**操作系统是一组能有效组织和管理计算机硬件和软件资源，合理地对各类作业进行调度，以及方便用户使用的程序的集合。**

1.1操作系统的目标和作用

目标：有效性、方便性、可扩充性、开放性

作用：os作为用户和计算机硬件之间的接口（三种方式来使用计算机：命令方式、系统调用和图形、窗口方式）os作为计算机系统的资源管理者（用户接口、存储器管理、设备管理、文件管理、处理机管理）

os实现了对于计算机资源的抽象

操作系统是一种\_\_\_\_\_\_\_\_\_\_。

A．通用软件 B．系统软件

C．应用软件 D．软件包

1.2操作系统的发展进程

无操作系统的计算机系统——单道批处理系统（联机批处理和脱机批处理、自动性、顺序性和单道性）——多道批处理系统（多道性、无序性、调度性）——分时系统——实时系统

单道和多道批处理的比较：

多道程序系统和多处理系统(Multi-Processing System)的区别：前者指多个程序同时在内存中交替运行，后者指多个处理器。——批处理：缺乏交互性

分时系统：多路性、独立性、及时性、交互性——应用在：实时控制、实时信息处理系统、多媒体系统、嵌入式系统

单用户单任务DOS——单用户多任务windows XP——多用户多任务

2、所谓\_\_\_\_\_\_\_是指将一个以上的作业放入主存，并且同时处于运行状态，这些作业共享处理机的时间和外围设备等其他资源。

A. 多重处理 B．多道程序设计

C. 实时处理 D．并行执行

3. \_\_\_\_\_\_\_操作系统允许用户把若干个作业提交给计算机系统。

A．单用户 B，分布式

C．批处理 D．监督

4.采用多道程序设计技术能充分发挥\_\_①\_\_与\_\_②\_\_并行工作的能力。

3、多道程序设计是指（）。

A 在实时系统中并发运行多个程序

B 在分布系统中同一时刻运行多个程序

C 在一台处理机上同一时刻运行多个程序

D 在一台处理机上并发运行多个程序

1、引入多道程序技术的前提条件之一是系统具有（）。

A 多个CPU B 多个终端 C 中断功能

D 分时功能

2、批处理系统的主要缺点是（）。

A CPU利用率低 B 不能并发执行 C 缺乏交互性 D 以上都不是

1.3操作系统的基本特征

①并发：并发性是指在一段时间内宏观上有多个程序在同时运行并发，微观上交替执行（单处理机），并行执行（多处理机）——指的是进程的并发

②共享：互斥共享和同时访问方式

③虚拟：时分复用技术和空分复用技术。设N是某物理设备所对应的虚拟的逻辑设备数，则：

采用时分复用时，每台虚拟设备的平均速度必然等于或低于物理设备速度的1/N。

采用空分复用时，每台虚拟设备平均占用的空间必然也等于或低于物理设备所拥有空间的1/N。

④异步：如果没有很好的同步机制，可能会导致程序执行结果不确定，不可再现。

1.4操作系统的主要功能

处理机管理：进程控制、进程同步、进程通信、调度

存储器管理：地址映射、内存分配、内存保护、内存扩充

设备管理：设备分配、设备处理、缓冲管理

文件管理：目录管理、文件存储空间管理、文件读写管理和保护

用户接口

1.5操作系统的结构设计

无结构os——模块化os——分层式os——现代结构（微内核os、客户端/服务器模式（用户态、内核态））

在\_\_\_\_\_\_\_操作系统控制下，计算机系统能及时处理由过程控制反馈的数据并作出响应。

A．实时 B．分时 C. 分布式 D．单用户

操作系统的\_\_\_\_\_\_\_\_\_\_管理部分负责对进程进行调度。A．主存储器 B．控制器 C．运算器 D．处理机

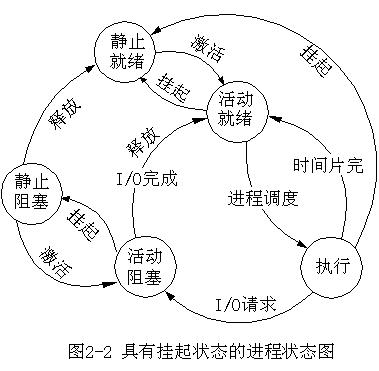
第二章进程管理

2.1进程的基本概念

顺序执行特征：顺序性、封闭性、可再现性

并发执行：间断性、失去封闭性、不可再现性

进程的定义和特征：程序段+相关数据段+PCB；动态性（！！进程最基本的特征）、并发性、异步、独立

进程的状态：三基本状态转换图

挂起状态：状态转换图：

1．在进程管理中，当\_\_\_\_\_\_\_\_\_\_时，进程从阻塞状态变为就绪状态。

A．进程被进程调度程序选中 B. 等待某一事件

C. 等待的事件发生 D. 时间片用完

1、进程的三个基本状态在一定条件下可以相互转化，进程由就绪状态变为运行状态的条件是\_\_①\_\_；由运行状态变为阻塞状态的条件是\_\_②\_\_。

A. 时间片用完 B．等待某事件发生

C．等待的某事件已发生 D．被进程调度程序选中

进程控制块：①PCB（当处理机被中断时，这些信息都必须保存到PCB中，以便该进程重新执行时，能从断点继续执行。 ）

②进程控制块的组织方式：链接方式和索引方式（就绪队列、阻塞队列）

2.2进程控制（最基本的功能、一般是原语操作、由os的内核完成）

操作系统内核；处理机执行状态分系统态和用户态

①创建进程

事件：用户登录；作业调度；提供服务；应用请求（用户进程自己创建）

创建过程：申请空白PCB——为新进程分配资源——初始化PCB——将新进程插入就绪队列

②终止进程

事件：正常结束；异常结束；外界干预

终止过程：根据标识，从PCB集合中找到该进程PCB，读进程状态——若为执行状态，立即终止执行。若还有子孙进程，终止其子孙进程——将所有资源归还给系统或父进程——将PCB从队列中移除

③进程状态转换

进程的阻塞和唤醒

事件：

阻塞过程：block原语，

立即停止执行——PCB将“执行”改为“阻塞”——将其插入阻塞队列——转进程调度程序，将处理机分配给某个就绪进程，并进行进程切换。

唤醒过程：

wakeup（）原语

将被唤醒进程的PCB从阻塞队列移出——将其PCB中进程状态由“阻塞”改为“就绪”——将改PCB插入到就绪队列中。

进程的挂起和激活

挂起过程：

suspend（）原语

检查被挂起的状态——将PCB复制到指定内存区域——若该进程正在执行，则转进程调度程序重新调度

激活过程：

active（）

将进程从外存调入内存——检查现行状态——若采用的是抢占式调度策略，则应检查被激活就绪进程的优先级，若其优先级比先行执行进程高，则应将处理机分配给被激活进程。

1、通常，用户进程被建立后，\_\_\_\_

A. 便一直存在于系统中，直到被操作人员撤销

B．随着作业运行正常或不正常结束而撤销

C．随着时间片轮转而撤销与建立

D．随着进程的阻塞或唤醒而撤销与建立

2、设系统中有n(n>2)个进程，且当前不在执行进程调度程序，试考虑下述4种情况，上述情况中，不可能发生的情况是\_\_\_\_\_\_\_ 。

A. 没有运行进程，有2个就绪进程，n-2个进程处于等待状态。

B.有1个运行进程，没有就绪进程，n-1进程处于等待状态。

C.有1个运行进程，有1个就绪进程，n-2进程处于等待状态。

D.有1个运行进程，n-1个就绪进程，没有进程处于等待状态；

2.3进程同步

1.基本概念：两种制约关系：直接制约和间接制约（进程互斥关系）

临界资源（一段时间内只允许一个进程访问的资源）

2.经典的生产者——消费者问题

produce（）{

while（true）{

produce an item in nextp;

wait（empty）;

buffer[in]=nextp;

in=(in+1)%n;

signal(full);

}

}

consumer（）{

wait (full);

nextc=buffer[out];

out=(out+1)%n;

signal(empty);

consume an item in nextc;

}

临界区：每个进程中访问临界资源的那段代码

所有同步机制都应遵循如下四条准则： 空闲让进，忙则等待，有限等待，让权等待

1、下面关于临界区的叙述中，正确的是\_\_\_\_\_\_\_\_。

A. 临界区可以允许规定数目的多个进程同时执行

B. 临界区只包含一个程序段

C. 临界区是必须互斥地执行的程序段

D. 临界区的执行不能被中断

2、两个进程合作完成一个任务。在并发执行中，一个进程要等待其合作伙伴发来消息，或者建立某个条件后再向前执行，这种制约性合作关系被称为进程的\_\_\_\_\_\_\_\_。

A . 同步 B. 互斥 C. 调度 D. 执行

3、若系统中有五个并发进程涉及某个相同的变量A，则变量A的相关临界区最少是由\_\_\_\_\_\_\_\_临界区构成。

A. 2个 B. 3个 C. 4个 D. 5个

信号量机制：

整型信号量（未遵循让权等待）——记录型信号量——AND型信号量——信号量集

2.4经典进程同步问题

①生产者-消费者问题

一个生产者、一个消费者、一个缓冲区：empty=1；full=0（表示有1个空缓冲区，表示开始时没有空缓冲区）

②读者-写者问题

一个读者，多个写者 S.value=1

reader（）{

wait（s）

read data

signal（s）

}

writer（）{

wait（s）；

write

signal（s）

}

多个读者，多个写者ns.value=1,s.value=1

reader（）{

wait(ns)

if(num==0)

wait（s）

num++

signal(ns)

read data

wait(ns)

num--

if(num==0)

signal（s）

signal(ns)

}

writer（）{

wait（s）；

write

signal（s）

}

③哲学者进餐问题

可采取的解决方法：至多允许4个哲学家同时取左边的筷子；

仅当哲学家左右两只筷子均可用时，才允许他拿起筷子进餐；规定奇数号哲学家先拿左边筷子，然后再拿右边筷子；而偶数号哲学家先拿右边筷子，然后再拿左边筷子。

(1)第一种情况

第i个哲学家的活动可以描述为：

chopsticks[i].value=1;s.value=4

do{

wait（s）

wait（chopsticks[i]）;

wait （chopsticks[i+1]%5）;

...

