Chapter One 操作系统概述

操作系统

概念：操作系统是指控制和管理整个计算机系统的硬件和软件资源，并合理组织和调度计算机的工作和资源分配，是最基本的系统软件。

特征：并发、共享（两个最基本的特征）、虚拟、异步。
并发：指同一时间间隔内发生，区别于并行。微观上分时地交替执行。
功能：是计算机系统资源（处理机、存储器、文件、设备）的管理者

用户与计算机硬件系统之间的接口：
①命令接口（允许用户直接使用）（1）联机（交互式）命令接口（适用于分时or实时）
（2）脱机（批处理）命令接口
②程序接口（=系统调用命令）
③GUI（图形接口调用系统命令）
注：在多道程序环境下，处理机的分配和运行都以进程（或线程）为单位。
系统调用是由操作系统提供给用户的，它只能通过用户程序间接使用。

操作系统的发展：批处理—>分时—>实时—>网络和分布式

①批处理（缺点：没有交互能力）
单道批处理—>顺序性（CPU大量时间在空闲等待I/O）
多道批处理（失去封闭性）—> 制约性、间断性、共享性
特点：多道、宏观上并行，微观上串行。
②分时系统：（以时间片为单位）允许多个用户以交互的方式使用计算机
特点：同时性、交互性、独立性、及时性
分时系统能较快、及时接收并处理命令，快速响应用户。
（通常采用优先级+非抢占式调度算法）
分时系统中，时间片一定时，用户数越多，响应时间越长。
③实时系统：在某个时间限制内完成某些紧急任务而不需时间片排队
特点：及时性、可靠性
（通常采用抢占式优先级高者优先算法）
④网络（网络资源共享）和分布式：区别是在分布式中，若干计算机相互协同完成同一任务

系统调用（运行在核心态）（涉及设备、文件、进程、内存）

用户程序凡是与资源有关的操作（存储分配、I/O、管理文件）都必须通过系统调用。
过程：传递系统调用参数—>执行陷入（trap）指令（用户态）—>执行系统调用相应服务程序（核心态）—>返回用户程序
系统调用功能是操作系统向用户程序提供的接口
注：系统调用是一种特殊公共子程序

陷入指令是唯一一个只能在用户态执行，而不可在核心态执行的指令。
广义指令：也就是系统调用命令（可能在用户态调用，但处理必须在核心态）

用户程序（用户自编or系统外层应用程序）工作在用户态；内核程序工作在核心态。
特权指令：只能在核心态运行的指令
如：I/O指令、置中断指令、存取用户内存保护的寄存器、送程序状态字（可区分目态、管态）到程序状态字寄存器。（包括系统调用类、时钟类、中断和原语指令，清内存、分配系统资源、修改虚拟存储里的页表段表、修改用户访问权限等）

中断和异常：引入中断技术的初衷是提高多道程序运行环境中CPU的利用率

中断的分类：①内中断（异常、例外、陷入trap）（不可被屏蔽！）
自愿中断—指令中断：访管指令（只能用户态使用）
强迫中断—硬件故障（缺页）
—软件中断（非法操作码、地址越界、算数溢出、虚存系统缺页以及专门的陷入）
②外中断（强迫中断）
外设请求：I/O结束、时钟中断
人的干预：用户按ESC or 退出键
注：区分内/外中断看信号来源：CPU内部/外部。
访管中断：用户程序在用户态下要使用特权指令（由访管中断引起）引起的中断。
用户程序需要输入/输出时（I/O），调用OS提供的接口，此时引起访管中断。
所有中断都是在核心态下执行的！（进程切换、对资源的释放）

用户态（发生中断 or 异常）—>核心态 （通过硬件、系统调用、访管指令实现）
核心态（使用特权指令）—>用户态 （通过中断返回指令）
注：中断系统（OS必需）和地址映射需要硬件支持，进程调度不需要。

