**第一章 操作系统引论**

**一、什么是操作系统**

**1.概念：**

操作系统 operating system(OS)是配置在计算机硬件上的第一层软件，是对硬件系统的首次扩充。操作系统是一组控制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。

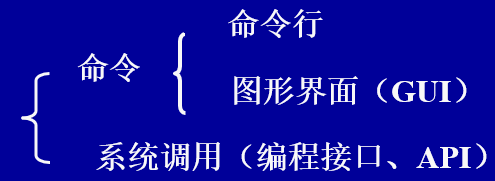
**2.操作系统地位：**

紧贴系统硬件之上，所有其他软件之下（是其他软件的共同环境，其属于**系统软件**）

**3.操作系统的作用：**

（1）OS是计算机硬件、软件资源的管理者。

（2）OS是用户使用系统的接口



（3）OS是扩展机(extended machine)/虚拟机(virtual machine)。在裸机上添加：设备管理、文件管理、存储管理（针对内存和外存）、处理机管理

**二、操作系统分类（发展历程）**

**1.** **操作系统发展的主要动力：（需求推动发展）**

**(1)提高资源的利用率和系统性能：**计算机发展的初期，计算机系统昂贵，尽力提高各种资源的利用率，有了批处理系统。

**(2)方便用户：**用户上机、调试程序，分散计算时的事务处理，有了分时操作系统。

**(3)器件的发展：**CPU的位宽度（指令和数据）、快速内外存

**2.早期手工操作**

如ENIAC计算机，此时没有程序设计语言的概念，更没有操作系统概念。

程序员提前预约一段时间,然后到机房将他的插件板插到计算机里，独占计算。

**3.单道批处理系统(simple batch processing)**

**（1）基本概念：**利用磁带把若干个作业分类编成作业执行序列，每个批作业由一个专门的监督程序（Monitor）自动依次处理。可使用汇编语言开发。

**（2）作业：**包括用户程序、数据和作业说明书（作业控制语言）

**（3）“批”：**供一次加载的磁带或磁盘，通常由若干个作业组装成，在处理中使用一组相同的系统软件（系统带）

**（4）两种批处理方式：**

**1）联机批处理：第一步，**用户提交作业（纸带或卡片）；**第二步，**操作员合成批作业（磁带）；**第三步，**批作业处理：从磁带读入用户作业和编译链接程序，编译链接用户作业，生成可执行程序；启动执行；执行结果输出。（**问题：**输入输出时，CPU处于等待状态）

**2）脱机批处理：**利用卫星机完成输入输出功能。主机与卫星机可并行工作。（**卫星机：**完成面向用户的输入输出（纸带或卡片），中间结果暂存在磁带或磁盘上）作业控制命令由监督程序(monitor)来执行，完成如装入程序、编译、运行等操作。

**（5）优缺点：**

**1）优点：**实现了同一批内各作业的自动依次更替，改善了主机CPU和I/O设备的使用效率，提高了吞吐量

**2）缺点：**内存中仅有一道程序运行，当该程序运行结束或出故障，由监控程序换入另一个程序。不能很好地利用系统资源（CPU和I/O设备忙闲不均）

**4.多道批处理系统(multiprogramming system)**

**（1）基本概念：**（集成电路），利用多道批处理提高资源的利用率和系统的吞吐量。

**（2）运行特点：**

**1）多道：**内存中同时存放几个作业；

**2）宏观：**并行运行，都处于运行状态；

**3）微观：**串行运行，各作业交替使用CPU。

**（3）优缺点：**

**1）优点：**资源利用率高：CPU和内存利用率较高；作业吞吐量大：单位时间内完成的工作总量增大；

**2）缺点：**用户交互性差：整个作业完成后或中间出错时，才与用户交互，不利于调试和修改；作业平均周转时间长：短作业的周转时间显著增长。

**5.分时系统**

**（1）基本概念：**多个用户分享使用同一台计算机。每个用户给一定的时间运行，然后切换到另一个用户，一个轮转周期在宏观上相当短，用户感觉不到。

**（2）分配方式：**时间片分配，时间片是各个程序在CPU上执行的轮换时间，其属于抢占式调度方式（**抢先式：**OS强迫作业出让CPU；**非抢先式：**程序主动出让CPU；）

**（3）特点：**

**1）人机交互性好：**在调试和运行程序时由用户自己操作。

**2）共享主机：**多个用户同时使用。

**3）用户独立性：**对每个用户而言好像独占主机。

**（4）分时系统vs批处理系统：**

**1）批处理：**交互性差, 但提高了对CPU利用率；

**2）分时处理：**用户与应用程序随时交互，控制程序运行，缩短了响应时间

**6.实时系统**

**（1）基本概念：**用于工业过程控制、军事实时控制、金融等领域，包括实时控制、实时信息处理。其要求：响应时间短，在一定范围之内；系统可靠性高

**（2）任务分类：**周期性实时任务和非周期性实时任务（有截止时间的要求。系统要考虑开始截止时间和完成截止时间）；硬实时任务和软实时任务

**（3）通用操作系统：**拥有分时、实时和批处理功能

**7.多处理机操作系统(Multi-processor Operating System)（了解即可）**

**（1）基本概念：**多处理机操作系统的出现是为了提高计算机系统性能和可靠性。提高性能有两条途径：提高各个组成部分的速度、增大处理的并行程度。

**（2）优点：**增加系统的吞吐量。但N个处理器加速比达不到N倍（额外的调度开销，算法的并行化）；提高系统可靠性：故障时系统降级运行

**（3）多处理机系统的类型：**

**1）紧密耦合(tightly-coupled)－并行（parallel）系统：**各处理机之间通过快速总线或开关阵列相连，**共享内存**，整体系统由一个统一的OS管理（**一个OS核心**）。

**2）松散耦合(loosely-coupled)－分布式系统：**各处理机带有各自的存储器、I/O设备,运行不同的操作系统，每个节点有自己的文件系统，通过通道或通信线路(如Internet)相连。

**（4）并行系统的类型：**

**1）非对称式多重处理：主处理器：**只有一个，运行OS。管理整个系统的资源，为从处理器分配任务；**从处理器：**可有多个，执行应用程序或I/O处理。**特点：**不同性质任务的负载不均，可靠性不够高，不易移植

**2）对称式多重处理：**OS交替在各个处理器上执行。任务负载平均，性能调节容易

**8.个人计算机操作系统**

**（1）基本概念：**针对单用户使用的个人计算机进行优化的操作系统。

**（2）特征：**应用领域：事务处理、个人娱乐；系统要求：使用方便、支持多种硬件和外部设备（多媒体设备、网络、远程通信）、效率不必很高，但响应性（交互性）要好。

**（3）分类：**单用户单任务（MS DOS）、单用户多任务（Windows）、多用户多任务（Unix）

**三、操作系统的特征和服务**

**1.操作系统的特征**

**（1）并发：**多个事件在**同一时间段**内发生。操作系统是一个并发系统，各进程间的并发，系统程序与应用程序间的并发。操作系统要完成这些并发过程的管理。在多道程序处理时，宏观上并发，微观上交替执行（在单处理器情况下）。

**（2）并行：**是指在**同一时刻**发生。并行的硬件环境一定是多处理机或多核系统。

（注：并发、并行一般指的是动态实体——进程）

**（3）共享：**多个进程共享有限的计算机系统资源。操作系统要对系统资源进行合理分配和调度。资源在一个时间段内交替被多个进程所用。分为：**互斥共享**（资源分配后到释放前，不能被其他进程所用）和**共享访问**

**（4）虚拟：**一个物理实体映射为若干个对应的逻辑实体——分时或分空间。虚拟是操作系统管理系统资源的重要手段，**可提高资源利用率。**

**（5）异步性：**也称不确定性，指进程的执行顺序和执行时间的不确定性；进程的运行速度不可预知；难以重现系统在某个时刻的状态；但无论快慢，应该结果相同——通过进程互斥和同步手段来保证。

**2.操作系统的服务**

**（1）服务类型：1）**程序执行和终止（包括分配和回收资源）；**2）**I/O操作；**3）**文件系统操作；**4）**通信：本机内，计算机之间（通常通信服务的使用者为进程，而不是笼统说“主机”）；**5）**配置管理：硬件、OS本身、其他软件；**6）**差错检测。