//eat

...

signal (chopsticks[i]);

signal (chopsticks[i+1]%5);

...

signal(s);

//think

}while(true)

（2）第二种情况

chopsticks[i].value=1

do{

...

//think

...

Swait(chopsticks[i+1]%5,chopsticks[i]);

...

//eat

...

Signal(chopsticks[i+1]%5,chopsticks[i]);

...

}while(true)

(3)第三种情况

奇数号哲学家先拿其左边的筷子，然后再那其右边的筷子；而偶数号哲学家先拿其右边的筷子，然后再那其左边的筷子。

算法：

Var Chopstick:array[0…4] of semaphore:=1,1,1,1,1;

Parbegin

Chopsticki:repeat

If (i mod 2 !=0) then

begin（先左后右）

Wait(chopstick[i]);

Wait(chopstick[(i+1) mod 5]);

End

Else

begin（先右后左）

Wait(chopstick[(i+1) mod 5]);

Wait(chopstick[i]);

End

…

//Eat;

…

signal(chopstick[i]);

signal(chopstick[(i+1) mod 5]);

//think;

until false;

parend;

2.6进程通信

高级通信：指用户可直接利用OS所提供的一组通信命令，高效地传送大量数据的一种通信方式。

进程通信方式：共享存储器通信；管道通信；消息传递通信（直接通信和间接通信——信箱通信）；客户机-服务器方式（套接字，远程过程调用）

2.7线程——作为分配CPU的基本单位，是花费最小开销的实体

线程只作为调度和分派的基本单位；进程资源分配的基本单位

一个进程通常包含多个线程

线程的属性：轻型实体，独立调度和分派的基本单位；可并发执行；共享进程资源

TCB和线程状态：执行——就绪——阻塞

线程间的同步与通信：互斥锁；条件变量；信号量机制

线程的实现：用户级线程（线程管理的全部工作由应用程序来做，在用户空间内实现，内核是不知道线程的存在的）；内核级线程；混合式线程

在操作系统中，进程是一个具有一定独立功能的程序在某个数据集上的一次 ＿＿＿＿＿＿＿＿＿＿。

A．等待活动 B．运行活动

C．单独操作 D．关联操作

多道程序环境下，操作系统分配资源以＿＿＿＿＿＿＿为基本单位

A．程序 B．指令 C进程 D．作业

第三章 处理机调度与死锁

3.1处理机调度的基本概念

高级、中级和低级调度

①高级调度：用于决定外存上后备队列中哪些作业调入内存，并为它们创建进程、分配必要的资源，然后将新创建的进程插入到就绪队列中，准备运行。

——接纳多少作业？接纳哪些作业？（调度算法）

作业

作业步：每个作业在运行期间，都必须经过若干个相对独立，又相互关联的顺序加工步骤，其中每一个加工步骤成为作业步。

作业控制块（JCB）

②低级调度：用来决定就绪队列中的哪个进程应获得处理机，然后再由分派程序执行把处理机分配给该进程的具体操作。

两种调度方式：抢占式调度（优先权原则、短作业优先、时间片原则）和非抢占式

③中级调度：挂起和激活，存储器管理中的对换功能。为了提高内存的利用率和系统的吞吐量。

④：三种调度比较：进程调度的运行频率最高 、作业调度频率最低 。

调度算法的目标：

（1）处理机调度算法的共同目标——资源利用率、公平性、平衡性、策略强制执行、

（2）各类系统的目标——周转时间短（评价批处理的准则：周转时间和带权周转时间）、响应时间快（分时系统）、截止时间的保证（实时系统的准则）

3.2调度算法

先来先服务调度算法——（比较适合于长作业和计算型cpu忙碌性作业）

短作业（进程）优先调度算法

高优先权优先调度算法——为了照顾紧迫性作业

优先权进程调度算法的类型——优先权类型：静态优先权、动态优先权

高响应比优先调度算法——响应比=（要求服务时间+等待时间）/要求服务时间

基于时间片的轮转调度算法

多级反馈队列调度算法——不必事先知道各个进程所需的执行时间，而且可以满足各种类型进程的需要

3.3 产生死锁的原因和必要条件

资源包括：可抢占式资源（可剥夺性资源）和不可抢占式资源

产生死锁的原因：

①竞争资源

②进程推进顺序不当

1.竞争资源引起的死锁

（1）竞争非剥夺性资源

（2）竞争临时性资源（临时性资源是指由一个进程产生的被另一进程使用一短暂时间后便无用的资源。也称可消耗性资源。如消息,中断信号，同步信号等）

2.进程推进顺序不当引起的死锁

（1）进程推进顺序合法

（2）进程推进顺序非法

产生死锁的必要条件：互斥条件、请求和保持条件、不可抢占条件、循环等待条件——只要有一个条件不满足，都不会形成“死锁”

3.4 预防和避免死锁的方法

3.5 死锁的检测和解除

处理死锁的基本办法：

预防死锁：事先预防策略，去破坏产生死锁的四个必要条件

避免死锁：事先预防，是在资源的动态分配过程

检测死锁：事后

解除死锁：事后。实现难度最大——撤销一些进程，收回他们的资源

①预防死锁

破坏“互斥”条件

破坏“请求和保持”条件：预分配资源策略；空手申请资源策略

破坏“不可抢占”条件：两种方式：方式一：申请资源，有则分配，无则释放进入等待状态；方式二：申请资源，没有资源则检查其他进程，若处于等待资源，则抢占，若没有等待，则自己处于等待资源状态。

破坏“循环等待”条件：采用 有序资源分配策略（所有的系统资源按类型进行线性排队，并赋予不同的序号）——一旦新设备出现，会导致混乱

②避免死锁——银行家算法：在资源动态分配过程中，若分配后系统状态仍是安全的，则同意分配，否则将拒绝分配，这样可防止系统进入不安全状态，从而避免死锁。

安全状态；不安全状态

\*并非所有不安全状态都是死锁状态，但它迟早会进入死锁状态。

\*只要系统处于安全状态，便可避免进入死锁状态。

安全性算法：

银行家算法：

第四章 存储器管理

4.1存储器管理功能

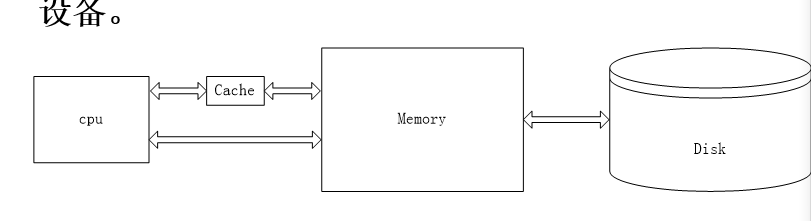
①内存管理（内存分配：是操作系统从空闲空间中划分出适当的空间给进程）——连续内存分配+离散内存分配）

②内存保护

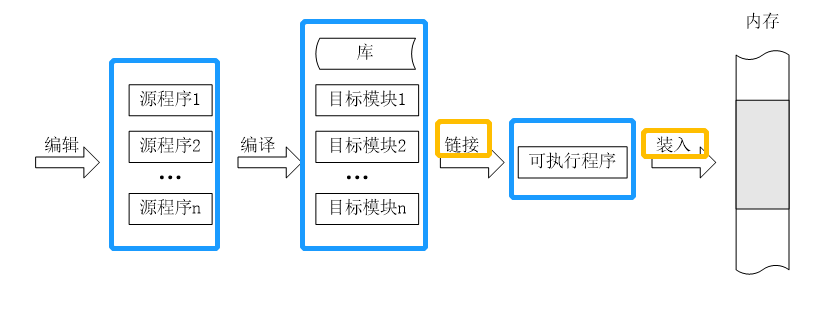
③地址映射（内存中每个存储单元都有一个与之对应的物理地址，程序编译时采用逻辑地址。运行程序时，需要将程序逻辑地址转换为内存物理地址。