原语

处于最底层；不可分割的指令序列；运行时间短，调用频繁
PV操作是一种低级的进程通信语言，由两个不可中断的过程组成，并非系统调用。

体系结构：

大内核（高性能；结构混乱）、微内核（内核功能少；在用户态、核心态之间切换频繁，性能低；结构清晰；添加系统服务时不必修改内核；使系统更可靠）

Chapter Two 进程管理

进程概念：

进程（动态）是资源分配的一个独立单位。程序：静态

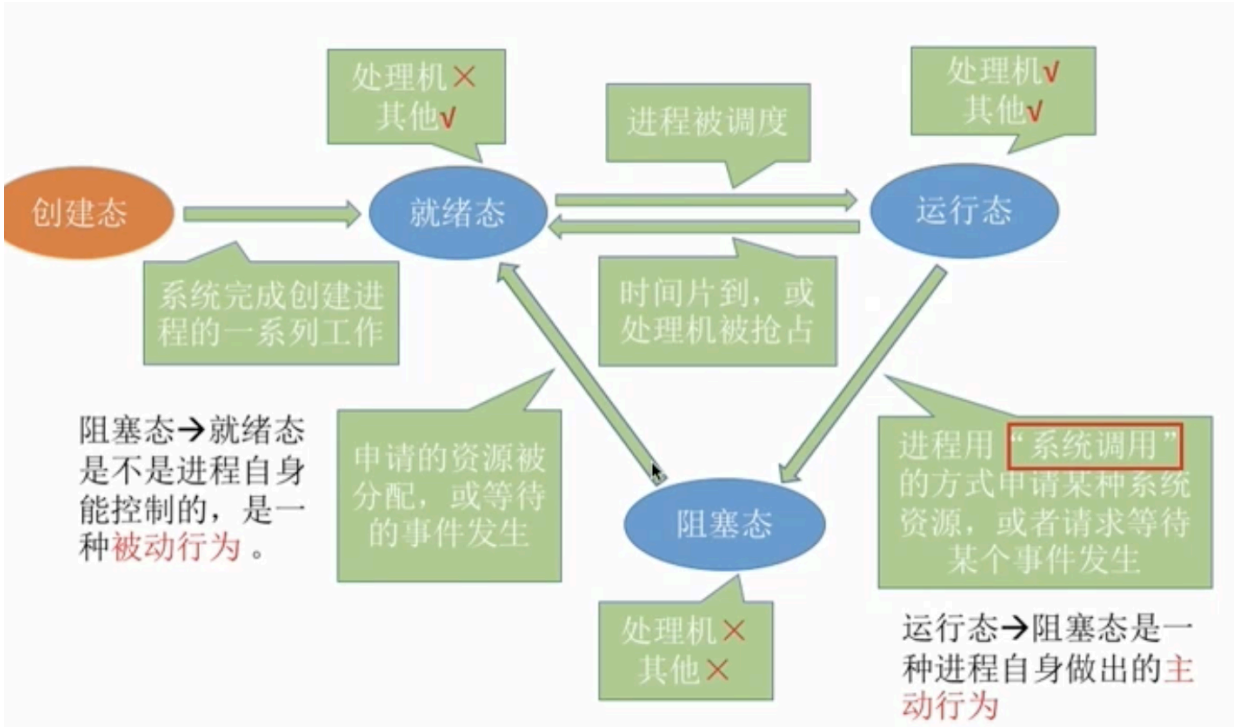
进程的特征：动态性（最基本）、并发性（重要特征）、独立性、异步性、结构性（进程实体（进程映像）由程序段、数据段、PCB三部分组成）