**（2）服务提供方式：**系统命令和系统调用

**四、操作系统的功能**

**1.存储管理**

提高存储空间利用率、方便用户使用、提供足够的存储空间、方便进程并发运行。包括：

**（1）存储分配与回收**

**（2）存储保护：**保证进程间互不干扰、相互保密；

**（3）地址映射（变换）：**进程逻辑地址到内存物理地址的映射；

**（4）内存扩充（覆盖、交换和虚拟存储）：**提高内存利用率、扩大进程的内存空间；

**2.处理机管理**

完成处理机资源的分配调度等功能。

**（1）进程控制**：创建、撤销、挂起、改变运行优先级等

**（2）进程同步：**协调并发进程之间的推进步骤，以协调资源共享；

**（3）进程通信：**进程之间传送数据，以协调进程间的协作；

**（4）进程调度：**作业或进程的运行切换

**3.设备管理**

方便设备使用、提高CPU与I/O设备利用率。

**（1）设备操作：**利用设备驱动程序（通常在内核中）完成对设备的操作。

**（2）设备独立性：**提供统一的I/O设备接口，使应用程序独立于物理设备，在同样的接口和操作下完成不同的内容。

**（3）设备分配与回收：**在多用户间共享I/O设备资源。

**4.文件管理**

解决软件资源的存储、共享、保密和保护。

**（1）文件存储空间管理**

**（2）目录管理：**解决文件检索问题。

**（3）文件的读写管理和存取控制**

**（4）软件管理：**软件的版本、相互依赖关系、安装和卸载等。

**5.用户接口**

提供一个友好的用户访问操作系统的接口。操作系统向上提供两种接口：

**（1）系统命令：**供用户用于组织和控制自己的作业运行。命令行、菜单式或GUI.

**（2）编程接口：**供用户程序和其它系统程序调用操作系统功能。系统调用和高级语言库函数；

**五、操作系统结构设计**

**1.传统操作系统结构**

**（1）无结构操作系统：**过程之间任意相互调用，复杂而混乱

**（2）模块化结构操作系统：**按照功能划分为若干个具有一定独立性的模块，每个模块具有某方面的管理功能，模块之间通过接口进行交互。

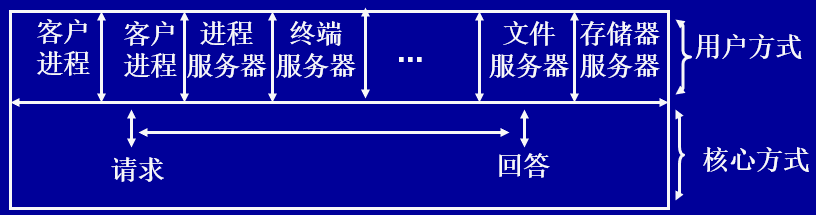
**（3）分层结构操作系统：**在目标系统An和裸机系统A0之间，铺设若干个层次的软件A1、A2、A3……。每一层仅能使用其底层所提供的功能和服务。（**优点：**易保证正确性；易扩充、易维护；**缺点：**系统效率降低。）

**2.微内核OS结构**

**（1）微内核基本概念：**

**1）足够小的内核：**内核是指精心设计的、能实现OS最基本核心功能的部分。（**OS最基本部分是指：**实现与硬件紧密相关的处理；实现一些较基本的功能；负责客户和服务器之间的通信）

**2）基于C/S模式：**将OS最基本的部分放入内核中，而把OS绝大部分放在微内核外的一组服务器（进程）中实现。两者借助微内核提供的消息传递机制来实现信息交互。



**3）应用机制与策略分离原理：**机制处于系统的基层，而策略处于系统的高层。故机制放在OS的微内核中，才有可能将内核做得很小。

**4）采用面向对象技术：**

**（2）微内核基本功能：（与硬件紧密相关的部分放入微内核）**

**1）进程（线程）管理：**进程（线程）调度、进程（线程）通信、进程的切换、多处理器之间的同步（确定用户优先级、修改优先级等放在微内核外的**进程管理服务器**中）

**2）低级存储器管理：**地址映像中的页表机制和地址变换机制（虚存管理、页面置换、内存分配与回收放在微内核外的**存储器管理服务器**中）

**3）中断和陷入管理：**捕获所发生的中断和陷入事件，进行前期处理，中断现场保护，识别中断类型，然后将有关事件的信息转换成消息后，发送给相关的服务器

**（3）微内核优点：**提高了系统的可扩展性；增强了系统的可靠性和可移植性；提供了对分布式系统的支持等等。

**（4）微内核缺点：**运行效率低下，完成一次系统调用至少需要进行四次上下文切换。

**六、硬件保护**

**1.双模式操作**

**用户模式：**（用户态、目态）执行普通的应用程序；

**监控模式：**（内核模式、系统模式、监控态、管态、特权模式）执行操作系统核心代码

为了区分在PSW中加入模式位：系统态（0），用户态（1）



用户需要操作系统服务时，必须从目态变为管态；将能引起损害的机器指令作为**特权指令**，而特权指令只能在系统态下运行；当用户试图调用特权指令时，发生自陷。（**注：**由管态转换到目态是特权指令，其他的特权指令有I/O控制、定时器管理和中断管理。）

**2.I/0保护**

所有I/O指令都是特权指令

**3.存储保护**

某进程不可以访问其它进程的地址空间（非共享情况）；用户进程必须通过系统调用访问OS核心区

在连续分配方式中，为了实现存储保护，在硬件上增加两个寄存器：**基地址寄存器：**存放本进程最低的内存物理地址；**界限寄存器 ：**存放本进程的存储区域大小

**硬件保护原则：**核心态下，OS可以无限制访问用户和核心进程的存储空间。修改上述两个寄存器的指令属于特权指令。

**4.CPU保护**

**设置时钟：**定时中断，该中断使操作系统获得CPU控制权，防止程序死循环一直占用CPU。定时通过时钟脉冲进行计数，减到0便发生中断。

定时时钟是分时操作系统的硬件基础。（设置时间片，时间片到必须让出CPU并等待下个时间片）

修改定时时钟的指令为特权指令。

**第二章 进程的描述和控制**

**一、进程的基本概念**

**1.程序的执行特质：**

**（1）顺序执行：**单道批处理系统的执行方式。特征为：**顺序性**、**封闭性**（独占全部资源，结果仅由控制逻辑决定，不受外界因素影响）、**可再现性**（初始条件相同结果相同）

**（2）并发执行：**为了提高资源利用率。特征为：**间断性**（走走停停）、**失去封闭性**（共享资源，受其他程序的控制逻辑的影响）、**失去可再现性**（外界环境在程序的两次执行期间发生变化，失去原有的可重复特征）

**2.进程的定义与特质：**

**（1）进程的定义：**一个具有一定独立功能的程序在一个数据集合上的一次动态执行过程。未引入线程的系统中，进程是处理机、存储器和外设等资源的分配和回收的单位。

**（2）进程的特征：**

**1）动态性：**创建产生，调度执行，受制于资源，撤销消亡。（具有动态地址空间）

**2）并发性：**多个进程同时存在于内存，宏观上同时运行。

**3）独立性：**各进程的地址空间相互独立，除非采用进程间通信手段。

**4）异步性：**进程按各自独立的、不可预知的速度向前推进。

**5）结构化**：**进程实体由代码段，数据段、堆栈和进程控制块(PCB) 组成。**

（说明：**数据段**存放全局变量；**堆**存放动态分配空间；**栈**存放临时数据；**核心栈** 在系统调用时用来保存中断/异常现场，保存函数调用的参数和返回值）

**（3）进程与程序的区别：**

1）进程是动态的，程序是静态的；程序是有序代码的集合；进程是程序的执行。

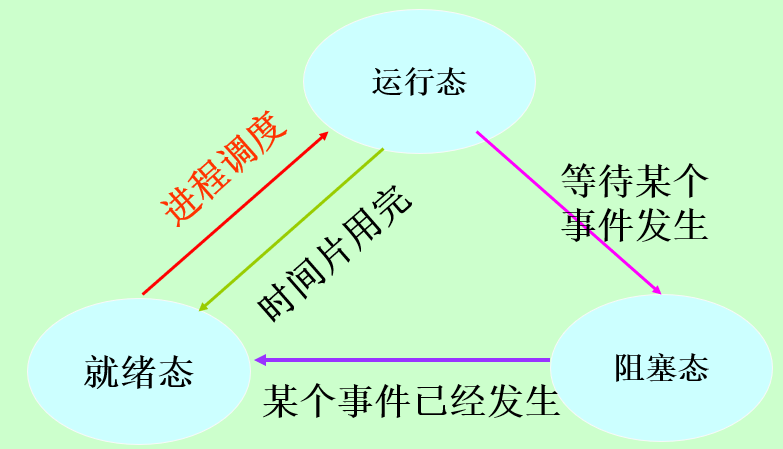
2）进程是暂时的，程序是永久的。

3）进程与程序的组成不同：进程的组成包括程序、数据、进程控制块和堆栈。

4）进程与程序的对应关系：通过创建进程系统调用，一个程序可对应多个进程；通过调用关系，一个进程可包括多个程序。

**3.进程的基本状态**

**（1）基本三态变化：**



**就绪状态：**进程已获得除处理机外的所需资源，分配到CPU就可执行

**运行状态：**占用处理机资源运行；处于此状态的进程的数目小于等于CPU的数目

**阻塞态：**由于进程等待某种条件，在条件满足之前无法继续执行

**（2）引起状态转化的事件：**

**1）创建新进程：**创建一个新进程，运行一个程序。创建进程后，它处于就绪状态。

**2）调度运行：**从就绪进程表中选择一个进程，进入运行状态；

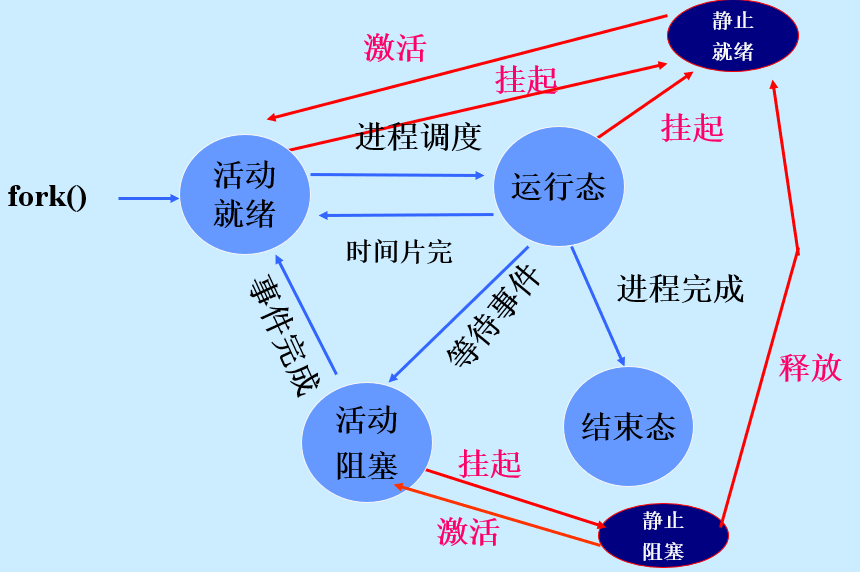
**3）释放(Release)：**由于进程完成或失败而终止进程运行，进入结束状态；运行到结束：分为正常退出Exit和异常退出Abort（执行超时或内存不够，非法指令或非法地址，I/O失败，被其他进程所终止）