）——逻辑地址：目标代码的相对编址；物理地址：内存存储单元的编址；重定位：逻辑地址转换为物理地址的操作——>若不进行地址转换，将会发生运行错误。

存储器概述：

内存——外存——cache——寄存器

④内存扩充

4.2 程序的装入和链接

4.2 程序的装入和链接

4.2.1程序的链接

静态链接（先将目标模块及他们所需的库函数链接成一个完整的装配模块，以后不再拆开）——须解决以下两个问题：将相对地址进行修改；变换外部调用符号

装入时动态链接（装入内存时，边装入边链接的方式）——便于修改和更新，便于实现对目标模块的共享

运行时动态链接（适用于大型软件）

4.2.2程序的装入

绝对装入方式（知道程序将驻留在内存的什么位置）

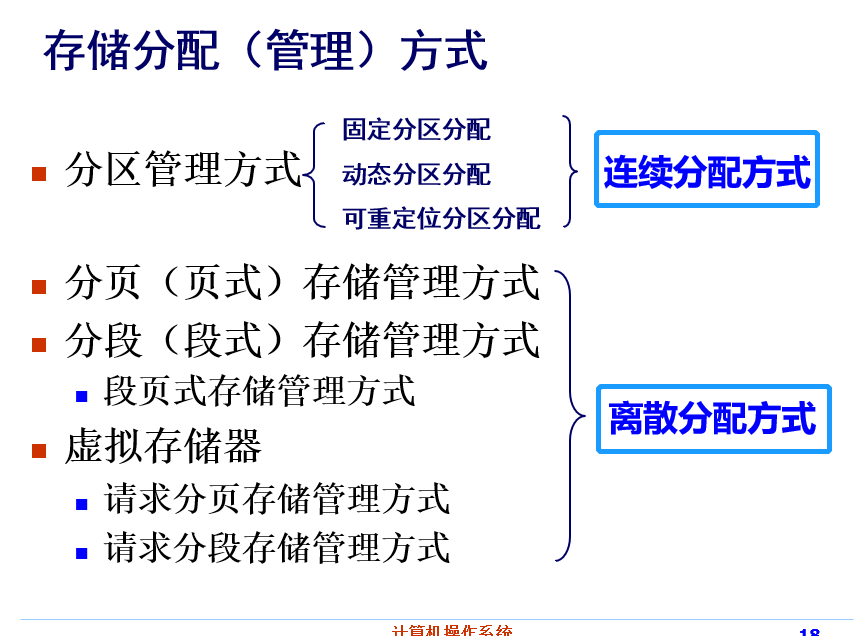
可重定位装入方式

动态运行时的装入方式（重定位寄存器）

分配方式：

连续分配：为作业（进程）分配地址连续的存储空间

离散分配：为作业（进程）分配不连续存储空间

存储分配方式：

4.3连续分配方式

单一连续分配方式——单用户单任务 系统区+用户区

①固定分区分配（分区大小相等、分区大小不等）——分配+回收——算法实现简单，但是存在内碎片浪费

②动态分区分配（事先不划分分区，待作业需要分配时，再按照需要划分）

数据结构+分配算法+分配及回收操作

数据结构：空闲分区表（分区号+分区起始地址+分区大小），空闲分区链（自身不占用存储空间，但是管理困难）

分区分配算法：

（1）基于顺序搜索的动态分区分配算法

首次适应算法（空闲分区链按地址递增排序。分配时从链首开始顺序查找，直至找到一个大小能满足要求的空闲分区）

循环首次适应算法（为进程分配内存时，不再是每次从链首开始查找，而是从上次找到的空闲分区的下一个空闲分区开始查找）

最佳适应算法（将空闲分区按其容量从小到大顺序排列，总是将能满足要求的最小分区分配给请求者。）

最坏适应算法（将空闲分区按其容量从大到小顺序排列，总是将能满足要求的最大分区分配给请求者。）

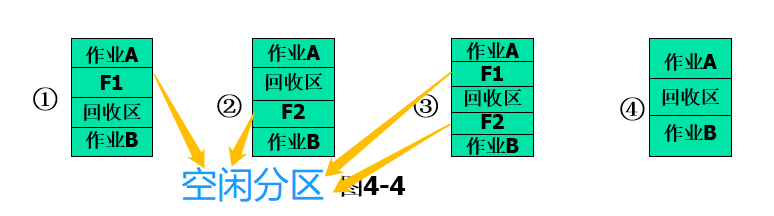
（2）基于顺序搜索的动态分区分配算法

快速适应算法

伙伴系统

哈希算法

分配及回收操作：

回收内存：四种情况

③动态重定位分区分配

引入：移动作业-紧凑

\*通过作业移动将原来分散的小分区拼接成一个大分区。

\*每次紧凑后，都必须对移动了的程序或数据进行重定位。

重定位的实现：基址寄存器+限长寄存器

动态重定位分区分配算法：与动态分区分配算法基本相同，只是增加了紧凑功能（当找不到满足请求的空闲区，但空闲区总和大于等于请求数时，进行紧凑操作。）

把作业地址空间中使用的逻辑地址变成内存中物理地址的过程称为\_\_\_\_\_\_\_\_\_\_\_\_。

4.2.5对换

①对换的引入

整体对换（以整个进程为单位）+部分对换（页面对换、分段对换）

②对换空间的管理

把外存分为 文件区（离散分配）和对换区（连续分配方式）

数据结构：空闲区表或空闲区链

分配算法：首次适应、循环首次适应

③进程的换出与换入

换出过程：

\*选择处于阻塞状态且优先级别最低的进程作为换出进程。

\*启动磁盘，将该进程的程序和数据传送到磁盘的对换区上；

\*若传送过程未出现错误，回收该进程所占用的内存空间，并对该进程的PCB做相应的修改。

换入：处于“就绪”状态但已换出的进程，将其中换出时间最久的进程换入，

4.3 基本分页存储管理方式

4.3.1页面和页表

（1）页面和物理块

将逻辑地址空间分成若干大小相等的片，称为页面或页(page)。页号从0开始。

内存空间分成与页大小相同的若干存储块，称为块或页框(frame)。也从0开始编号。——最后一个页不满，称为页内碎片

（2）地址结构：页号+页内地址

若逻辑地址为A,页面大小为L,则页号P和页内地址d可按下式求得：

P=int(A/L)