注：进程的组织（结构性）：PCB、程序段（多个进程可运行同一程序）、数据段

PCB是进程存在的唯一标志。主要包括了：进程描述信息（ID）、进程控制（优先级）和管理信息、资源分配和处理机相关（不重要）。

二进制代码和常量放在正文段；动态分配的存储区在数据堆段；临时用的变量在数据栈段。

进程的状态：运行、就绪、阻塞、创建、结束



运行—>阻塞（等待）主动阻塞

阻塞—>就绪 被动唤醒

注：在可剥夺OS中，当有更高优先级的进程就绪时，调度程序将正在执行的进程—>就绪态，让更高优先级的执行。

就绪态：进程已处于准备运行的状态（只缺CPU了！）

进程切换：（区别于调度！切换是执行行为，而调度是决策行为）：时间片用完、主动放弃处理机、被更高优先级的进程剥夺

注：进程切换的过程包括更新PCB信息

引起创建进程的操作：终端用户登录系统、作业调度、系统提供服务、用户程序的应用请求

注：用户进程被创建后，随着运行的正常或不正常结束而撤销。（进程是有一定生命周期的！）

进程的终止：①异常结束：存储区越界、保护错、非法指令、特权指令错、I/O故障 ②正常结束：任务已完成 ③外界干预（人为、OS干预、父进程的请求or终止）

阻塞（等待资源）：请求资源失败、等待某操作的完成、数据未到达、无事可做等

唤醒（资源到达）：I/O操作已完成 or 数据已到，调用唤醒原语

进程的通信

一个进程不能直接访问另一个进程的地址空间

①共享存储（互斥访问）：低级方式：基于数据结构的共享；高级方式：基于存储区

②消息传递：直接通信方式：接收进程从消息队列中取得消息；

间接通信方式：将消息挂到某个中间实体（邮箱）

③管道通信：利用一种特殊的pipe文件连接两个进程。

管道只能采用半双工通信，某一时间段内只能实现单向传输。如果要实现双向同时通信，则需设置两个管道。（原理：Chapter 5缓冲区）

注：从管道读数据是一次性操作，数据一旦被读取，它就从管道中被抛弃

线程

线程的引入：减小程序的时空开销，提高程序并发执行的程度，提高系统效率

线程是程序执行的最小单元，并不拥有任何系统资源（进程才有），是独立调度的基本单位。

同一进程中，线程的切换不会引起进程的切换；切换到另一进程中的线程才会切换。

同一进程或者不同进程内的线程都可以并发执行。

用户级线程：所有工作都由应用程序完成，无需内核干涉。

多线程模型：多对一模型：缺点—>一个线程阻塞会导致整个进程都被阻塞

注：线程包含CPU现场，可以独立执行程序。

只有内核级线程才是处理机分配的单位！

CPU调度

①作业调度（高级DD）：内存与辅存（外存）之间的DD；对于每个进程只调入/调出一次。调入建立PCB，调出才撤销PCB。

②内存DD（中级DD）：将暂时不运行的进程调至外存等待。引入中级DD为了提高内存利用率和吞吐量（调到外存等待的进程状态为挂起态）

③进程DD（低级DD）：内存—>CPU，是OS中最基本的一种DD；一般OS中必须配置，使用频率很高。

带权周转时间=作业周转T/作业实际运行T

不能进行进程调度/切换的情况：

①处理中断过程中

②进程在OS内核程序临界区—>需要独占式访问共享资源（不能进行进程DD但还是能进行CPU调度！前提：不能破坏临界资源使用规则）

③需要完全屏蔽中断的原子操作（不可分割！连中断都要屏蔽，DD更别说了）

（如：加锁、解锁、中断现场保护/恢复）

应该进行进程调度/切换的情况：

①发生引起DD的条件且当前进程无法继续执行下去（非剥夺方式）

②中断or trap处理结束后，返回被中断进程的用户态程序执行现场前，可以马上进行DD与切换。（剥夺方式）

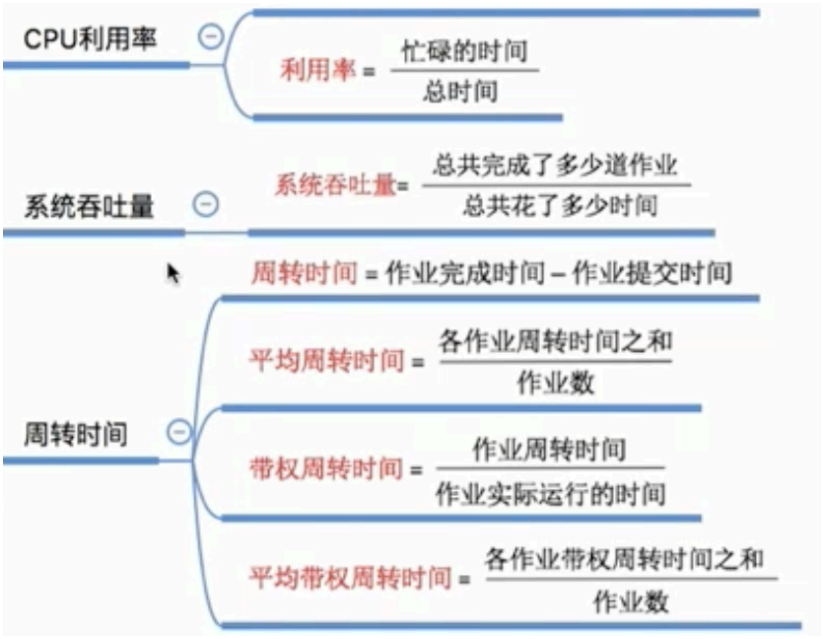
调度方式：剥夺式（抢占）、非剥夺式（非抢占）

剥夺式：当某个更紧急的进程要CPU时，立即暂停正在执行的进程，先分给更紧急的。（必须遵循一定规则，如：优先权、SJF or 时间片）

优点：提高系统吞吐率和响应效率

非剥夺式：一旦CPU分配给一个进程，该进程保持CPU直到终止 or 转换到等待态。

特点：实现简单、系统开销小；适用于批处理，不能用于分时 or 实时！



调度算法：

FCFS、SJF、优先级DD、高响应比优先、时间片轮转、多级反馈队列DD。

FCFS：属于不可剥夺算法！

特点：算法简单；有利于CPU繁忙型作业，不利于I/O繁忙型作业。

SJF：会产生饥饿现象，是调度策略问题。（默认"非抢占"，也有抢占式）

特点：平均等待时间、平均周转时间最少！