**4）超时（Timeout）：**由于用完时间片，或高优先级进程就绪等导致进程暂停运行；进程由运行态→就绪态

**5）事件等待（Event Wait）：**进程要求的事件未出现；运行态→阻塞态；

**6）事件出现（Event Occurs）：**进程等待的事件出现；阻塞态→就绪态

**（3）挂起的引入**



**挂起的原因：**

**1）终端用户请求：**终端用户发现自己运行的程序有问题，希望暂停程序的执行；

**2）父进程请求：**父进程希望挂起自己的某个子进程，以便考察和修改该子进程

**3）负荷调节的需要：**内存不足，挂起低优先级的进程

**4）操作系统的需要：**挂起某些进程以便检查运行中的资源使用情况或进行记账

**新状态的引入：**

**1）活动就绪(Ready)：**进程在内存且可立即进入运行状态；

**2）就绪挂起（静止就绪）：**进程在外存，但只要进入内存，即可运行；

**3）阻塞状态(活动阻塞)：**进程在内存并等待某事件的出现；

**4）阻塞挂起状态（静止阻塞）：**进程在外存并等待某事件的出现；

**4.进程控制块**

**（1）作用：**PCB是每个进程在OS中的登记表项，OS据此对进程进行控制和管理；PCB记录用于描述进程执行情况及控制进程运行的全部信息；PCB常驻于由OS维护的内存核心区，需要系统调用才可访问

**（2）PCB中包含的信息：**

**1）进程描述信息：**进程标识符、进程名、用户标识符

**2）处理机状态信息：**寄存器的值（通用、PC、PSR、堆栈指针）以及中断保存现场

**3）进程调度信息：**进程的当前状态、优先级、运行统计信息（执行时间、已等待CPU的时间）、事件（阻塞原因）

**4）进程控制信息：**程序段数据段地址、进程间同步和通信信息、资源占用信息、链接指针（下一PCB首地址）

**（3）PCB组织方式：**将处于同一状态的PCB组织在一起，有链表和索引表两种方式

**二、进程控制**

**1.进程的创建**

**（1）子进程与父进程：**父进程创建子进程，子进程还可创建子进程。子进程可以继承父进程所拥有的资源。当子进程被撤销应将继承的资源归还给父进程。撤销父进程也必须同时撤销所有的子进程。

**（2）引起进程创建的事件：**用户登陆、作业调度、提供服务、应用请求

**（3）进程创建原语要完成的工作：1）**分配空白PCB；**2）**为新进程分配资源；**3）**初始化PCB;**4）**新进程插入就绪队列

**2.进程的终止**

**（1）引起终止的事件：**正常结束、异常结束（程序运行错误、超时或I/0故障）、外界干预（操作员或系统干预、父进程请求、父进程终止）

**（2）进程终止过程：**

1）终止所有子进程.

2）回收资源：释放内外存空间、关闭所有打开文件、释放当前目录、释放共享内存段和各种锁定lock。

3）若本身是运行的，通知调度程序进行CPU的调度.

4）将PCB从所在队列移出，等待其他进程来收集信息。

**3.进程的阻塞和唤醒**

**（1）引起阻塞的事件：**向系统请求共享资源失败、启动某种操作.如启动了I/O、新数据未到达、等待新任务的到达

**（2）进程阻塞过程：**自身调用有关阻塞原语，进入等待队列。进程的**主动性行为**。

进程由运行态变为阻塞态。**引起处理机调度**。

**（3）引起唤醒的事件：**等待的事件到达

**（4）进程唤醒过程：**由阻塞队列转入就绪队列。进程由阻塞态变为就绪态.**可能引起处理机调度**。

**4.进程的挂起和激活**

**（1）挂起的原因：**父进程请求挂起子进程，用户进程请求挂起自己。负荷调节的需要 **（2）进程挂起过程：**从内存调到外存，**若被挂起进程在执行，则引起处理机调度**

**（3）唤醒的原因：**父进程或用户进程请求，或内存已有足够空间

**（4）进程唤醒过程：**从外存调入内存，改变进程的状态。**可能引起处理机调度**。

**三、进程同步**

**1.背景**

**（1）消费者与生产者问题**

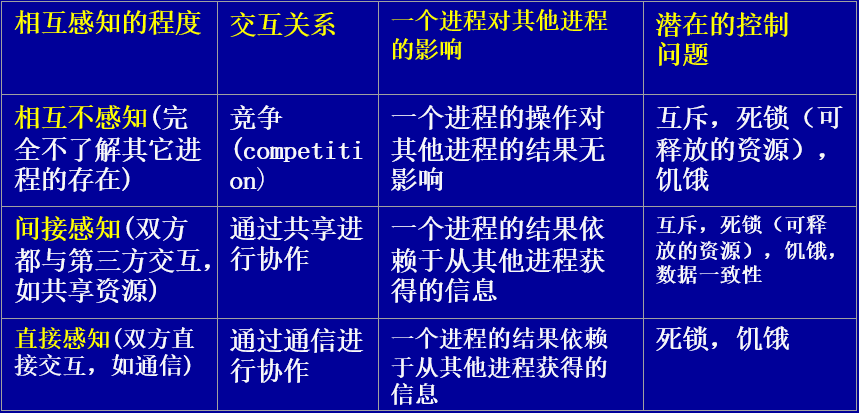
**（2）原语：**由若干条指令构成的“**原子操作**(atomic operation)”过程，作为一个整体而不可分割－－要么全都完成，要么全都不做。许多系统调用是原语。但并不是所有的系统调用都是原语。原语是要在核心态下执行的。因为执行原语时要关中断。

**2.进程同步和互斥**

**进程互斥：**由于共享资源要求的排他性，进程间要相互竞争，以获得这些资源的使用权。

**进程同步**：多个进程中发生的事件存在某种时序关系，必须协同工作、相互配合，以共同完成一项任务

**（1）进程间的制约关系**



**（2）临界资源：**硬件或软件（如外设、共享代码段、共享数据结构），多个进程在对其进行访问时（关键是进行写入或修改），必须互斥地进行。有些共享资源可同时访问，如只读数据，其不是临界资源。

**（3）临界区：**在每个进程中，访问临界资源的一段代码。

**进入区(entry section)：**在进入临界区之前，检查可否进入临界区的一段代码。如果可以进入临界区，通常设置相应“正在访问临界区”标志

**退出区(exit section)：**位于临界区后面，用于将“正在访问临界区”标志清除。

**剩余区(remainder section)：**代码中的其余部分。

**（4）同步机制应遵循的准则：**

**空闲则入：**其他进程均不处于临界区，应允许请求进入临界区的进程进入；

**忙则等待（互斥进入）：**已有进程处于其临界区，请求进入临界区的进程应等待；

**有限等待：**等待进入临界区的进程不能“死等”；

**让权等待：**不能进入临界区的进程，应释放CPU（如转换到阻塞状态）

**（5）三个算法、利用硬件同步指令解决临界区问题：**见PPT

**3.信号量机制**

**（1）整型信号量机制：**把信号量定义为整型量s和两个原子操作（除初始化操作）：P和V, 现又称为：wait和signal。（问题：不能实现“让权等待”）

**（2）记录型信号量机制：**

typedef struct {

int count;  
 struct process \*queue;  
 } semaphore;

每个信号量s 除一个整数值s.count外，还有一个进程等待队列s.queue，是阻塞在该信号量的各个进程的PCB链成的队列。信号量只能通过P、V操作进行访问。

**具体操作：重点！见PPT**

**（3）互斥锁机制：**

**互斥锁定义、互斥锁的acquire、release操作：**见PPT

互斥锁不同于记录型信号量，互斥锁没有实现让权等待。进程若获取锁不成功，则会不断“自旋”。因此又称为**自旋锁**(spin locks)

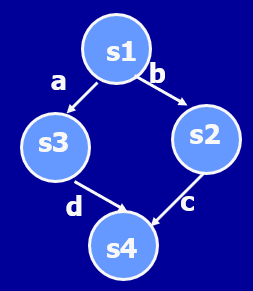
**自旋锁的优势:**进程等待锁的时候，不需要进行上下文切换，上下文切换要花费很多处理机时间。所以若锁只是被协作进程持有较短的时间，可以考虑使用自旋锁实现临界区的互斥访问。尤其是在多处理机环境（并行系统）中。（硬件互斥指令实际上实现的就是自旋锁。）

**4.信号量的应用**

**（1）利用信号量实现互斥：**为临界资源设置一个互斥信号量mutex，其初值等于临界资源的初始值；在每个进程中将临界区代码置于wait(mutex)和signal(mutex)原语之间



**（2）利用信号量实现同步：**前趋关系：并发执行的进程P1和P2中，分别有代码C1和C2，要求C1在C2开始前完成；通过设置signal-wait实现同步。具体如下例：

**四、经典的进程同步问题（略，见PPT！）**

**1.生成者消费者问题**

**2.读者——写者问题**

**3.哲学家进餐问题**

**五、管程机制**

**1.定义**

管程是由**高级语言**实现的。是由过程、变量及数据结构等组成的一个集合，它们组成一个特殊的模块或软件包。管程是语言概念，C语言不支持它。类Pascal， Java，C#等支持，它被作为一个程序库实现。使用信号量的效率比管程高。

**管程由四部分组成：**（1）管程的名称（2）局部于管程的共享变量说明（3）对该数据结构进行操作的一组过程（4）对管程中数据设置初值的语句

任何管程外的过程都不能访问管程内的数据结构。管程相当于围墙，将共享变量和对它进行操作的若干过程围了起来，进程只要访问临界资源就必须通过管程。在管程中的进程，每次只能有一个是活动的，从而实现了互斥。