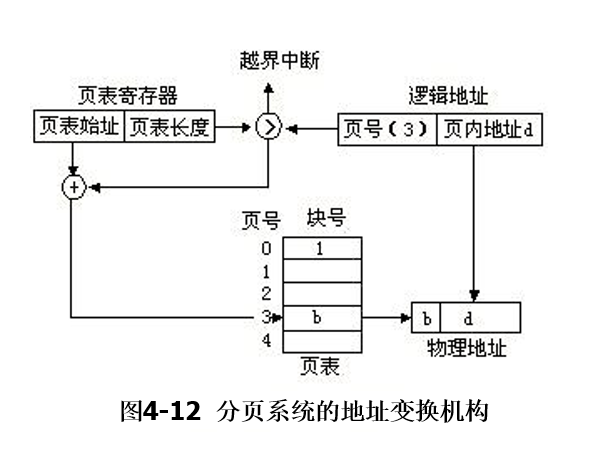
d=A mod L

（3）页表——实现了从页号（用户程序）到物理块号（内存）的地址映射

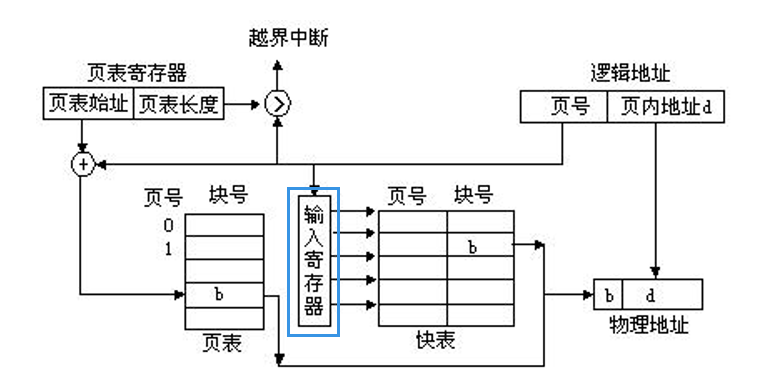
4.3.2地址变换机构

（1）基本的地址转换机构（页表存放在内存中）

逻辑地址变换：先查 页表寄存器（页表始址+页表长度）——进行变换



（2）具有快表的地址转换机构



4.3.3两级和多级页表

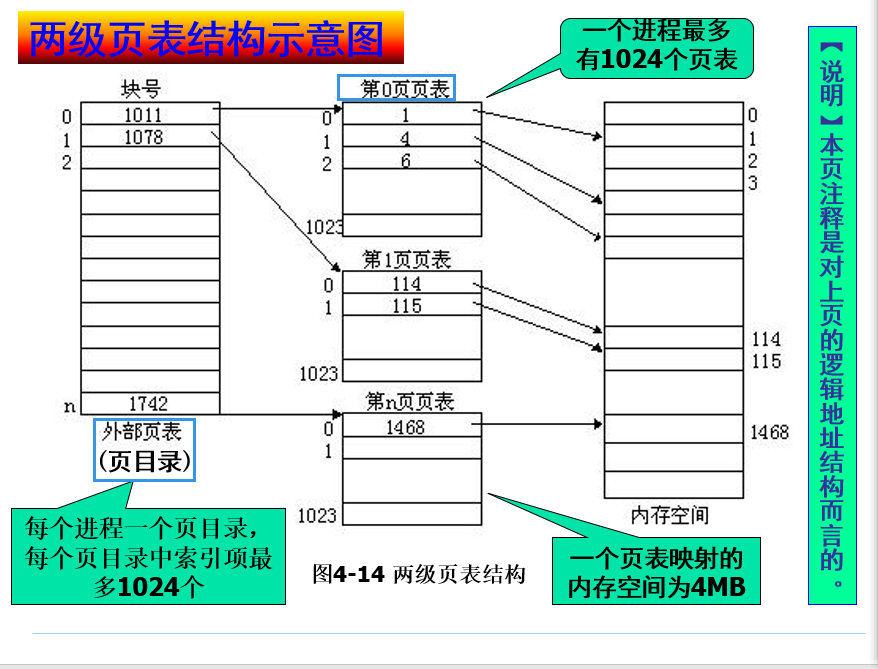
①采用离散分配方式来解决难于找到一块连续的大内存空间的问题；

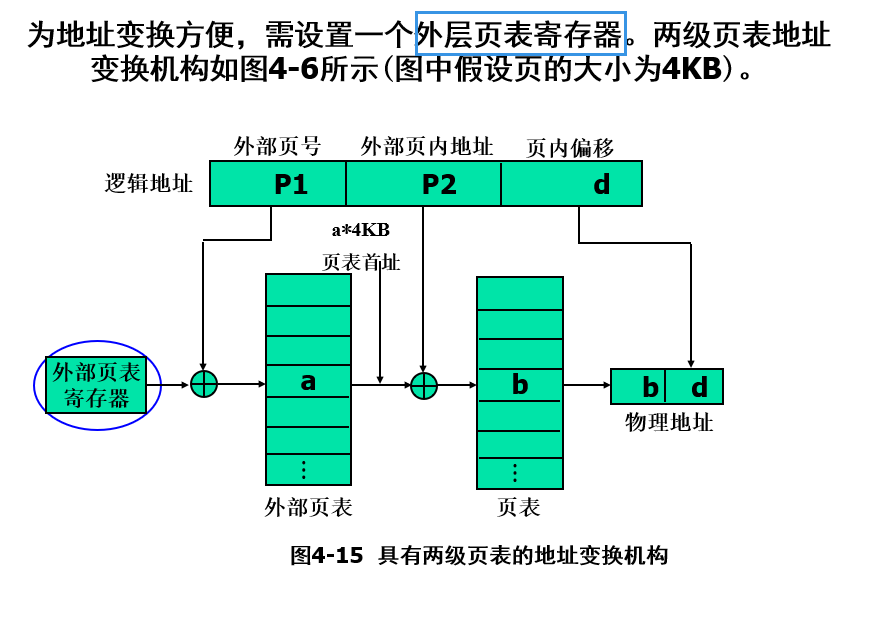
办法：两级和多级页表

②只将当前需要的部分页表项调入内存，其余的页表项驻留在磁盘上，需要时再调入；

——①两级页表

逻辑地址结构：外层页号+外层页内地址+页内地址





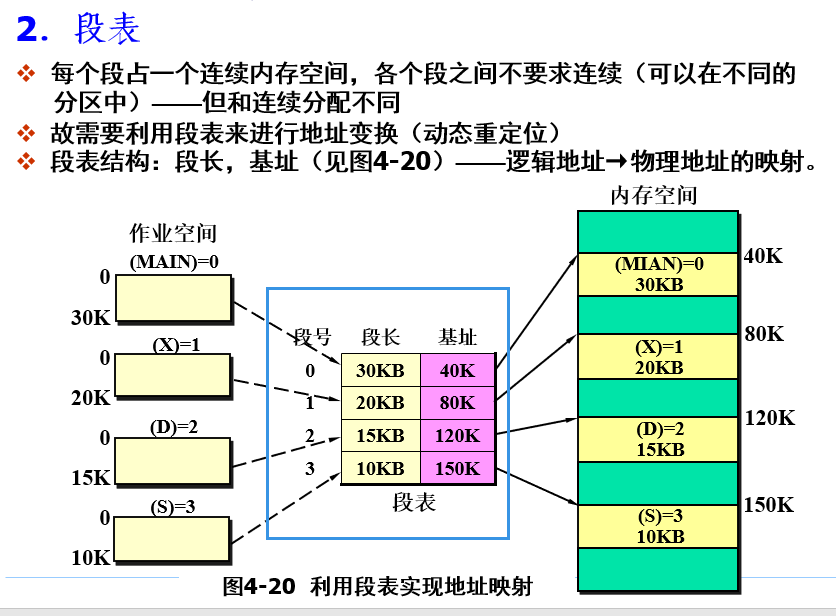
不足：不利于内存信息的共享和保护

4.4 基本分段存储管理方式

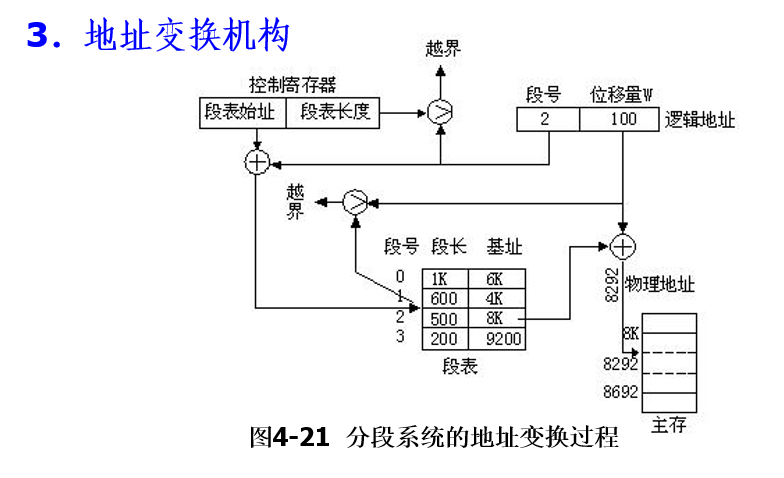
4.4.1分段存储管理方式的引入

分段：段号+段内地址

段表：



地址变换机构：分段共享



分页与分段的不同：

1.信息单位不同

页是信息的物理单位，段则是信息的逻辑单位。

2.大小不同

页的大小固定，而段的大小不固定。

3.维数不同

分页存储管理方式的程序逻辑地址空间是一维的，而分段存储管理方式的程序逻辑地址空间是二维的。

4.4.4段页式存储管理方式

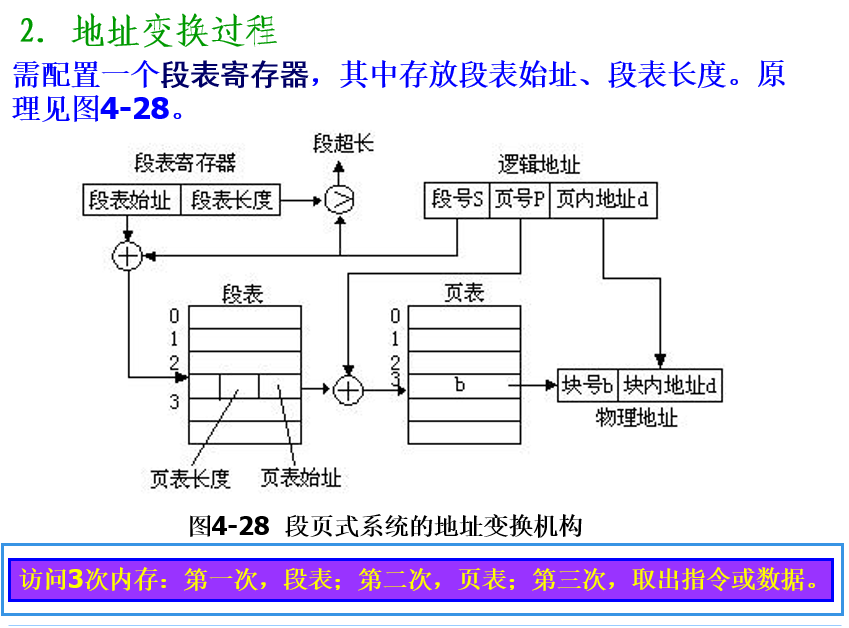
——取长补短：因为分页式能够提高内存的利用率+分段式能很好的满足用户的需要

1. 基本原理

先把用户程序分成若干段，再把段分成若干页。

|  |  |  |
| --- | --- | --- |
| **段号S** | **段内页号P** | **页内地址W** |

1. 地址变换过程



4.5虚拟存储器的基本概念

（1）常规存储器管理方式的特征——一次性+驻留性

（2）局部性原理

时间局部性——循环程序

空间局部性——顺序执行

（3）虚拟存储器的定义

原理：部分装入+请求调页+页面置换

4.5.2实现方法

①分页请求系统：在分页系统基础上，增加了请求调页和页面置换功能

必须提供硬件和软件功能：请求分页的页表机制+缺页中断机构+地址变换机构——请求调页、实现页面置换的软件

②请求分段系统：

请求分段的段表机制+缺页中断机构+地址变换机构——请求调段、实现段置换的软件

4.5.3虚拟存储器的特征

多次性——对换性——虚拟性

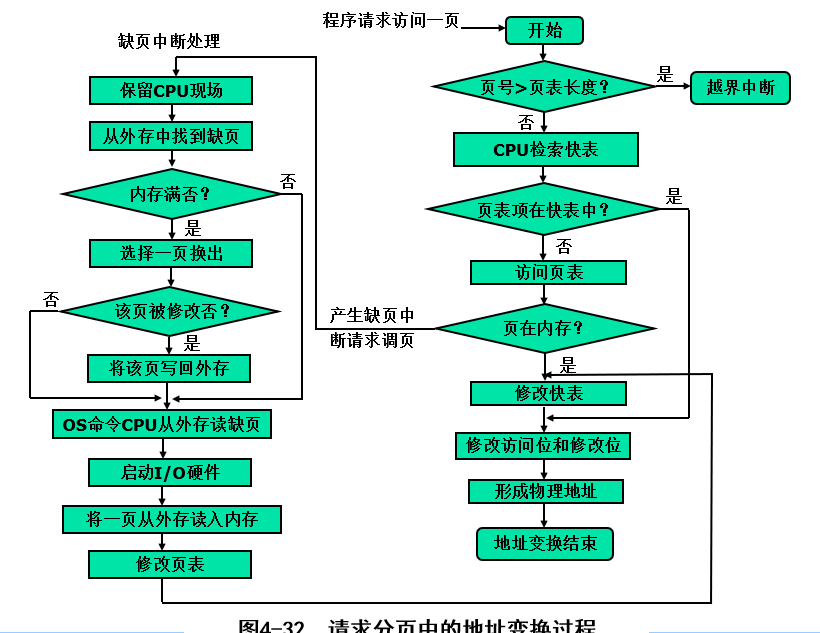
4.6请求分页存储管理方式

4.6.1请求分页中的硬件支持

1.页表机制

2.缺页中断机构——在指令执行期间产生和处理

3.地址变换机构



4.6.2内存分配策略和分配算法

为进程分配内存时，涉及到以下问题：

最小物理块数的确定

物理块的分配策略——固定分配局部置换，可变分配全局置换，可变分配局部置换

物理块分配算法

4.6.3调页策略

何时调入页面——预调页策略，请求调页策略

从何处调入页面——

系统有足够的对换区——全部从对换区调入

* + 系统缺少对换区——全部从文件区调入
  + UNIX方式——凡未运行的页面，从文件区调入；曾经运行过又被调出的页面，从对换区调入。共享页面有时无须调入

页面调入过程

4.7页面置换算法

* + - 最佳置换算法：**选择以后永不使用的或者是未来最长时间内不再使用的页面淘汰**
    - **先进先出算法：选择在内存中驻留时间最长的页面淘汰。 设置一个指针，指向最老的页面。**
    - **最近最久未使用算法：用“最近的过去”作为“最近的将来”的近似。**
    - **CLOCK算法：设置一个指针pointer，用于指示页面查找的起始位置，置换页面时，从指针位置开始，查看指针所指页面的访问位是否为1，如果是1，则将该页面的访问位修改为0。置换后，指针指向下一位**