优先级DD：①静态优先级：优先级在创建进程时确定，整个运行期间不变

②动态优先级：随着进程执行时间增加，其优先权下降。

高响应比优先：Rp=(waitT+ServeT)/ServeT

时间片轮转（队列的思想）：主要适用于分时系统；绝对可抢占；时间片过大时，相当于FCFS

注：I/O型作业优先权高于计算型作业！I/O作业要及时完成，无法长期保存输入/输出的数据。

处理机DD算法不影响作业执行或输入/输出操作的时间，只影响作业在就绪队列中等待所花的时间。（即DD算法优劣只需考虑等待时间）

进程同步

临界资源（独占资源）：一次仅允许一个进程访问使用的资源

（如：打印机、共享变量、共享缓冲区、公用队列）

共享资源：磁盘存储介质、可重入代码（一次可供多个进程使用，不允许任何修改的代码—>共享程序）

临界区：进程中访问临界资源的那段代码

注：进程处于临界区时，不能进行进程DD，但是能进行处理机/CPU调度！但要不能破坏临界资源使用规则

同步机制遵循的原则：①空闲让进②忙则等待③有限等待④让权等待

④：当进程不能进入临界区时，应释放处理器

实现临界资源互斥的基本方法：（以下都不满足”让权等待”！）

软件：①单标志法（只能按顺序进入）②双标志法（同时进入临界区）③双标后检测（可能造成饥饿）④Peterson’s 算法（双重，主动谦让，将”钥匙”送给对方，最终只有一个可通过）P73方四行代码

硬件：TestAndSet（原子操作） or Swap（简单了解）

特点：实现简单；适用于多处理机；不满足”让权等待”

信号量

①整型信号量：表示资源数量 （不满足”让权等待”）

②记录型信号量：s.value<0时（=0也不算是等待！），|s.value|代表链表中已被阻塞的该信号进程的数目（即等待进入临界区的）遵循了”让权等待”原则

信号量机制实现同步与互斥

①互斥：设置互斥信号量mutex，初值为1。semaphore mutex=1;

②同步：必须保证”一前一后”（前V后P）执行的操作。设置同步信号量S，初值为0。

注：同步：要为每一对前驱关系各设置一个信号量。

P、V操作必须成对出现

题型：①生产者消费者：

```
semaphore mutex = 1; //互斥信号量
semaphore empty = n; //同步信号量，空闲缓冲区的数量
semaphore full = 0; //同步信号量，产品的数量（非空缓冲区的数量）
```

多生产者多消费者：P081吃苹果吃橘子
②吸烟者问题：P085
semaphore mutex = 1; //（桌子）互斥信号量—>但是此题不需要！
semaphore offer1 = 0; //同步互斥量，桌子上组合1的数量
semaphore finish = 0; //表示抽烟完成的信号
int i = 0; //对i取余然后V(offer)，用于实现”轮流”抽烟（根据题目要求）
③读者写者：P082（读写公平法略繁）
semaphore mutex = 1; //对count==0时进行保护，实现第一个读者和第二个一起
semaphore rw = 1; //保证读者写者互斥访问文件
int count = 0; //读进程的数量
④哲学家进餐

死锁

定义：多个进程因竞争资源而造成的互相等待
原因：①竞争系统资源（不可剥夺资源）；独占资源分配不当
②进程推进顺序非法；请求和释放资源的顺序不当 or 信号量使用不当

死锁的必要条件：互斥；不剥夺；请求和保持；循环等待（任一条件不满足时则不发生死锁）
不剥夺：未使用完之前不能被其他进程强行夺走，只能是主动释放！
请求和保持：保持资源时又请求被其他进程占有的资源，对已获得的资源保持不放。
死锁的处理策略：
①死锁预防（通过设立限制条件，破坏四个必要条件，但互斥无法破坏，超级少见！）
SPOOLing技术：把互斥资源改造为允许共享使用
包括：采用静态分配法，一次申请完全部资源；采用顺序资源分配法，按编号…
②死锁避免：在资源动态分配过程中，用一些算法防止系统进入不安全状态。
包括：银行家算法（Max、Allocation、Need、Available）

③死锁检测：资源分配图（当不可完全化简时为死锁状态）；死锁定理
④死锁解除：①资源剥夺法（被动）②撤销进程 ③进程回退法（主动释放）
注：出现环路不一定都会导致死锁。
死锁and饥饿（不安全状态）：都是由资源分配策略不当引起
死锁发生进程数>=2（互相等待）
饥饿不一定是死锁，但是至少一个进程被无限期推迟。