管程是**编程语言的组成部分**，编译器可以采用与其他过程调用不同的方法来处理对管程的调用。进入管程的互斥由编译器负责，通常是用一个互斥量或二元信号量。写管程的人无需关心编译器如何实现互斥，只需要**把所有的临界区转换为管程即可**。

**2.条件变量**

由于管程通常是用于管理资源的，因而在管程内部，应当存在某种等待机制。当进入管程的进程因资源被占用等原因不能继续运行时，要使其等待。

为了区别不同的等待原因，设置了条件变量和在条件变量上进行操作的两个同步原语wait, signal。

条件变量说明形式为：condition：x，y;同步原语wait使调用进程等待，并将它排在相应的等待队列上；signal唤醒等待队列的队首进程。使用方式为：x.wait，x.signal。

**六、进程通信（了解即可）**

**进程通信：**是指进程之间的信息交换，所交换的信息量，少者是一个状态，多者是成千上万个字节。进程间的通信可分为低级通信和高级通信两种

**低级通信：**只能传递状态和整数值（控制信息），包括进程互斥和同步所采用的信号量和管程机制。**低级通信的缺点为：**传送信息量小，效率低，每次传递信息量固定，大量信息需要多次通信；编程复杂，用户需要实现具体通信细节，容易出错。

**高级通信：**主要指利用较复杂的一种特定数据结构进行较大信息量的通信，通信效率高。包括四类：**共享存储器系统、消息传递系统、管道通信系统和客户服务器系统。**

**1.进程通信的类型**

**（1）共享存储器系统：**相互通信的进程共享某些数据结构或存储区，进程之间通过这些空间进行通信。分两类：

**1）共享数据结构：**通信的进程共享某些数据结构，如缓冲区。（数据结构需程序员自己设定，通信效率低，低级通信）

**2）共享存储器：**为了传递大量数据，在存储器中划出一块共享存储区。（申请分区并指定关键字 -> 获得分区描述符并连接 -> 可进行读写操作）

**（2）消息传递系统：**程序员直接利用系统提供的一组通信原语进行通信。OS隐藏了通信的细节，简化了通信程序的编制，得到广泛应用。

**（3）管道通信系统：**管道是指用于连接一个读进程和一个写进程以实现他们之间通信的一个**共享文件**，又名pipe文件。

**1）普通管道：**普通管道只允许单向通信（一端读，一端写）。要想实现双向通信，必须建立两个管道，每个支持一个方向的通信。普通管道在Windows系统中称为**匿名管道**。它是单向通信，通信进程间是**父子关系**。因此只支持**本机上进程间的通信**。通信一旦终止，管道便不再存在。

**2）命名管道：**通信是双向的，不要求父子关系。一旦命名管道建立起来，几个进程都可通过它通信。命名管道可有几个写者。而且通信进程结束后，命名管道仍然存在。

**UNIX命名管道：**双向通信，半双工，字节流通信，支持本地通信不支持网络通信。

**Windows命名管道：**双向通信，全双工，字节流通信和消息通信，本地网络通信均支持。

**2.消息传递通信的实现方法**

**（1）直接通信方式：**发送进程利用OS提供的**发送命令**，直接把消息发送给目标进程。

发送进程和接收进程都以显示方式提供对方的标识符。

**（2）间接通信方式：**借助于收发双方进程之外的**共享数据结构**作为通信中转，如**信箱**。 系统为信箱通信提供了若干条原语，如信箱的创建、撤销和消息的发送、接收等。信箱的创建者是信箱的拥有者。信箱分为三类：

**私用信箱：**用户进程自己创建，其他用户只能发送信息

**公用信箱：**OS创建，核准进程即可以向信箱发消息也可接收消息。

**共享信箱：**进程创建的可共享信箱，拥有者指定可共享的进程。拥有者、共享者都可以从信箱中取走自己的消息

**七、线程**

**1.线程的基本概念：**

**（1）引入线程的目的：**用它来提高系统内程序的并发程度，提高系统效率，增大系统作业的吞吐量。

**（2）线程的属性：**在引入线程的OS中，线程是进程的实体。线程是处理机调度的实体，是最接近硬件、最低级的调度。线程有如下属性：

**1）共享进程资源：**属于同一个进程的所有线程具有相同的地址空间，可访问进程拥有的打开文件、定时器、信号量、全局数据等。

**2）轻型实体：**每个线程只有少量资源，包括：线程标识符、线程控制块TCB（保存PC和一组寄存器）、核心栈（核心态工作时保存参数和返回地址）、用户栈（局部变量）

**3）线程是处理机调度的单位**

**4）线程可并发执行**

**（3）线程的优点：**

1）线程的创建时间比进程短；

2）线程的终止时间比进程短；

3）同一进程内的线程切换时间比进程短；

4）同一进程内线程之间共享内存和文件资源，可直接通信不需要通过OS的内核。

**（4）线程与进程的比较:**

1）在支持多线程的系统中，**进程是资源分配的单位**（如存储器、打开文件等，不含处理机）。**线程是处理机调度单位**，但不是资源的分配单位。

2）线程拥有必不可少的资源，如：线程控制块（TCB）、PC、寄存器上下文和堆栈

3）线程同样具有就绪、阻塞和执行三种基本状态

4）线程减少了并发执行的时间和空间开销（线程的创建、退出和调度），因此容许在系统中建立更多的线程来提高并发度。

5）进程间的地址空间和其他资源（如打开文件）相互独立。但同一进程的各线程间共享进程的所有资源。

6）进程间通信复杂。线程间可以直接读写进程数据段（如全局变量）来进行通信。

**2.用户级线程和内核支持线程**

**（1）用户级线程：**仅存在于用户空间中。对于线程的创建、撤销和切换，无需内核的支持，内核不了解用户线程的存在。一个线程发起系统调用而阻塞，则整个进程都要等待。时间片分配给进程，进程内有多线程时，则每个线程执行时间相对就少。

**（2）内核线程：**由OS支持，无论是用户进程中的线程，还是系统进程中的线程，其创建、撤销和切换等，都是依靠内核实现的。线程的创建和管理慢于用户级线程。一个线程执行了阻塞系统调用，整个进程不会阻塞。能在多处理机上并行执行。

**3.多线程模型**

**（1）多对一：**多个用户级线程对应一个内核线程。线程管理在用户空间进行，效率高。但线程阻塞，其整个进程都将阻塞。

**（2）一对一：一**个用户级线程对应一个内核线程。开销大，每创建一个用户线程就要创建一个内核线程，限制了线程并发数量。但线程阻塞，其进程的其他线程任能正常运行。

**（3）多对多：**将多个用户线程多路复用到同样数量或更小数量的内核线程上。内核线程的数量与特定应用程序或特定机器有关。采用这种方法，内核只识别内核级线程，并对其进行调度。其中一些内核线程会被多个用户级线程多路复用。

**第三章 调度与死锁**

**一、处理机调度的基本概念**

**1.高级调度（调度对象是作业）**

**（1）基本概念：**又称作业调度、长期调度（long-term scheduling)、宏观调度。实时系统、分时系统无作业调度。批处理系统有。

作业进入系统后，被依次存放在**外存**的后备队列上。系统会为每个作业建立一个作业控制块（JCB）。作业调度就是按照一定的算法，从外存后备队列中选取某些作业调入内存，并为它们创建进程，分配必要的资源，然后将创建进程的PCB插入就绪队列。