“抖动”与工作集

1. 多道程序度与抖动

如果多道程度过高，页面在内存与外存之间频繁调度，以至于调度页面所需时间比进程实际运行的时间还多，此时系统效率急剧下降，甚至导致系统崩溃。这种现象称为颠簸或抖动

2. 产生“抖动”的原因

同时运行的进程数过多，进程频繁访问的页面数高于可用的物理块数，造成进程运行时频繁缺页

3.缺页率与系统为进程分配物理块的多少有关

4.降低缺页率，防止抖动的内存管理方法：——工作集策略

工作集是指在某段时间间隔 ∆ 里，进程实际要访问的页面的集合

——虽然程序只需要少量的几页在内存便可运行，但为了较少地产生缺页，应将程序的全部工作集装入内存中

5.“抖动”的预防方法

采取局部置换策略

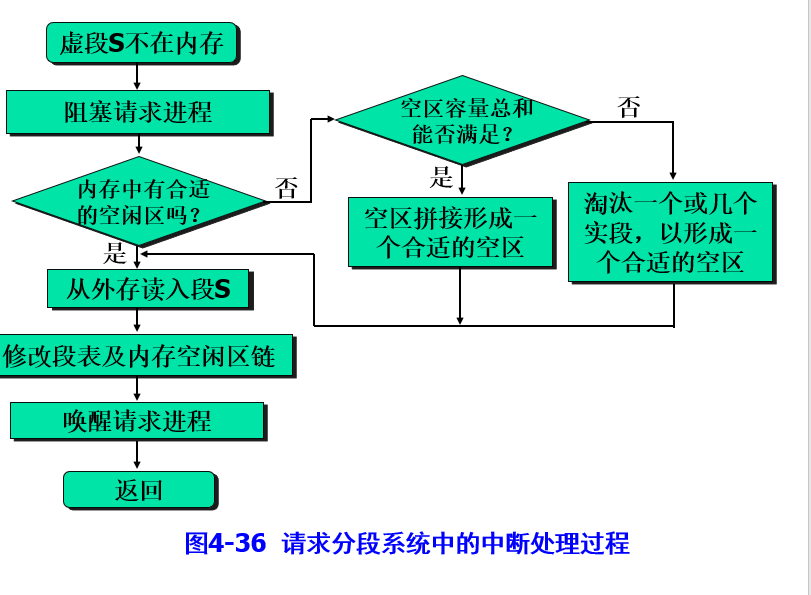
把工作集算法融入到处理机调度中

利用“L=S”准则调节缺页率

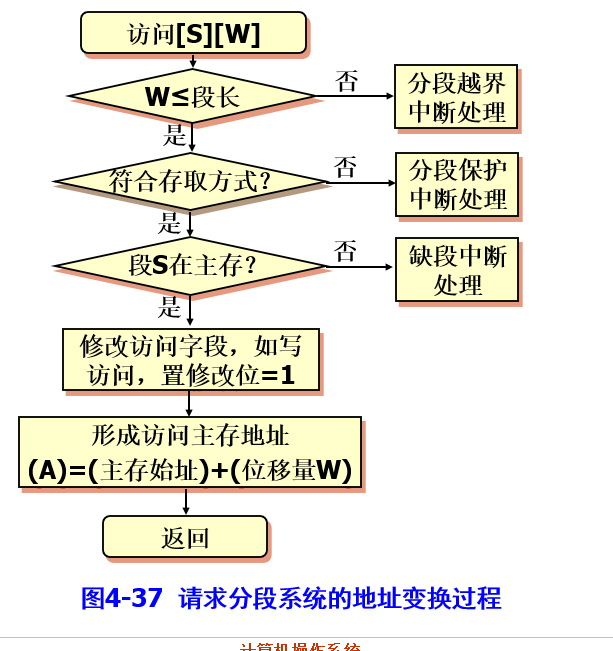
选择暂停的进程

4.8.1 请求分段中的硬件支持

段表机制

缺段中断机机构——缺段中断的处理比缺页中断复杂

地址变换机构

4.8.2分段的共享和保护

分段保护

1) 越界检查  
　　2) 存取控制检查  
　　3) 环保护机构  
第五章 设备管理

5.1 I/O系统的功能、模型和接口

I/O系统管理的 主要对象 是I/O设备和相应的设备控制器。

5.1.1I/O系统的基本功能

（1）隐藏物理设备的细节

（2）与设备的无关性

（3）提高处理机与I/O设备的利用率

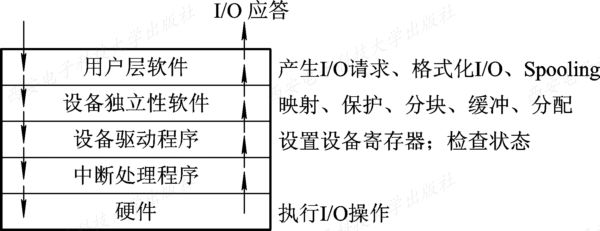
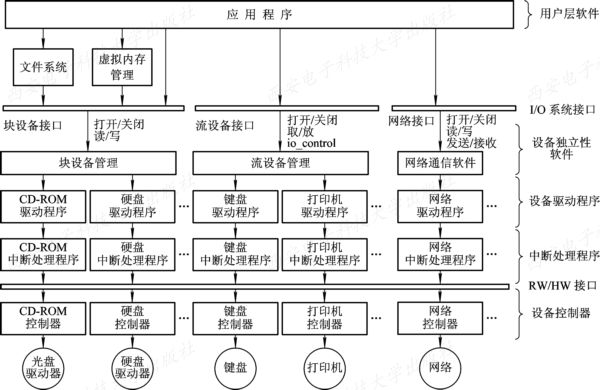
（4）对I/O设备进行控制——是驱动程序的功能。四种控制方式：

采用轮询的可编程I/O方式；采用中断的；直接存储器访问方式；I/O通道方式

（5）确保对设备的共享——共享设备；独占设备

（6）错误处理——临时性错误和持久性错误

5.1.2I/O系统的层次结构和模型

5.1.3I/O系统接口——块设备接口、流设备接口和网络接口

（1）块设备接口——块设备（传输是以数据块为单位的设备，采用的是DMA方式）

隐藏了磁盘的三维结构（盘面号，磁道号，扇区号）将抽象命令映射为底层操作

（2）流设备接口——字符设备，数据的存取、传输是以字符为单位的设备，不可寻址、采用顺序存取方式。字符设备输入输出时，常采用中断驱动的方式；get和put操作；in-control指令。用于统一管理各种类型的字符设备。

（3） 网络通信接口

5.2I/O设备和设备控制器



**主机I/O系统（通道I/O系统结构）**

* + **I/O设备**
  + **设备控制器**
  + **I/O通道**
  + **计算机**

**微机I/O系统：CPU不直接与I/O通信，是通过设备控制器**

I/O系统：**是用于实现数据的输入、输出以及数据存储的系统**

* + - 包括：**直接用于I/O和存储数据的设备**
    - **相应的设备控制器**
    - **I/O通道（I/O专用处理机）——大、中型计算机系统中**

**设备管理的对象**

* + **I/O设备、设备控制器、I/O通道。**

I/O设备的分类：

（1）从资源分配角度分类: 独占设备 共享设备 虚拟设备

（2）按设备的使用特性分类 存储设备 I/O设备

（3）按信息交换方式分类 块设备 字符设备

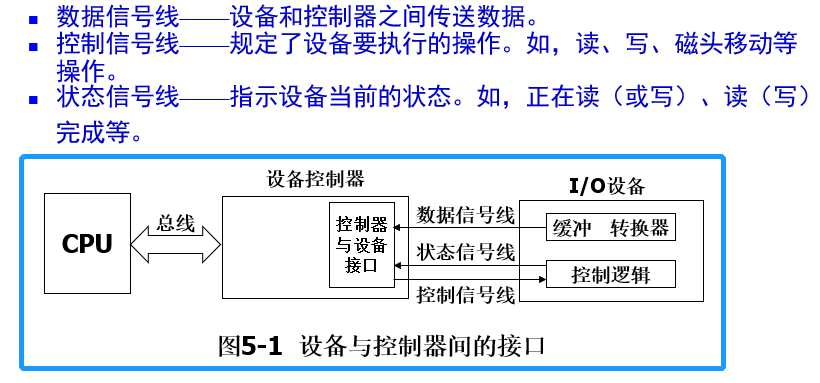
（4）从数据传输率分：低速，中速，高速（磁盘机、磁带机）

* + 按信息交换方式分类：块设备

以数据块为单位存储、传输信息。例如： 硬盘

* + 字符设备（慢速字符设备）

以字符为单位存储、传输信息。例如：终端显示设备。

1. 设备与控制器之间的接口——**该接口中包含3类信号：数据信号、控制信号、状态信号**
2. 设备控制器

**I/O设备一般由电子和机械两部分组成**

**设备控制器：**电子部分叫做设备控制器，完成设备与主机之间的通信

**物理设备：**机械部分是设备本身

1. 通道——代替CPU，负责对输入、输出操作进行控制，从而使得输入、输出操作与CPU并行操作。（**指令单一（与I/O操作的有关指令）、没有自己的内存，它与CPU共享内存**