Chapter Three 内存管理

内存是用于存放数据的硬件。程序执行前需要先放到内存中才能被CPU处理。

内存管理：更好地支持多道程序并发执行，提升系统性能，通过虚拟技术从逻辑上扩充内存

注：1字节=2字

程序执行过程：编译、链接、装入

链接（链接时形成逻辑地址）：

①静态链接：将各目标模块及库函数连城一个完整可执行程序，以后不再拆开

②装入时动态链接：边装入，边链接（早期多道批处理）

③运行时动态链接：需要用到时才链接（现代操作系统）

装入：（将逻辑地址转换为物理地址）

①绝对装入：编译时产生绝对地址。程序中逻辑地址与实际内存地址完全相同；只适用于单道程序环境

②可重定位装入（静态重定位）：地址变换通常在装入时一次完成

特点：一个作业进入内存，必须分配其要求的全部内存空间。

在运行期间就不能再移动位置。没有足够内存时，不能装入作业！

装入时把逻辑地址变换为物理地址，装入后不能改变。

③动态运行时装入：不立即转换地址，真正要执行时才进行（需要重定位寄存器支持）

特点：动态重定位在作业执行过程中进行。

程序运行前只装入部分代码即可运行。

运行期间动态申请分配内存，便于程序段的共享。

地址重定位（地址映射）

逻辑地址（又称相对/虚地址）—>物理地址（内存、绝对、实地址）

注：物理地址可以直接寻址；不能直接用逻辑地址在内存中读取信息！

不同进程可以有相同的逻辑地址，但是会映射倒不同的主存位置上。

关系图（略）P142

重定位寄存器（用来"+"的，用来算物理地址）整个系统仅设置一个！

界地址寄存器（用来"比"，判断有无越界的）

覆盖与交换——解决空间不足的问题，但并非物理上扩充，是从逻辑上扩

覆盖：特点是打破了必须将一个进程的全部信息装入主存后才能运行的限制。（对用户不透明，已成历史）

内存中能够更新的地方只有覆盖区的段，不在覆盖区的段会常驻内存。

交换：把处于等待状态（或在CPU调度原则下被剥夺运行权利）的程序从内存移到外存（对应进程的中级调度），把内存空间腾出来。

注：交换技术主要在不同进程（或作业）之间进行，而覆盖技术则用于同一进程（或作业）。

连续分配——一个用户程序分配一个连续的内存空间

①单一连续分配：无需内存保护（内存中只有一道程序，肯定不会访问越界干扰其他程序）；实现简单；无外部碎片；可采用覆盖技术

注：换而言之，多进程要保证彼此互不干扰，OS需要通过内存保护来实现。

缺点：只适用于单用户、单任务的OS中；有内部碎片；存储器利用率低

②固定分区分配：最简单的一种多道程序存储管理方式，有内部碎片。

③动态分区分配：在装入时，根据进程的大小动态建立。产生外部碎片。

注：

内部碎片：分区大小固定，程序小于固定分区时也占用一个分区，内部浪费

外部碎片：在所有分区外的存储空间会变成越来越多的碎片（可以用拼接、紧凑技术解决，需要动态重定位寄存器支持）

分配算法：

①首次适应：空闲分区以地址递增的次序链接，找大小满足要求的第一个分区

②最佳适应：XX按容量递增XX

③最坏适应：XX按容量递减XX

④邻近适应：循环首次适应算法，分配内存时，从上次查找结束的位置开始继续查找。

注：首次适应（First fit）是最简单，最好，最快的。

最佳适应（Best fit）性能通常很差，产生最多外部碎片。

分区分配内存管理方式的主要保护措施是：界地址保护

编址空间大小取决于硬件的访存能力

非连续分配（离散）——一个程序分散地装入不相邻的内存分区

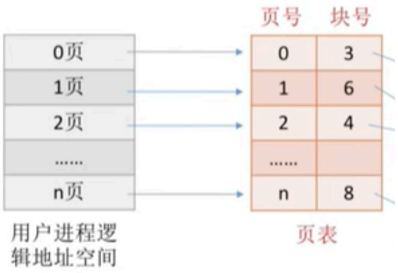
分页存储：有内部碎片；基地址变换由硬件自动完成

基地址变换机构 ①有快表（高速缓冲寄存器）TLB ②无快表

页表：由页表项组成。

页表项：页号P+物理内存中的块号

逻辑地址结构：页号P+页内偏移



页表长度：指的是这个页表中总共有几个页表项，即总共几个页；

页表项长度：每个页表项占多大的存储空间；页面大小：一个页面占多大存储空间。

物理地址=物理内存中的块号+页内偏移（理解：P148便利贴）

注：OS对内存采用页式存储管理时，所划分的页面大小必须相同。

若页表全部放在内存中，则存取一个数据或一条指令至少要两次访问内存。（若在快表中找到，则仅需一次）—>涉及题型：快表命中率

快表的有效性基于局部性原理。

多级页表：如N级页表需访问N+1次

页表管理中，地址空间是一维的；段式XX是二维的。
分页管理方式—>主要方便操作系统。通过硬件机制实现，提高内存的利用率，提升性能，对用户透明
分段存储：有外部碎片
分段管理方式—>考虑用户和程序员。方便编程、信息保护和共享、利于动态增长和动态链接
地址空间要求段内连续，段间不做要求。
注：内存保护需要由OS与硬件机构合作完成
段页式管理方式：有内部碎片。P154地址变换机构
用分段的方式来分配和管理用户地址空间；用分页的方式来管理物理存储空间。
在进行地址变换时，进行一次访问实际需要三次访问主存。有快表则两次。
在一个进程中，段表只有一个，而页表可能有多个。

虚拟内存管理—>从逻辑上扩充内存（只能基于非连续分配技术）

属于高速缓存技术，基于局部性原理。（即不必装入全部程序）
装入时，只需装一部分，其余留在外存就可以启动；执行时，当访问的信息不在内存，则用OS将其调入内存，将暂时不用的换出至外存（内<—>外）