**（2）作业和作业步：**

**1）作业：**（用户）利用计算机进行一次运行所需工作的集合。要完成一个工作，用户必须先提交一个作业。一个作业可能由多个程序构成。（普通PC机没有作业的概念）

**2）作业步：**编译作业步、链接作业步和运行作业步

**（3）作业控制块：**作业标识符、用户名、用户账号、作业类型（CPU繁忙、IO繁忙、批处理型、终端型）、作业状态、调度信息（优先级、运行时间）、资源需求等

**（4）作业运行三个阶段和三个状态：**（a）收容阶段----后备状态；（b）运行阶段----运行态；（c）完成阶段----完成状态

**（5）作业调度的主要任务：**接纳多少作业（多道程序度）；接纳哪些作业（调度算法）

**2.低级调度（调度对象是进程）**

**（1）基本概念：**又称进程调度、短期调度、微观调度、CPU调度。用于决定就绪队列中哪个进程获得处理机，之后派发程序将处理机分配给该进程

**（2）进程调度两种方式：**

**非抢占式调度：**1）执行的程序正常结束或因错误而终止；2）运行中进程提出I/O请求，在等待I/O完成前被阻塞，进程调度；3）进程通讯中，执行中的进程执行了某种原语操作，如P操作、阻塞原语，都可能引起进程调度

**抢占式调度：**1）分时系统中，时间片用完；2）按照优先级调度，有更高优先级进程变为就绪时

**（3）进程调度的任务：**1）保存处理机的现场；2）按某种算法选取进程；3）把处理机分配给进程

**（4）进程调度机制：**1）排队器；2）派发器；3）上下文切换

**（5）上下文切换需要以下两个步骤：**

1）保存当前进程的上下文，装入派发程序的上下文（先运行派发程序）

2）移出派发程序的上下文，装入新进程的上下文（派发程序决定新装入的进程）

上下文切换需要大量的load、store指令，花费执行上千条指令的时间。有的CPU采用两组寄存器。一组在管态下用，一组在目态下用。

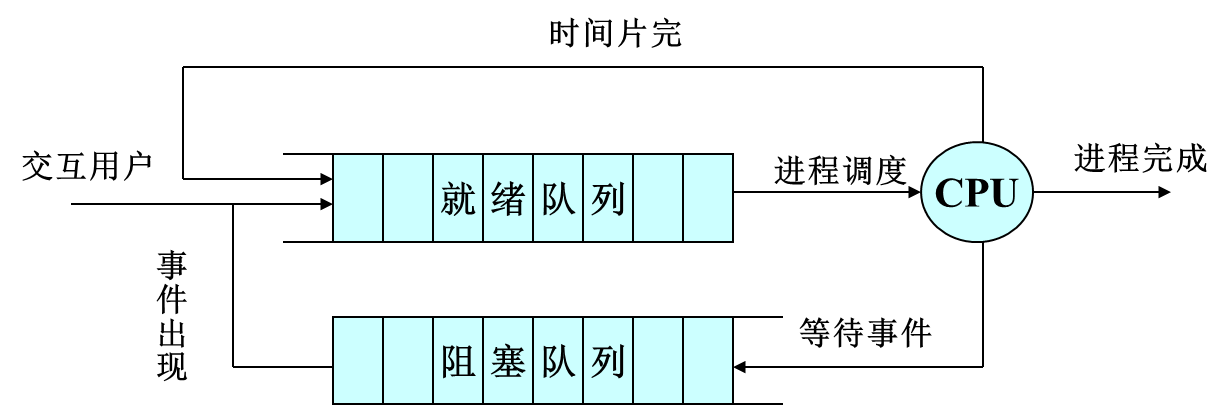
**3.中级调度：（负责进程的挂起和激活）**

**（1）基本概念：**有时需要选择内存中阻塞或就绪的进程暂时放到外存（一般是硬盘），即所谓的挂起。当这些进程又具备了运行条件、且内存又稍有空闲时，中级调度把外存上的就绪进程调入内存，放入就绪队列。这种内外存的数据交换称为对换。

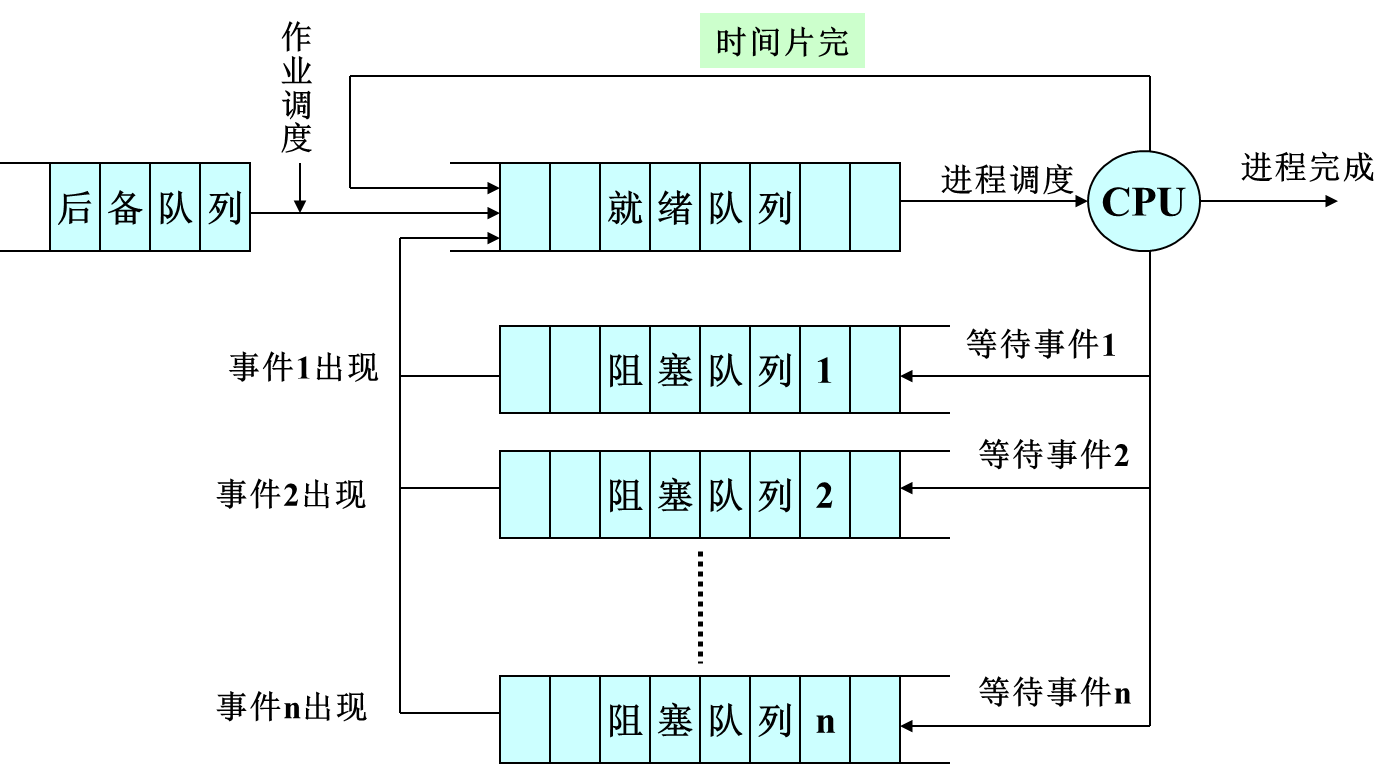
**（2）需要解决的问题：**在阻塞或就绪的进程中选择哪个（些）进程挂起；在条件允许下，在外存挂起的进程集合中选择哪个进程激活并调回内存。

