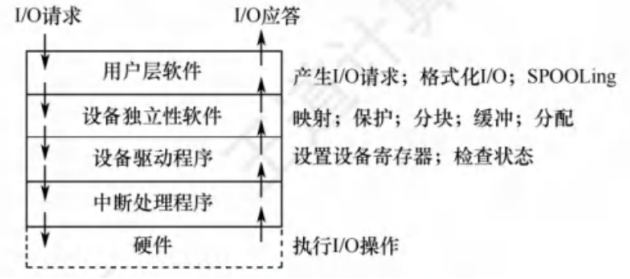
③ 提高 CPU 与 I/O 设备的并行性.

④ 解决基本数据单元大小不匹配的问题.

6.5.3 I/O 软件

[**I/O 软件**]

I/O 层次结构:



[**假脱机技术**, **SPOOLing**]

(1) 主要目的: 提高独占设备利用率.

(2) 策略:

① 模拟脱机输入输出, 将独占设备改造为共享设备, 即将一个物理设备变为多个逻辑设备.

② 系统控制设备与输入井/输出井间的数据传送.

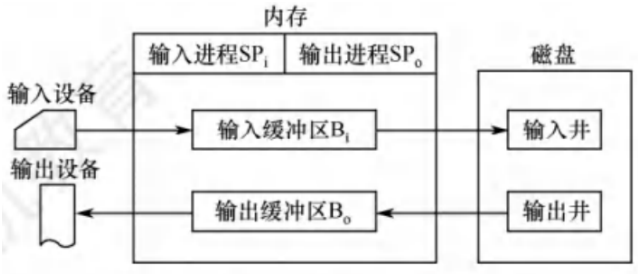
(3) 前提:

① 多道程序系统.

② 大容量的高速随机外存, 通常采用磁盘存储技术.

③ SPOOLing 软件.

(4) 示意图:



(5) 优点:

① 缓和 CPU 与 I/O 设备速度不一致的矛盾.

② 提高 I/O 速度: 将对 I/O 设备的操作转化为对磁盘缓冲区的操作.

③ 将独占设备改造为共享设备, 提高独占设备利用率.

④ 实现了虚拟设备.

6.5.4 磁盘管理

[磁盘的存取时间]

(1) **寻道时间** T_s :

① 定义: 磁头移动到目标磁道所需的时间.

② 计算: 设启动磁臂的时间 s , 移动 n 条磁道, 则 $T_s = m \times n + s$,

其中 m 是与磁盘驱动器有关的常数.

(2) **旋转延迟时间** T_r :

① 定义: 磁头定位到要读/写扇区的时间, 即扇区旋转到磁头下的时间.

② 计算: 设磁盘旋转速度为 r , 则 $T_r = \dfrac{1}{2r}$, 即平均旋转半周.

(3) **传输时间** T_t :

① 定义: 从磁盘读出数据或向磁盘写入数据的时间.

② 计算: 设每次读/写的字节数为 b , 磁盘旋转速度为 r , 一个磁道上的字节数为 N ,

则 $T_t = \dfrac{b}{rN}$, 即 b B 平均占用 $\dfrac{b}{N}$ 个磁道.

(4) **总平均存取时间** $T_a = T_s + T_r + T_t = T_s + \dfrac{1}{2r} + \dfrac{b}{rN}$.

(5) 磁盘的存取时间中:

① 寻道时间占大头, 与磁盘调度算法相关.

② 旋转延迟时间和传输时间由硬件决定.

[先来先服务算法, First Come First Served, FCFS]

(1) 优点:

① 实现简单.

② 有公平性.

③ 访问磁盘的进程少且访问集中时, 性能好.

④ 无 "磁臂粘着" 现象.

(2) 缺点:

① 大量进程访问磁盘时, FCFS 的性能接近随机调度.

② 平均寻道时间较大.

③ 只适用于磁盘 I/O 较少的场合.

[**最短寻道时间优先算法**, **Shortest Seek Time First**, **SSTF**]

(1) 优点: 平均寻道时间较短.

(2) 缺点:

① 每次的寻道时间最短不能保证平均寻道时间最短.

② 可能饥饿, 即某些进程长时间得不到访问.

③ 有 "磁臂粘着" 现象, 即磁头长时间停留在同一磁道.

[**扫描算法**, **SCAN**, **电梯调度算法**]

(1) 优点:

① 不饥饿.

② 平均寻道时间较短.

(2) 缺点:

① 有 "磁臂粘着" 现象.

② 偏向于处理最内侧和最外侧的请求.

③ 对最近扫过的磁道不公平, 即对与当前磁头距离近但在磁头移动的反方向的磁道不公平.

④ 访问局部性方面不如 FCFS 算法和 SSTF 算法.

[**循环扫描算法**, **C-SCAN**]

(1) 优点:

① 不饥饿.

② 平均寻道时间较短.

③ 最长等待时间比 SCAN 算法少一半.

④ 对各磁道公平.

(2) 缺点: 有 "磁臂粘着" 现象.