局部性原理：时间局部性、空间局部性（理解）
注：虚拟存储器的大小由计算机的地址结构决定（CPU地址线数），并非简单的内外存相加。容量与主存的实际大小没直接关系，由主存和辅存（硬盘）的容量之和所确定。
在虚存的页表项中，合法位决定是否会发生缺页故障。
虚拟存储器三大特征：多次性、对换性、虚拟性

请求分页管理方式：虚拟存储器的实现，目前最常用
组成：①页表机制 ②地址变换机构
③缺页中断机构：页面不在内存时，产生一个缺页中断（属于内中断），请求OS将所缺页调入（一条指令在执行期间可产生多次缺页中断）
注：页式虚拟存储管理的主要特点：不要求将作业同时全部装入到主存的连续区域

页面置换算法：

①最佳置换算法OPT（向后看）
②先进先出FIFO：基于队列实现（会有Belady异常）
③最近最久未使用LRU（向前看）
特点：是堆栈类算法。性能较好，但需寄存器和栈的硬件支持。（需要对所有页进行排序）
④最近未用NRU（CLOCK）
CLOCK置换：（改进型淘汰次序，性能接近LRU）
先未被访问（u=0），未被修改（m=0）；
再未被访问（u=0），已被修改（m=1）；

然后（u=1，m=0）；最后（u=1，m=1）。
注：LRU和OPT都不可能发生Belady异常

影响缺页次数的因素：分配给进程的物理页面数、页面本身的大小、程序编程的方法、页面淘汰的算法

页面分配策略：固定分配局部置换（物理块数一定）、可变分配局部置换、可变分配全局置换（是最易于实现的物理块分配和置换策略）
驻留集：指请求分页存储管理中给进程分配的物理块的集合。（太小会导致缺页频繁）
二种调页策略：预调页：用于进程首次调入时（运行前调入）
请求调页：易于实现，虚存中大多采用此策略。缺点为每次只调入一页（运行时/期间调入）

抖动：刚换出的页面马上又要换入内存，刚换入的页面马上又要换出内存
频繁换入、换出使系统效率低下
原因：淘汰算法不合理、物理页面太少
解决方案（提升系统性能、改进CPU利用率）：增加内存容量；撤销部分进程；换淘汰算法；减少多道程序的度数（所有增大交换区的选项都不要选！）
注：FIFO最易发生抖动，但所有页面调度策略都不可能完全避免抖动。

Chapter Four 文件管理

在用户进行输入、输入中，以文件为基本单位。
在系统运行时，计算机以进程为基本单位进行资源的调度和分配。
注：从系统角度看，文件系统负责对文件的存储空间进行组织、分配，负责文件的存储并对存入文件进行保护、检索。
从用户角度看，操作系统引入文件系统的目的是：实现对文件的按名存取。

基本操作：创建文件、写文件、读文件、文件寻址、删除、截断

创建文件步骤：①在文件系统中为文件找到空间 ②在目录中为新文件创建条目
文件的打开：①首次打开：系统调用open将指明文件的属性从外存拷贝到内存打开文件目录表的一个表目中 ②非首次：open根据文件名搜索目录（FCB调入内存），并将目录条目复制到打开文件表（包含所有打开文件信息的表）。
用户需要一个文件操作时，通过打开文件表的索引指定文件，省略了搜索。
文件的关闭：OS从打开文件表中删除某一条目，回收分配给文件的内存空间资源，释放文件的文件控制块（FCB）。
文件控制块（FCB）：存放控制文件所需的各种信息的数据结构，实现"按名存取"。一个FCB就是一个文件目录项，即本身就也被看成一个文件（相似进程中的PCB）

文件的逻辑结构：是从用户观点出发看到的文件组织形式

文件逻辑上都可以看作连续的，但在物理设备上并不完全连续。文件的物理结构从实现出发。
无结构文件（流式文件）：最简单的文件组织形式
有结构文件（记录式文件）：①顺序文件：批量操作效率高；对单个（增删改查）较困难 ②索引文件：成百上千倍提高访问速度 ③索引顺序文件 ④直接文件和散列文件：没有顺序的特性。
注：索引表本身是定长记录的顺序文件
对于含有N个记录的顺序文件，查找某关键字值的记录平均需要查找N/2次；在索引顺序文件中，需要$\sqrt{N}$次。
索引文件&索引顺序文件都提高了存取速度，但因为配置索引表而增加了存储空间。
散列文件有很高的存取速度，但是会引起冲突。
存放在磁盘上的索引结点称为磁盘索引结点，UNIX中的每个文件都有一个唯一的磁盘索引结点。

目录结构—>目录管理要实现"按名存取"；目录存取的效率直接影响系统性能

①单级目录结构：在整个文件系统中只建立一张目录表，每个文件占一个目录项
特点：不便于文件共享；按名存取；文件不允许重名
②两级目录结构：将文件目录分成：主文件目录和用户文件目录
特点：解决文件重名问题；可以在目录上实现访问限制；缺乏灵活性，不能分类