通道的分类——

1. **字节多路通道**

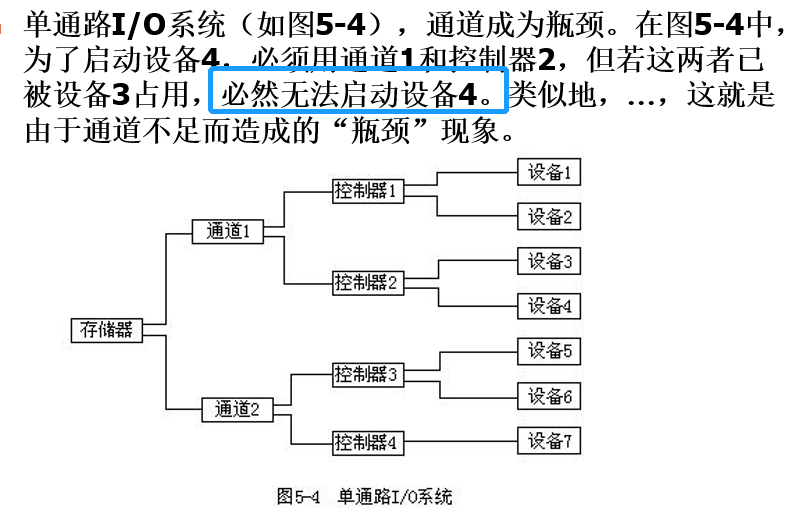
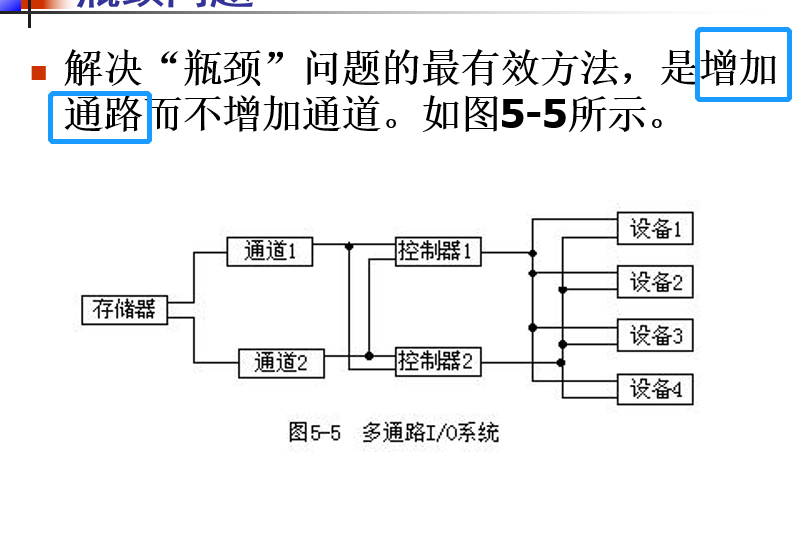
以字节为单位交叉传输，当一台传送一个字节后，立即转去为另一台传送字节（**时间片方式工作）**

1. **数组选择通道**

每次传送一批数据，在一段时间内只能执行一个通道程序，只允许一台设备进行数据传输（独占性

**（3）数组多路通道**

结合了选择通道传送速度高和字节多路通道能进行分时并行操作

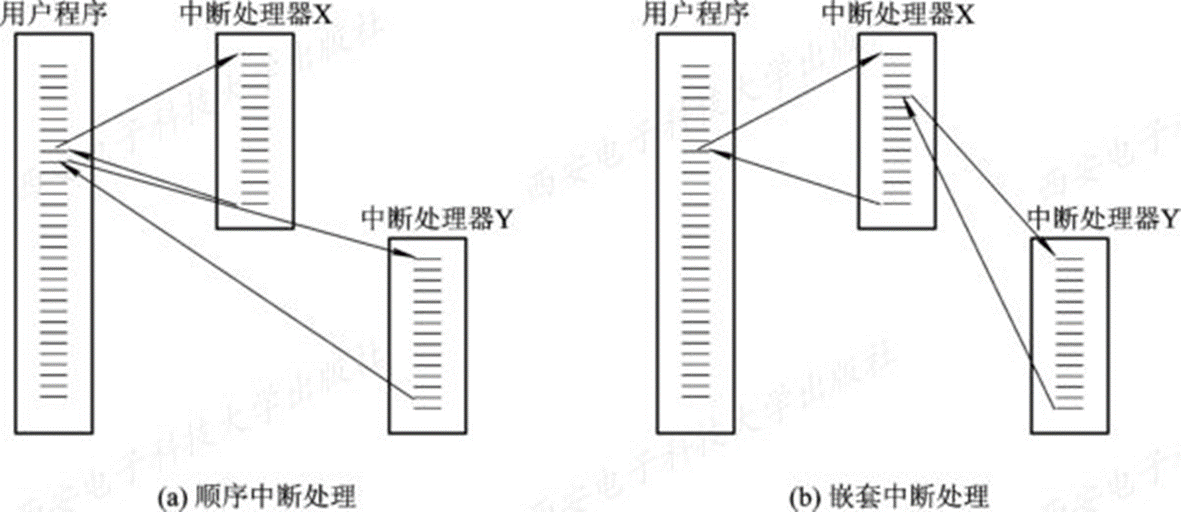
1. 瓶颈问题

**5.3 中断机构和中断处理程序**

5.3.1 中断简介

1. 中断和陷入  
　（1）中断 中断指CPU对I/O设备发来的中断信号的一种响应。  
　（2）陷入 由CPU内部的事件而引发的中断。

2. 中断向量表和中断优先级  
　（1）中断向量表  
　（2）中断优先级

3. 对多中断源的处理方式   
　（1） 屏蔽(禁止)中断  
　（2） 嵌套中断

5.3.2**中断处理程序**

**来了一个中断请求信号，**唤醒被阻塞的驱动程序进程——>对被中断进程的CPU环境进行保护——分析中断原因，转入相应的中断处理程序——恢复被中断进程的现场。

5.4 设备驱动程序

5.4.1设备驱动程序的概述

1. 设备驱动程序的功能   
　　(1) **接收**由与设备无关的软件发来的**命令和参数**，并将命令中的抽象要求转换为与设备相关的低层操作序列。  
　　(2) **检查用户I/O请求的合法性**，了解I/O设备的工作状态，传递与I/O设备操作有关的参数，设置设备的工作方式。  
　　(3) **发出I/O命令**，如果设备空闲，便立即启动I/O设备，完成指定的I/O操作；如果设备忙碌，则将请求者的请求块挂在设备队列上等待。  
　　(4) **及时响应由设备控制器发来的中断请求**，并根据其中断类型，调用相应的中断处理程序进行处理**。**

2. 设备驱动程序的特点   
　　(1) 驱动程序是实现在与**设备无关的软件和设备控制器之间通信和转换**的程序，

(2) 驱动程序与设备控制器以及I/O设备的硬件特性紧密相关，对于不同类型的设备，应配置不同的驱动程序。但可以为相同的多个终端设置一个终端驱动程序。  
　　(3) 驱动程序与I/O设备所采用的I/O控制方式紧密相关，常用的I/O控制方式是**中断驱动和DMA方式。**　　(4) 由于驱动程序与硬件紧密相关，因而其中的一部分必须用汇编语言书写。目前有很多驱动程序的基本部分已经固化在ROM中。  
　　(5) 驱动程序应允许**可重入**。一个正在运行的驱动程序常会在一次调用完成前被再次调用。

5.4.2设备驱动程序的处理过程——任务是：启动设备，完成上层的I/O工作

(1)将抽象要求转换为具体要求。

(2)对服务请求进行校验。

(3)检查设备的状态。

(4)传送必要的参数。

(5)启动I/O设备。

5.4.3I/O控制方式

* + **可编程I/O、程序控制I/O方式**
  + **中断驱动I/O方式**
  + **直接存储器访问控制(DMA)**
  + **I/O通道控制方式**