**4.三种调度的关系**

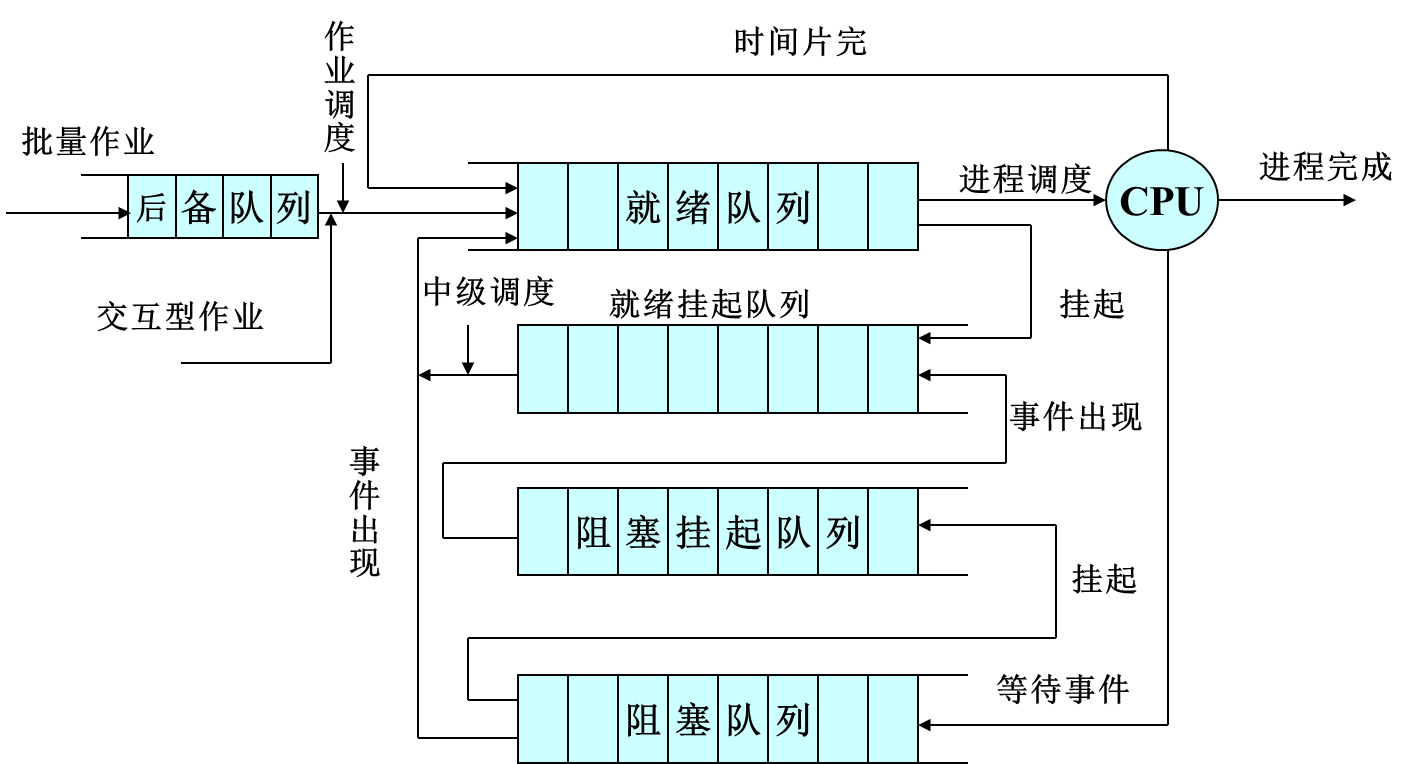
**（1）仅有进程调度的调度队列模型**

****

**（2）具有高级和低级的调度队列模型**

****

**（3）同时具有三级调度的调度队列模型**

****

**5.选择调度方式和算法的若干准则**

**（1）处理机调度算法的共同目标：**

**1）资源利用率**

**2）公平性：**诸进程都获得合理的CPU时间，**不发生饥饿。**

**3）平衡性：**以CPU为主的作业和以I/O为主的作业合理搭配，使资源均衡使用。

**4）策略强制执行：**如安全策略，即使会使某些工作延迟也要强制执行。

**（2）批处理系统的目标：**

**1）周转时间短：**作业**从提交到完成（得到结果）**所经历的时间为周转时间。

**平均周转时间：**（其中Ti表示第i个作业的周转时间)

**平均带权周转时间：**（其中Tsi表示第i个作业的服务时间）

**2）系统吞吐量高：**吞吐量指单位时间内所完成的作业数

**3）处理机利用率高：**大中型主机多用户系统性能指标，系统价格昂贵

**（3）分时系统的目标：**

**1）响应时间快：**分时系统重要指标。输入一个请求到系统给出首次响应的时间。

**2）均衡性：**系统响应时间的快慢应与用户请求的复杂性相适应。

**（4）实时系统的目标：**

**1）截止时间的保证：**开始截止时间和完成截止时间

**2）可预测性：**如多媒体系统中电影连续播放，是可预测的，采用双缓冲区，播放i帧，同时处理i+1帧

**二、调度算法（重点！）**

**1.先来先服务和短作业优先算法**

**（1）FCFS先来先服务算法：**

**1）算法描述：**按照作业提交或进程变为就绪状态的先后次序，分派CPU。当前作业或进程占用CPU，直到执行完或阻塞，才出让CPU（**非抢占方式**）。

**2）算法特点：**比较有**利于长作业**，而不利于短作业；有**利于CPU繁忙的作业**，而不利于I/O繁忙的作业；可用于作业调度和CPU调度

**（2）SJF短作业优先算法：**

**1）算法描述：**对预计执行时间短的作业（进程）优先分派处理机。（**非抢占方式**）

**2）优点：**改善平均周转时间和平均带权周转时间，缩短作业的等待时间；提高系统的吞吐量；可用于作业调度和CPU调度

**3）缺点：**对长作业非常不利，可能长时间得不到执行；未能依据作业的紧迫程度来划分执行的优先级；难以准确估计作业（进程）的执行时间，从而影响调度性能。

**（3）SJF的变型：**

**1）SRT最短剩余时间优先算法：**允许比当前进程剩余时间更短的进程来**抢占**

**2）HRRN最高响应比优先算法：**响应比R=(等待时间+要求执行时间)/ 要求执行时间（**非抢占方式**）

**2.优先权调度算法**

**（1）优先权调度算法类型：**

**1）非抢占式方式：**除非自愿或时间片到，当前的进程不可以被优先级更高的进程抢用CPU。

**2）抢占式方式：**当前的进程在其时间片未用完时就可被优先级更高的进程抢用CPU

具体分为以下几种算法：

**1）完全不可抢占或用户态不可抢占：**用户态核心态都不可被抢占

**2）内核完全不可抢占：**用户态可抢占，核心态不可抢占

**3）内核部分可抢占：**用户态可抢占，核心态只在某些时刻（可抢占点）可抢占

**4）完全可抢占或内核完全可抢占：**用户态核心态都可被抢占（注：并不是100%可抢占，只是将内核中不可抢先的代码段尽量减少而已）

**（2）优先权的类型：**

**1）静态优先级：**创建进程时就确定，直到进程终止前都不改变。通常是一个整数

**2）动态优先级：**在创建进程时赋予的优先级，在进程运行过程中可以自动改变，以便获得更好的调度性能

**（3）高响应比优先调度算法：**响应比R=(等待时间+要求执行时间)/ 要求执行时间。该算法时FCFS和SJF的折中，可用于进程和作业调度（**非抢占式方式**）

**3.基于时间片的轮转调度算法**

**（1）时间片轮转调度算法：（用于进程调度）**

**1）算法描述：**将系统中所有的就绪进程**按照FCFS原则**，排成一个队列。每次调度时将CPU分派给队首进程，让其执行一个时间片。在一个时间片结束时，发生时钟中断。调度程序据此暂停当前进程的执行，将其送到就绪队列的末尾，并通过上下文切换执行当前的队首进程。进程也可在未用完时间片，就让出CPU（阻塞），此时引起进程调度。

**2）时间片长度确定：**

**（a）长度需合理：**过长 -> 退化为FCFS；过短 -> 用户的一次请求需要多个时间片才能处理完，上下文切换次数增加，响应时间长

**（b）就绪进程的数目：**数目越多，时间片越小

**（c）系统的处理能力：**应当使用户输入通常在一个时间片内能处理完

**（2）多级反馈队列调度算法：**

**1）算法描述：**设置多个就绪队列，分别赋予不同的优先级，队列1的优先级最高。每个队列执行时间片的长度也不同，规定优先级越低则时间片越长。新进程进入内存后，一般先进入最高级队列的末尾，若在该级一个时间片内未完成，则降级，放到低一级队列的末尾，以此类推直到最后一级，最后一级一般是FCFS队列。

**2）调度方式：**仅当较高优先级的队列为空，才调度较低优先级的队列中的进程执行。如果进程执行时有新进程进入较高优先级的队列，则抢先执行新进程，并把被抢先的进程投入原队列的末尾

**3）算法性能：**

**（a）终端型进程：**让其进入最高优先级队列，以及时响应I/O交互。通常执行一个小时间片，可处理完一次I/O请求的数据，然后转入到阻塞队列。

**（b）计算型进程：**一般是**长批处理作业**，每次都执行完时间片，进入更低级队列。最终采用最大时间片来执行，减少调度次数

**（c）短批处理作业：**先放入第1级，一般经过1，2级即可完成

**4）优点：**提高系统吞吐量和缩短平均周转时间而照顾短进程；为获得较好的I/O设备利用率和缩短响应时间而照顾I/O型进程；不必估计进程的执行时间，动态调节

**三、实时调度（非考点，略）**

**四、产生死锁的原因和必要条件**

**1.产生死锁的原因**

**（1）死锁的概念：**指多个进程因竞争共享资源而造成的一种僵局，若无外力作用，这些进程永远不能向前推进。

**（2）产生死锁原因：**

**1）竞争不可剥夺资源：**双方都拥有部分资源，同时在请求对方已占有的资源

**2）竞争临时性（可消耗性）资源：**可以动态生成和消耗，一般不限制数量

**3）进程推进顺序不当：**多个进程并发执行，相互的推进顺序不确定

**2.产生死锁四个必要条件：（重点！）**

**（1）互斥：**任一时刻只允许一个进程使用资源

**（2）请求和保持：**进程保持了至少一个资源，但又提出了新的资源请求，该资源又被其他进程占用。

**（3）不可剥夺：**进程已经占用的资源，未使用完，不能被剥夺。

**（4）环路等待：**存在进程－资源环形链，即有进程集合{P0, P1, P2,….Pn},P0等待P1占用的资源，P1等待P2占用的资源…..Pn等待P0占用的资源。

**3.处理死锁的方法：**

**（1）预防死锁：**加限制条件，破坏产生死锁的四个必要条件中的一个或几个。

**（2）避免死锁：**在资源的动态分配过程中，防止系统进入不安全状态。

**（3）检测死锁：**允许系统进入死锁，但系统及时检测，并采取措施。

**（4）解除死锁：**当检测到系统进入了死锁，采取措施解除。

（注：上述四个方法，从上到下防范程度减弱、资源利用率提高、进程并发程度提高）

**五、预防死锁**

**1.摒弃“请求保持”条件**

**（1）实现方式：**预先静态分配法：（针对死锁的第2个条件）预先分配进程运行所需的全部资源，保证不等待资源

**（2）优点：**简单、易于实现、安全

**（3）缺点：**降低了资源的利用率，降低进程并发程度；有可能无法预先知道所需资源

**2.摒弃“不可剥夺”条件**

**（1）实现方式：**当一个已经保持了某些资源的进程，再提出新的资源请求而不能立即得到满足时，必须释放它已持有的资源

**（2）缺点：**需付出代价。反复申请释放资源，降低系统吞吐率

**3.摒弃“环路等待”条件**

**（1）实现方式：**有序资源使用法：把资源分类按顺序排列，所有进程对资源的请求必须按照资源序号递增次序提出。保证不形成环路

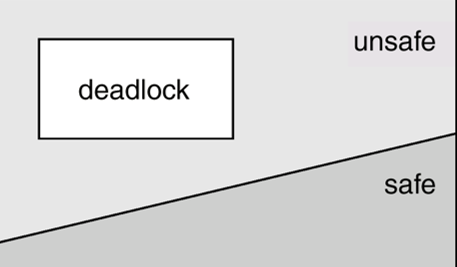
**（2）缺点：**资源序号固定，限制新设备的增加；降低资源利用率；限制了用户简单、自主地编程

**六、避免死锁**

**1.系统的安全状态**

**（1）安全状态：**是指系统能按某种进程顺序（P1, P2, ….Pn），为进程Pi分配其所需资源，直至进程的最大需求，使每个进程都可顺利完成。 则（P1, P2, ….Pn）称为安全序列。系统处于安全状态。若系统无法找到安全序列，则称系统处于不安全状态。

**（2）安全状态、不安全状态、死锁之间的关系：**

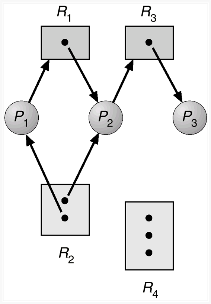


**2.银行家算法（重点！略，见PPT）**

**七、死锁的检测与解除**

**1.死锁的检测**

**（1）资源分配图：**从资源R到进程P的边表示资源R已分配1个给进程P。从进程P到资源R的边表示P正因请求R而处于等待状态。如下图所示：（R表示资源，P表示进程）



**（2）死锁定理：**

**1）资源分配图的化简方法：**删除即不处于等待状态又不独立的进程的所有弧（包括请求边和分配边），该点变为孤立点。重复上述过程，若最后所有进程结点是孤立点，则称该资源图是完全可简化的，否则是不可完全简化的。

**2）死锁定理：**S为死锁状态的充分条件是：当且仅当S状态的资源分配图不可完全简化。其中的有边进程为死锁进程。

**（3）死锁检测算法（略，见PPT）**

**（4）死锁检测算法的应用（略，见PPT）**

**2.死锁的解除**

**（1）终止所有死锁的进程：**代价大。易实现。

**（2）一次只终止一个进程，直到消除环路为止：**选择被终止进程（牺牲者）参考的因素：进程的优先权；进程已经执行了多久，离完成计算任务还有多长时间；进程使用了多少资源；进程还需要多少资源才能完成；有多少进程需要终止；是交互进程还是批处理进程。