③多级目录结构（树型）：方便分类，结构清晰；但增加了磁盘访问次数，影响查询速度；不便于实现文件的共享

绝对路径：从根目录出发的路径。

相对路径：进程对各文件的访问都是相对于当前目录进行的。当用户要访问某个文件时，使用相对路径标识文件。

④无环图目录结构：方便实现了文件共享。

注：仅当共享计数器为0时才真正删除结点，否则仅删除请求用户的共享链。（不可删除非空目录）

文件共享：使多个进程共享同一份文件，系统中只需保留该文件的一份副本

①基于索引结点的共享方式（硬链接）

索引结点中存放文件的物理地址和属性

文件目录中只存放文件名和指向索引结点的指针

链接计数count：用于表示链接到本索引文件上的用户目录项的个数。

当用户A创建一个新的文件时，count=1；B共享该文件时，count=2。

若A不需要此文件，不能将其直接删除，只能将count-1（此时B还能使用此文件）；只有当count=0，表示没有用户使用时，系统将其负责删除。（理解）

②利用符号链实现文件共享（软连接，类似快捷方式）：删除文件时，Link还在

注：文件共享是”软”+”硬”链接。硬链接就是多个指针指向一个索引结点，保证只要还有一个指针指向索引结点，索引结点就不会被删除；软连接就是把到达共享文件的路径记录起来，访问时，根据路径寻找文件。（硬链接的查找速度比较快）

文件保护—>口令保护、加密保护、访问控制

访问控制：为每个文件和目录增加一个访问控制列表（ACL）

特点：安全性较高；灵活性低；非系统实现

加密保护：防止文件被窃取；编码、解码需要时间；并没有控制用户对文件的访问类型

特点：安全性较差；灵活性高；由系统实现

注：对一个文件的访问通常由用户访问权限和文件属性共同控制。

对多级目录结构的文件保护：不仅需保护单个文件，还需保护子目录内的文件，即需目录保护机制。

系统级安全管理包括：注册和登录。

文件系统实现

文件系统的层次结构：（当用户请求访问某个文件时）

用户调用接口—>文件目录系统—>存取控制验证层（FCB里看是否有访问权限）—>（真正寻址，得到逻辑地址）逻辑文件系统和文件信息缓冲区—>（逻址变换物址）物理文件系统

文件实现——研究文件的物理结构

文件的分配方式：

①连续分配：要求每个文件在磁盘上占有一组连续的块

特点：支持顺序访问和直接访问；实现简单，存取速度快；文件长度不宜动态增加（类似于顺序表）

②链接分配：采用离散分配的方式

显式：把用于链接文件各物理块的指针从块末尾提出，显式地放入内存的一张链表（文件分配表FAT—>该表在整个磁盘仅设一张）中。没有碎片问题；外存利用率高；支持随机访问

隐式：无法直接访问盘块，只能通过指针顺序访问文件。方便拓展；没有碎片问题；外存利用率高；只支持顺序访问，不支持随机，查找效率低。

③索引分配：索引块应尽可能小（索引表—>一个文件分配一张，要占一定空间）

注：m级索引访问记录需访问磁盘m+1次

对文件的存取：随机存取时，索引文件速度快；顺序存取时，顺序文件速度快。

在磁盘上，最容易导致存储碎片发生的物理文件结构是：顺序存放。

文件的存储空间管理：实际上是对空闲块的组织和管理

①空闲表法 ②空闲链表法 ③位示图法（要推算出盘块号与字号/位号相互转换的公式：P230） ④成组链接法

磁盘

扇区：是磁盘可寻址的最小存储单位

一次磁盘读写操作时间由寻道时间、延迟时间和传输时间决定。

但实际上存取时间与磁盘调度算法密切相关。调度算法直接决定了寻道时间。

DD算法：①FSFS ②SSTF：会产生”饥饿”

③SCAN（电梯算法）：（只有到了最边上的磁道才能改变磁头方向！）要知道磁头当前位置和磁头移动方向

④C-SCAN：只需要到达最远端的一个请求即可返回

⑤LOOK和C-LOOK：只要到请求磁道的最边上。

注：磁盘是共享设备，但在每一时刻，至多只能由一个作业启动它。

光盘、磁盘、U盘既可以顺序访问有可以随机访问。磁带只能顺序访问。

Chapter Five I/O管理

I/O设备分类：

（按）使用特性：人机交互、存储、网络通信
（按）传输速率：低速（键盘鼠标）、中（打印机）、高（磁盘光盘）
（按）信息交换单位：块设备、字符设备

块设备：以数据块为单位存取（磁盘）
基本特征：传输速率较高；可寻址；可对他随机读/写任一块
字符设备：用于数据输入/输出。以字符为单位传输；无结构类型（交互式终端机、打印机）
基本特征：传输速率低；不可寻址；在I/O时常采用中断驱动方式