1. 程序IO方式

发送I/O命令后，CPU一直检测设备的状态busy是否为0，为0，将数据从控制器中取走。

由于CPU的高速性和I/O设备的低速性， 使CPU的绝大部分时间都处于等待I/O设备完成数据I/O的循环测试中

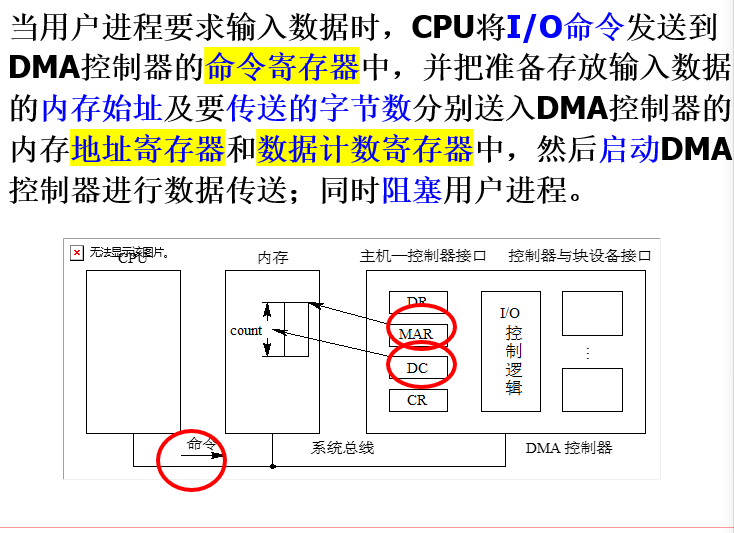
1. 中断驱动方式

发送I/O命令后，CPU继续执行指令。当外设完成后，会发送中断信号告知系统。

每传送几个字节（即数据寄存器每满一次），就请求一次中断，若传输大批数据，CPU需花费大量开销处理频繁的I/O中断。

1. 直接存储器访问I/O控制方式

在传送一个数据块期间，数据寄存器每满一次，由DMA控制器挪用CPU工作周期，占用系统总线，将数据寄存器的数据传输到内存。

DMA控制方式工作原理：

DMA控制方式，仅在传送一个数据块的开始和结束时，需要CPU干预。期间数据传送在DMA控制下直接完成

4.I/O通道控制方式

(4) 通道程序结束位P。该位用于表示通道程序是否结束。**P=1表示本条指令是通道程序的最后一条指令。**

(5) 记录结束标志R。**R=0表示本通道指令与下一条指令所处理的数据是同属于一个记录；R=1表示这是处理某记录的最后一条指令**。

5.5与设备无关的I/O软件

5.5.1与设备无关的I/O软件

设备独立性（设备无关性）基本含义：应用程序中所用的设备，不局限于使用某个具体的物理设备。

5.5.2 与设备无关的软件

**设备独立性软件的主要功能：**

1. 设备驱动程序的统一接口

2. 缓冲管理

3. 差错控制

4. 对独立设备的分配与回收

5. 独立于设备的逻辑数据块

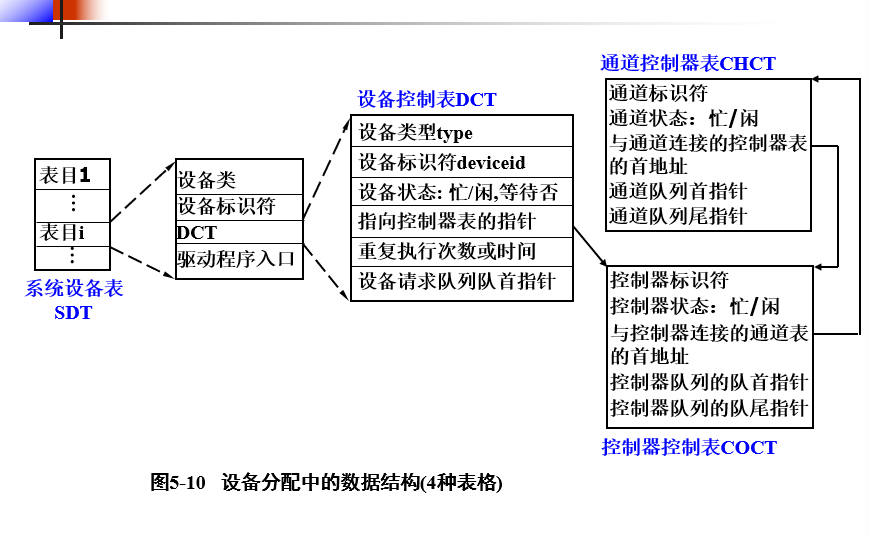
5.5.3设备分配

1 设备分配中的数据结构

设备控制表DCT

控制器控制表COCT

通道控制表CHCT

系统设备表SDT

2设备分配时应该考虑的因素

（1设备的固有属性

（2设备分配算法——先来先服务；优先级高者优先

（3设备分配中的安全性

安全分配方式——进程提出I/O请求后即阻塞，屏弃“请求和保持”条件

不安全分配方式——发出I/O请求后仍继续运行——有可能死锁

1. 独占设备的分配程序

1）基本的设备分配程序

按下述步骤进行设备分配（利用SDT、DCT、COCT、CHCT）：

1）分配设备

2）分配控制器

3）分配通道

（2）设备分配程序的改进

增加设备独立性——进程应使用逻辑设备名请求I/O

考虑多通路情况

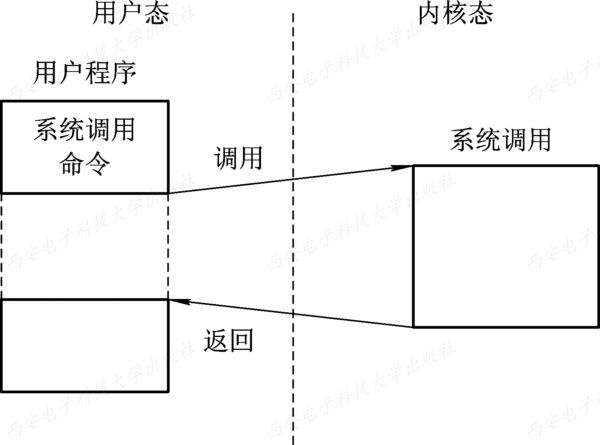
5.5.4 逻辑设备名到物理设备名映射的实现

逻辑设备表LUT：逻辑设备名、物理设备名、设备驱动程序入口地址。

5.6用户层的I/O软件

**5.6.1 系统调用与库函数**

1.系统调用

应用程序可以通过它间接调用OS中的I/O过程，对I/O设备进行操作。

**5.6.2 SPOOLing技术——脱机输入、输出方式**

**利用专门的外围机，将低速I/O设备上的数据传送到高速磁盘上；或者相反。**

**2. SPOOLing系统的组成**

SPOOLing系统主要有以下四部分组成：

* 输入井和输出井——**输入井模拟脱机输入时的磁盘设备，用于暂存I/O设备输入的数据。**
* 输入缓冲区和输出缓冲区——**输入缓冲区（内存中开辟的两个缓冲区）用于暂存由输入设备送来的数据，以后再送入输入井。**
* 输入进程SPi和输出进程Spo——**输入进程（又叫预输入进程）模拟脱机输入时的外围控制机，将用户要求的数据从输入设备通过输入缓冲区再送到输入井。当CPU需要输入数据时，直接从输入井读入内存。**
* 井管理程序——当作业执行过程中，向某台设备发出启动输入和输出请求时，由操作系统调用井管理程序，由其控制着从输入井读取信息或将信息输出到输出井。

**3. SPOOLing系统的特点**

提高了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。

**守护进程**

凡是需要将独占设备改造为多个进程共享的设备时，都需要配置一个守护进程和一个假脱机目录。

**5.7缓冲管理**

缓冲的实现：采用专用的硬件缓冲区

在内存中划出一块专门的空间，作为缓冲区来存放输入输出的数据，又称“软缓冲”

1. 单缓冲
2. 双缓冲——实现双向传输，一个做为输入缓冲区用，一个做为输出缓冲区用。
3. 循环缓冲——组成：

多个缓冲区

多个指针

循环缓冲区的使用：

（1）Getbuf过程

（2）Releasebuf过程

进程同步

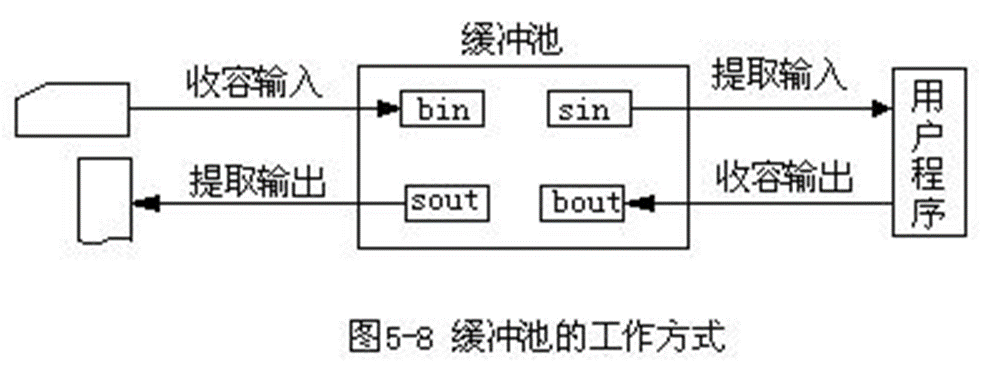
缓冲池：1．缓冲池的组成

三种类型的缓冲区，分别构成三个队列：

空缓冲队列emq。头指针F(emq)，尾指针L(emq)

输入队列inq。头指针F(inq)，尾指针L(inq)

输出队列outq。头指针F(outq)，尾指针L(outq)