**（3）剥夺资源：**选择一个牺牲者——代价最小.

**后退（也称回滚-Rollback）：**退回到安全状态，在此重新启动进程。

**饿死：**如果仅仅是基于代价来选择进程的话，某些进程可能会饿死。因此“代价”还应增加一个因素：做牺牲品的次数。

**第四章 存储器管理**

**存储器管理的功能：**

**（1）存储分配和回收：**是存储管理的主要内容。讨论其算法和相应的数据结构。

**（2）地址变换：**可执行文件生成中的链接技术、程序加载时的重定位技术，进程运行时硬件和软件的地址变换技术和机构。

**（3）存储共享和保护：**代码和数据共享，对地址空间的访问权限（读、写、执行）

**（4）存储器扩充：**它涉及存储器的逻辑组织和物理组织；（虚存技术）

**一、程序的装入和链接**

编程得到可执行文件的步骤：编译、链接、装入

**编辑：**得到如test.cpp ,a.asm等源文件

**编译：**从每个源文件得到对应的.obj目标文件

**链接：**将若干有关目标文件（在VC++环境中为一个workspace中的文件）及有关系统库目标文件进行链接，得到相应的可执行文件（PC机系统后缀为EXE的文件或动态链接文件DLL)，即装入模块。

**装入：**装入模块再由OS装入内存，创建进程。

逻辑地址空间和物理地址空间：

**逻辑地址：**由CPU执行指令时生成的地址，也称虚地址、相对地址

**物理地址：**实际的物理单元内存地址，也称绝对地址、实地址

**重定位：**存储管理模块要解决逻辑地址到物理地址的映射问题，称为重定位（地址映像工作在**编译或装入**阶段完成，则逻辑与物理相同，此时称为**静态重定位**；地址映像工作在**执行**阶段完成，则逻辑与物理不同，此时称为**动态重定位**）

**1.程序的装入**

**（1）绝对装入：**

**1）概念：**编译程序知道程序将驻留在内存的地址，产生绝对地址的目标代码；绝对装入模块装入时直接定位在上述内存地址，不需修改程序和数据的地址。

**2）优点：**其装入过程简单。

**3）缺点：**但过于依赖硬件结构，不适于多道程序系统。

**（2）可重定位装入：**

**1）概念：**在可执行文件中，列出各个需要重定位的地址单元和相对地址值（可重定位表），装入时再根据所定位的内存地址去修改每个重定位地址项，添加相应偏移量。

**2）优点：**不需硬件支持，可以装入有限多道程序

**3）缺点：**一个程序通常需要占用连续的内存空间，程序装入内存后不能移动。不易实现共享（地址变换在装入时进行，装入内存后不能移动，属于**静态重定位**）

**（3）动态运行时装入：**

**1）概念：**进程**开始执行时**，未全部装入内存，而是部分装入，运行时，需要哪个模块再装入哪个模块。程序装入内存后，并不立即将相对地址转换为绝对地址，地址转换推迟到程序执行时才进行，即**动态重定位**。装入内存中进程的**所有地址都是相对**的。

**2）优点：**OS可以将一个程序分散存放于不连续的内存空间，可以移动程序，有利于实现共享。能够支持程序执行中产生的地址引用，如指针变量。

**3）缺点：**需要硬件支持（通常是CPU），OS实现较复杂－－是虚存的基础

**2.程序的链接：**

源程序经过编译后，得到一组目标模块，再利用链接程序将这组目标模块链接，形成装入模块，根据链接时间的不同，链接分为以下三种：

**（1）静态链接：**

**1）概念：**在程序运行前，先将各目标模块及它们所需的库函数，链接成一个完整的装入模块，以后不再拆开

**2）缺点：**对多用户、多任务系统显然有冗余，比如多个用户调用了sin(x)，则每个目标代码中都有这部分代码，装入到内存则也都有这部分代码。

**（2）装入时动态链接：**

**1）概念：**源程序编译得到的目标模块是在装入内存时，**边装入边链接**的，即在**装入一个目标模块**时，若发现一个外部模块调用事件，装入程序去找出相应的外部目标模块，并将它装入内存，同时修改相对地址。

**2）优点：**共享：多个进程可以共用一个目标模块，节省内存，减少文件交换；

便于修改和更新，各目标模块是分开存放的，便于修改。

**（3）运行时动态链接**

**1）概念：**运行时动态链接是将某些模块的链接推迟到执行时。即，**执行**时发现调用的模块未被装入，由OS找到该模块并装入，并将其链接到调用者模块上。

**2）优点：**

**部分装入：**一个进程可以将多种操作分散在不同的DLL中实现，而只将与当前操作相对应的DLL装入内存。

**便于局部代码修改：**即便于代码升级和代码重用；只要函数的接口参数（输入和输出）不变，则修改函数及其DLL，无需对可执行文件重新编译或链接。

**便于适应运行环境：**调用不同的DLL，就可以适应多种使用环境和提供不同功能。如：不同的显示卡只需厂商为其提供特定的DLL，而OS和应用程序不必修改。

**3）缺点：**

**链接开销：**增加了程序执行时的链接开销；

**管理开销：**程序由多个文件组成，增加管理复杂度。

**二、连续分配存储管理方式**

**1.单一连续分配**

**（1）概念：**内存分为两个区域：系统区，用户区。应用程序装入到用户区，可使用用户区全部空间。未采取存储保护措施。最简单的分配方式，适合**单用户、单任务**的OS。

**（2）优点：**易于管理

**（3）缺点：**对要求内存空间少的程序，造成内存浪费；程序全部装入，很少使用的程序部分也占用内存

**2.固定分区分配**

**（1）概念：**是最简单的一种**运行多道程序**的存储管理方式。把内存划分为若干个固定大小的连续分区，**每个分区只装入一个作业**。

**（2）划分分区的方法：**

**1）分区大小相等：**只适合于多个相同进程的并发执行

**2）分区大小不等：**多个小分区、适量中等分区、少量大分区。根据程序大小分配

**（3）内存分配：**OS将分区按大小进行排队，并建立一张分区使用表或位示图



**（4）优点：**易于实现，开销小

**（5）缺点：**内碎片造成浪费，分区总数固定，限制了并发执行的程序数目

**（6）内碎片和外碎片：**内碎片指占用分区之内未被利用的空间；外碎片是指占用分区之间难以利用的空闲分区

**3.动态分区分配**

**（1）概念：**动态分区分配是指OS根据进程的实际需要为各进程分配连续的物理内存

**（2）动态分区分配数据结构：**空闲分区表和空闲分区链表，**下图为空闲分区表示例**



**（3）分区分配算法：**

**1）首次适应算法：**按分区的先后次序，从头查找，找到符合要求的第一个分区。

**2) 循环首次适应算法:** 按分区的先后次序，从上次分配的分区的下一个位置开始查找（到最后一个分区时再回到开头），找到符合要求的第一个分区。

**3）最佳适应法：**找到其大小与要求相差最小的空闲分区

**4）最坏（差）适应法：**找到最大的空闲分区。算法要求空闲分区表将空闲分区按容量由大到小排序。

**5）快速适应算法：**又称分类搜索法。将空闲分区根据容量大小分类，每类分区容量相同。为每类分区设立一个空闲分区链表。系统中设立一张管理索引表，每个表项记录的是每类空闲分区链表的表头。

**（4）分区分配操作：**

**1）分配内存：**按照具体分配算法进行分配

**2）回收内存：**（设前一空闲分区为F1，后一空闲分区为F2）

与前一个空闲分区相邻：只需修改F1的大小

与后一个空闲分区相邻：回收区首地址作为新空闲区首址，大小为两者之和

与前、后空闲分区都相邻：F1首地址作为新空闲区首址，大小为三者之和

不与任何空闲分区相邻：新建表项

**4.伙伴系统**

**（1）概念：**分配的分区和空闲分区都是固定大小都为2^k，且l<=k<=m。

**（2）分区分配方法：**开始时整个分区都是2^m，按照分区大小进行分类，每类建立一个空闲分区双向链表。系统中也要建立一张管理索引表，指明每个链表表头。分配时，找2^(i-1)<n<=2^i的分区，若2^i找不到就找更大的2^(i+1),将2^(i+1)分为两个2^i一个分配出去，一个加入到2^i大小的空闲队列。

**（3）分区回收方法：**若回收大小为2^i的分区，若有伙伴分区，则合并为2^(i+1)的分区，进而可能需要合并为2^(i+2)的分区

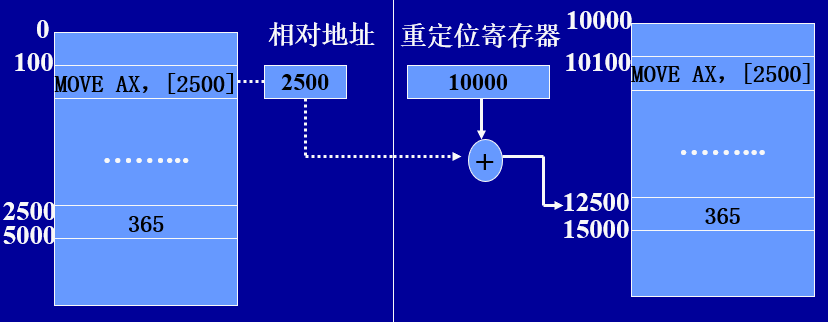
**5.可重定位分区分配（动态分区分配+紧凑）**

**（1）紧凑：**将各个占用分区向内存一端移动。使各个空闲分区聚集在另一端，合并为一个较大的空闲分区



对占用分区进行内存数据搬移**占用CPU时间**，如果对占用分区中的程序进行“浮动”，则其重定位需要硬件支持。**何时执行紧凑操作：**⑴每个分区释放后⑵内存分配找不到满足条件的空闲分区时