I/O控制方式：

①程序直接控制方式 ②中断驱动方式
③DMA方式：DMA方式是在I/O设备和主存之间建立一条直接数据通路。
注：DMA交换方式：I/O设备与存储设备进行数据交换不经过CPU来完成。（CPU只参与预处理和结束过程）
④通道控制方式：
工作过程：向通道发一条I/O指令，给出通道程序的首地址和要访问的I/O设备，通道接到指令后，执行通道程序便可完成CPU指定的任务，数据传送结束向CPU发中断请求。
I/O通道：专门负责I/O的处理机，实现CPU、通道、I/O三者并行工作
注：通道用于实现内存与外设之间的信息传输。通道是一种特殊的处理器，属于硬件技术。
SPOOLing、缓冲池、内存覆盖都是在内存基础上通过软件实现。
字节路通道：用于连接大量低速、中速设备
I/O通道与一般处理机的区别：通道指令类型单一；没有自己的内存，通道程序放在主机内存中，与CPU共享内存
I/O通道与DMA方式的区别：DMA：CPU控制传输的数据块大小、传输的内存位置；每个DMA控制器对应一台设备与内存传递数据。通道：上述信息由通道控制，可控制多台设备与内存的数据交换。
注：通道技术指的是一种硬件机制
磁盘设备的I/O控制器主要是采取：DMA方式

I/O层次结构（从上到下）：P264具体例子理解

①用户层 ②与设备无关的软件层（系统调用的处理程序）
③设备驱动程序（与硬件直接相关）：负责执行OS发出的I/O命令（如：计算磁头号、柱面号、扇区号），因设备的不同而不同，几类设备就配几个设备驱动程序程序！
注：（对比）系统调用：OS提供给用户程序的通用接口，不会因为具体设备的不同而不同

④中断处理程序 ⑤硬件 •ꈍ(ᵒωᵒ*•ꈍ))ↄꈍ)
注：设备独立性：用户编程时使用的设备与实际使用的设备无关
在应用程序中，使用逻辑设备名来请求使用某类设备；而在系统实际执行时，必须将逻辑设备名映射成物理设备名使用

I/O核心子系统： (I/O层次结构的②③④)

提供的服务主要有：I/O调度、缓冲与高速缓存、设备分配与回收、假脱机SPOOLing 、设备保护和差错处理等

引入缓冲区的主要目的（背）：①缓和CPU与I/O间速度不匹配的矛盾
（如果I/O所花时间比CPU处理时间短得多，则没必要设置缓冲区）
②减少对CPU的中断频率 ③解决基本数据单元大小不匹配的问题 ④提高CPU和I/O设备之间的并行性
缓冲区特点：在缓冲区非空时，不能冲入数据，只能传出；当缓冲区为空时，可以冲入数据，但是要缓冲区充满后，才能传出。（管道通信中的"管道"也是缓冲区原理）
单缓冲：设备和处理机之间设置一个缓冲区。处理每块数据用时：Max（C,T）+M
双缓冲：处理每块数据用时：Max（C+T,M）
M+C<T（传输慢），可以设备连续输入；
M+C>T（处理慢），CPU不必等待设备输入。
缓冲池：由多个系统公用的缓冲区组成。
四种缓冲区：收容输入、提取输入、收容输出、提取输出（过程P274理解）

设备控制器要提供：控制寄存器、状态寄存器和控制命令
设备控制器中用于实现对设备控制功能的是：I/O控制逻辑
设备绝对号：用于区分硬件和识别设备的代号。系统中每一台设备按某种原则统一进行编号

设备分配时应考虑的因素：

①设备的固有属性（独占设备、共享设备、虚拟设备）
共享设备：必须可寻址，可随机访问。分配共享设备不会引起死锁（独占设备才可能）。在一段时间内允许多个进程访问，但在同一时刻，只允许一个进程访问。
虚拟设备：把一个物理设备变换成多个对应的逻辑设备
②设备分配算法
③设备分配中的安全性：应防止发生进程死锁。一般不需要考虑及时性
安全分配方式：设备分配安全，但CPU和设备并行工作
不安全分配方式：进程可同时操作多个设备，推进迅速；但有可能产生死锁
逻辑设备表的设置：①整个系统只设置一张LUT：不允许有相同逻辑设备名；主要适用于单用户系统 ②为每个用户设置一张LUT（该表放入进程PCB中）
注："设备、控制器、通道"的关系：树型（一个设备分配成功，必须这三个都可用）

一个通道可控制多个设备控制器，每个控制器可以控制多个设备。

SPOOLing技术（假脱机技术）—>用软件控制

实质上是用户层软件，但也归为I/O核心子程序

脱机技术：为了缓和CPU的高速性与I/O设备低速之间的矛盾

SPOOLing系统的主要特点：提高I/O速度；将独占设备改造为”共享”设备；实现了虚拟设备功能