**四种工作方式：收容输入、提取输入、收容输出、提取输出。见图5-8**

**5.8 磁盘存储器的管理**

* **5.8.1 磁盘概述**
* **每个盘面一个读写磁头。所有盘片上相对位置相同的磁道的组合称为柱面。**

磁盘读写方式：移臂——>旋转——读写

存取所需数据的时间由下面三部分组成：寻道时间+旋转延迟时间+传送时间

* **5.8.2 磁盘调度**

先来先服务算法

最短寻道时间算法

扫描算法：——电梯算法①开始时向磁道号增加方向访问

**服务顺序：150，160，184，90，58，55，39，38，18**

②开始时向磁道号减小方向访问

循环扫描算法——单向扫描

①设开始时向磁道号增加方向访问：

**服务顺序：150, 160, 184, 18, 38, 39, 55, 58, 90**

第六章 文件管理

6.1 文件和文件系统

**数据分成数据项、记录、文件三级。**

文件分为 有结构文件（若干条记录组成——记录式文件）+无结构文件（字符流文件——流式文件）

1. 数据项——基本数据项；组合数据项
2. 记录：是一组数据项的集合。为了能唯一的标识记录，我们选择一个或者几个数据项的集合作为**关键字**。

6.1.2文件名和类型

1.文件名和扩展名

2.文件类型

(1) 按用途分类。

系统文件、用户文件、库文件  
　 (2) 按文件中数据的形式分类。

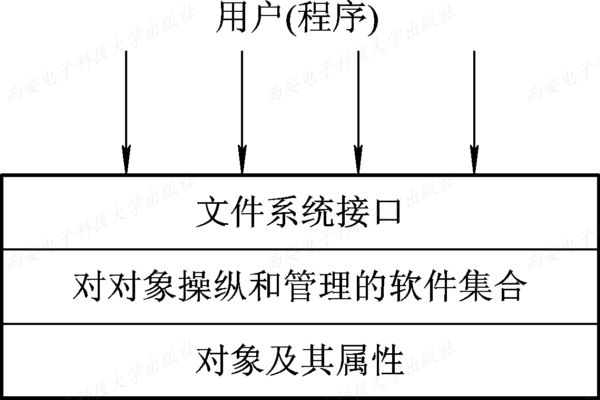
源文件、目标文件、可执行文件

(3) 按存取控制属性分类。

只执行文件、只读文件、读写文件

(4) 按组织形式和处理方式分类。

普通文件、目录文件、特殊文件

6.1.3文件系统的层次结构

**1、对象及其属性**

文件管理系统管理的对象如下：  
　　(1) 文件。  
　　(2) 目录。  
　　(3) 磁盘(磁带)存储空间。

**2.对对象操纵和管理的软件集合**

**3、文件系统的接口**

——命令接口

程序接口

6.1.4文件操作

1.最基本的文件操作

创建文件

删除文件

读文件

写文件——通过设置文件读写指针的位置，可以改顺序存取为随机读写。

设置文件的读写位置

2.文件的“打开”和“关闭”操作

——是指系统将指明文件的属性（包括该文件在外存上的物理位置）从外存拷贝到内存**打开文件表**的一个表目中，并将该表目的编号（或称索引号）返回给用户。

6.2 文件的逻辑结构

* + **这些记录构成一个文件的方法——逻辑结构**
  + **将一个文件存储到外存上的方法——存储结构**

6.2.1有结构文件

（1）定长记录

（2）变长记录——**记录中包含的数据项数目不相同；也可能是数据项长度不相同**

——顺序文件；索引文件（索引表）；索引顺序文件（为文件建立一张索引表，为每一组记录中的第一个记录设置一个表项。——至少每组内是定长记录。 ）

无结构文件

\*长度以字节为单位。

\*采用读写指针来指出下一个要访问的字符。

6.2.2顺序文件

1.顺序文件的排列方式

（1）串结构——按照存入的先后顺序排序

（2）顺序结构——按照关键字排序

2.顺序文件的优缺点：

2. 顺序文件的优缺点

批量存取时(即每次要读或写一大批记录)效率是最高的。

对于顺序存储设备(如磁带)，也只有顺序文件才能被存储并能有效地工作。

交互应用场合，需要单个文件的查找和修改时，效率偏低

想增加或删除一个记录比较困难。

6.2.3记录寻址

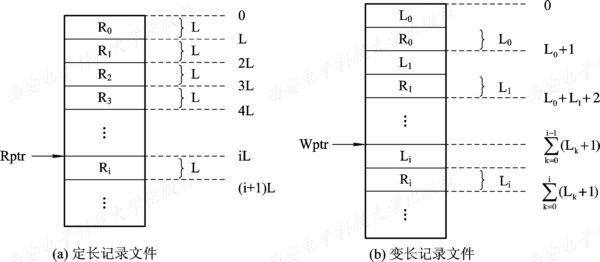
1. 隐式寻址方式

对于定长记录的顺序文件，如果已知当前记录的逻辑地址，便很容易确定下一个记录的逻辑地址。

2. 显式寻址方式

定长记录有两种方式实现随机访问：

(1) 通过文件中记录的位置

****(2) 利用关键字

6.2.4索引文件

1. 按关键字建立索引

对于变长记录较难实现直接存取。为了解决这一问题，可为变长记录文件建立一张**索引表**。

2具有多个索引表的索引文件

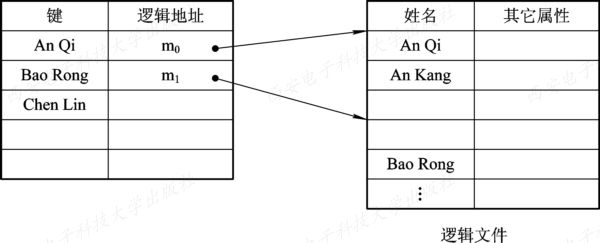
6.2.5索引顺序文件

1. 索引顺序文件的特征

* 它又增加了两个新特征：

（1）引入了文件索引表，通过该表可以实现对索引顺序文件的随机访问；

（2）增加了溢出(overflow)文件，用它来记录新增加的、删除的和修改的记录。

2.一级索引顺序文件

3.两级索引顺序文件

可以为顺序文件建立多级索引，即为索引文件再建立一张索引表，从而形成两级索引表。

6.2.6直接文件和哈希文件

对于直接文件，则可根据给定的关键字直接获得指定记录的物理地址。

6.3 文件目录

对目录管理的要求：

* + **实现“按名存取”**
  + **提高对目录的检索速度**
  + **文件共享**
  + **允许文件重名**

**6.3.1 文件控制块和索引结点**

* 一个文件对应一个**FCB（文件控制块）。文件控制块**的有序集合成为**文件目录**。即一个文件控制块就是一个文件目录项。
* 一个文件目录也看做一个文件**，成目录文件**

1. **文件控制块**

基本信息类

存取控制信息

使用信息类

1. 索引结点

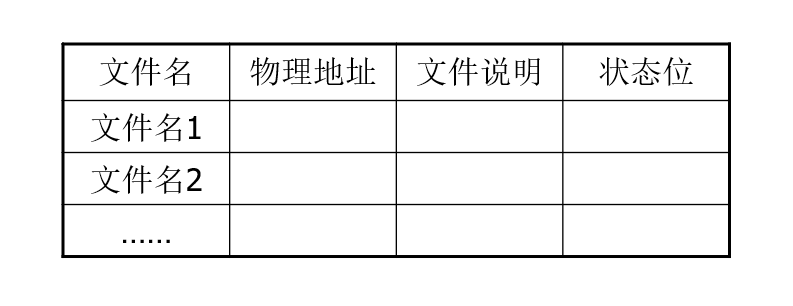
**内存索引结点**

放在内存中的索引结点。当文件被打开后，将磁盘索引结点拷贝到内存索引结点中以便使用。比磁盘索引结点又增加了以下内容

6.3.2目录结构

1.单级目录结构

整个文件系统中只建立一张目录表，每个文件一个目录项，目录项含有文件相关信息。状态位表明每个目录项是否空闲。

* + 优点
    - 简单且能实现目录管理的基本功能——按名存取。
  + 缺点
    - 查找速度慢。
    - 不允许重名。
    - 不便于实现文件共享**。**

2.两级目录

**每个用户一个文件目录；主文件目录**

* 两级目录的优点：
  + 提高了检索目录的速度。
  + 在不同的用户目录中，可以使用相同的文件名。
  + 不同用户使用不同的文件名访问系统中同一个共享文件。

3多级目录结构

当前目录

绝对路径名+相对路径名

删除目录

* **不删除非空目录**：当目录(文件)不空时， 不能将其删除，而为了删除一个非空目录，必须先删除目录中的所有文件。
* **可删除非空目录**：当要删除一目录时，如果在该目录中还包含有文件，则目录中的所有文件和子目录也同时被删除

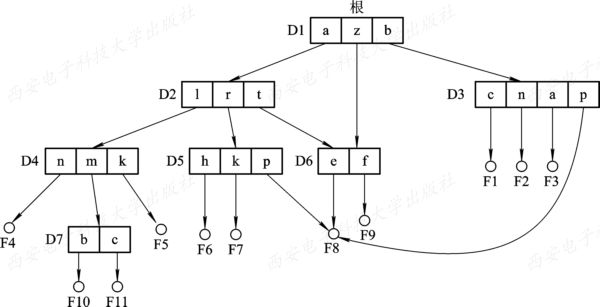
6.3.3目录查询技术

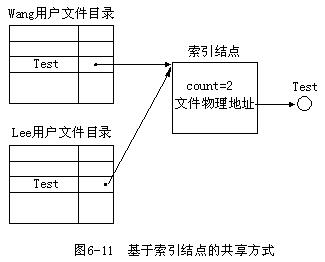
线性检索法

**Hash方法**

6.4 文件共享与保护

6.4.1 基于有向无循环图实现文件共享



6.4.2 基于索引结点的共享方式

6.4.2利用符号链接实现文件共享

为使B能共享C的一个文件F，可以由系统创建一个LINK类型的新文件，也取名为F，并将它写入B的目录中，以实现B的目录与文件F的链接。在新文件中只包含链接文件F的路径名。这种方法称为**符号链接**。

允许文件主删除文件。只不过当B要使用文件F时，会“找不到文件”，不会产生其他影响。

**缺点**：

共享文件时，可能要按路径多次读盘；

新文件也要占用磁盘空间。

6.5 文件的外存组织方式

* 目前常用的外存组织方式有**：连续组织方式、链接组织方式、索引组织方式。由此**形成三种文件物理结构**：顺序文件、链接文件和索引文件。**

6.6 文件存储空间的管理