**（2）动态重定位**



**6.对换**

**（1）概念：**对换是将暂时不能执行的程序或数据送到外存中，从而获得空闲内存空间来装入具备运行条件的进程或进程所需要的程序和数据。交换单位为进程的整个地址空间。常与可重定位分区分配配合使用，称为滚进、滚出。

**（2）对换空间的管理：**在具有对换功能的OS中，外存被分为对换去和文件区。为了提高进程换入、换出速度，对换区采取连续分配方式。

**（3）进程的换入与换出：**

**1）换出：**当前执行进程需要更多内存或系统创建了高优先级进程，而无内存空间时，OS选择处于阻塞态且优先级低的进程传送到磁盘对换区，修改PCB，回收内存空间。

**2）换入：**OS定期查看进程的状态，在内存有空时，找出就绪且换出时间最久的进程换入内存。

**三、基本分页存储管理方式（区分请求分页）**

**1.分页存储管理的基本方法**

**（1）基本概念：物理内存**被划分为固定大小的**页框**(page frame)，也叫**页帧**。进程的**逻辑地址空间**也分成同样大小的**页**(Page)。程序加载时，分配其所需全部页，这些页不必连续。分页存储管理模式不支持虚存，需要装入整个作业才能运行。（**虚存为运行时动态装入**）

**（2）进程装载：**在装入一个进程时，需找空闲页框，OS要将这些页框分配给装入的进程，进程地址空间的**每个页占用一个页框**，进程占用的所有页框**不要求连续**。

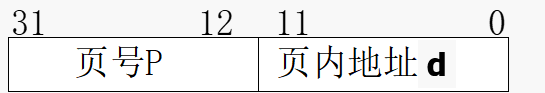
**（3）基本分页管理中的数据结构：**

**1）进程页表：每个进程**有一个页表，描述该进程的每个逻辑页占用的物理页框号。

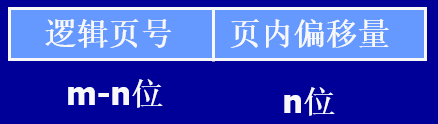
**2）物理页面表：整个系统**有一个物理页面表，描述所有物理页框的分配使用状况。

**3）请求表：整个系统**有一个请求表，描述系统内各个进程页表的位置和大小，用于地址变换。请求表也可以结合到各进程的PCB里，此时在PCB中记录本**进程页表所在的物理页框号**。上下文切换时，由OS将其加载到**页表寄存器**中。

**（4）逻辑地址结构：**一维逻辑地址被被分为：页号P、页内地址d。



P=INT[A/L] d=[A] MOD L （A：逻辑地址 L：页面尺寸）



逻辑地址空间为2m，页框大小为2n(m>n)。

**（5）页面大小的选择：**过大则内碎片少，但开销大、效率低（过小则相反）

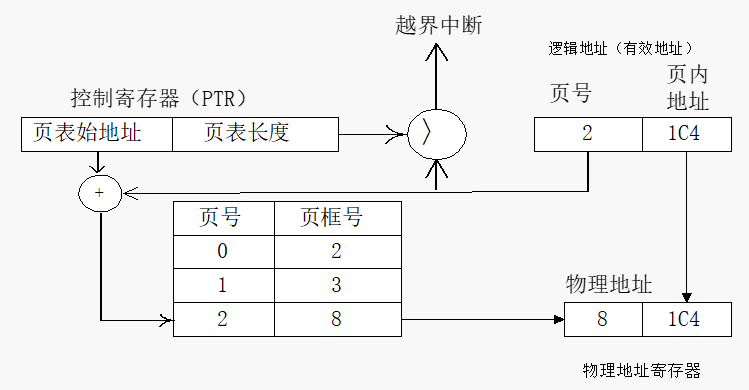
**（6）优缺点：**

**1）优点：**没有外碎片，内碎片也不超过页大小；且程序不必连续存放

**2）缺点：**程序需全部装入内存，无法支持虚存

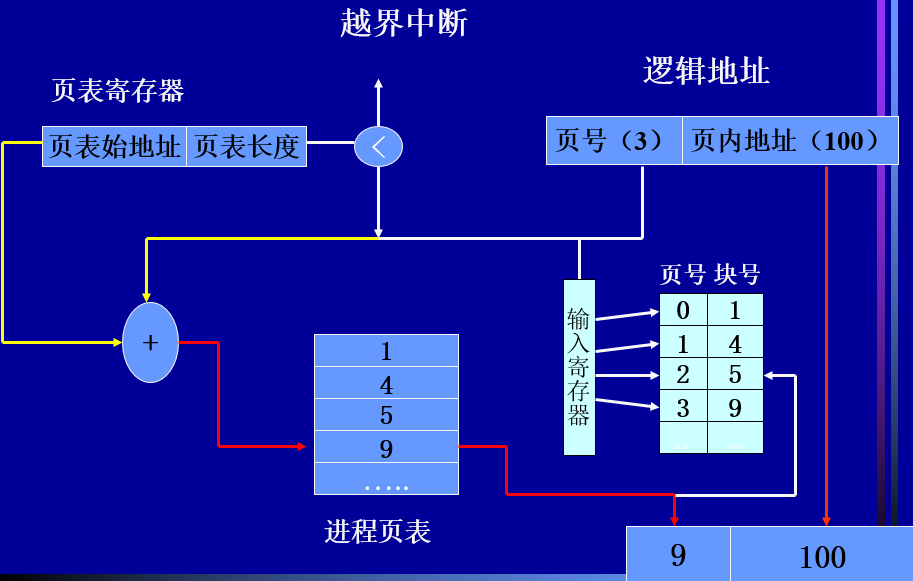
**2.地址变换机构**

**（1）基本地址变换机构：**通过查**进程页表**，得**物理页框号**，从而形成物理地址。



**（2）具有快表的地址变换机构：**页表利用内存储存，一次内存操作需要两次访存，效率低下。采用一种具有并行查找功能的“联想存储器（关联存储器）”可根据内容查找（**根据逻辑页号找物理块号**），根据程序局部性原理，将页表的一部分放在里面。

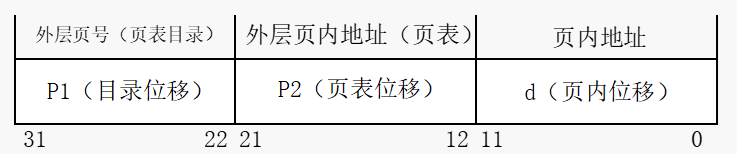
访存时先查找快表，命中则直接生成物理地址；未命中则查页表并按照一定的置换算法，置换快表中的页表项。



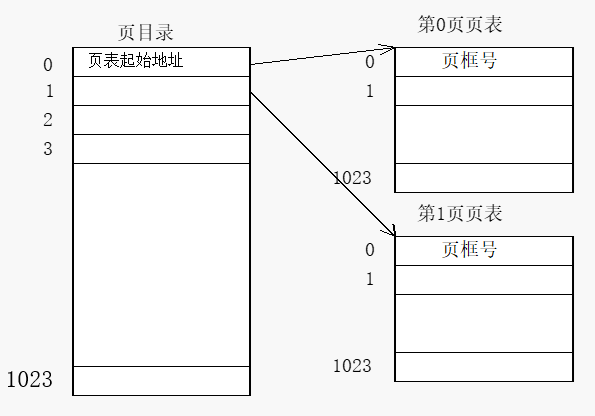
**3.多级页表**

**（1）两级页表：**增加页表的页表，即**外层页表**，也叫**页目录表**。页目录表也存放在内存中。此时，PCB中存放的是本进程对应的**页目录表所在的物理页框号**。（单级分页PCB中存放的是**进程页表所在物理页框号**）上下文切换时，由OS加载到专用寄存器。

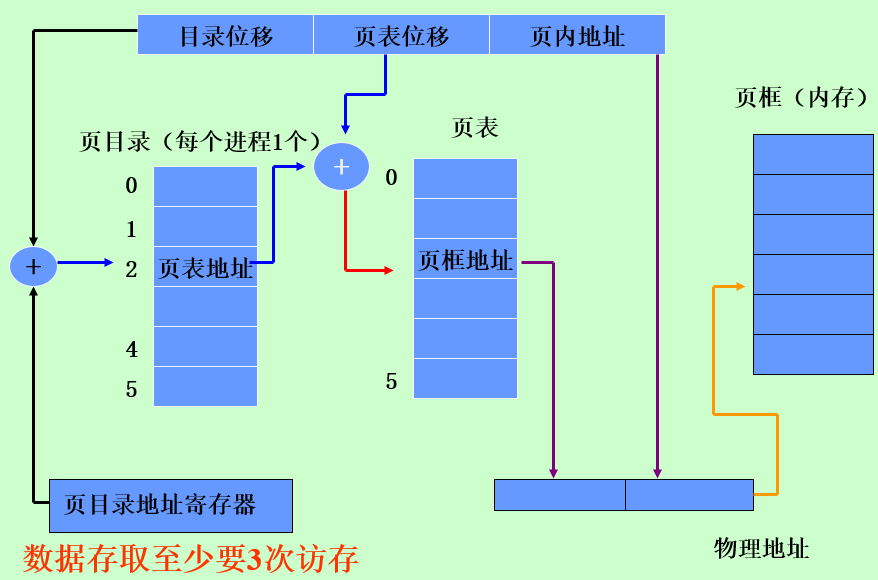
**1）逻辑地址结构：**



**2）两级页表结构：**

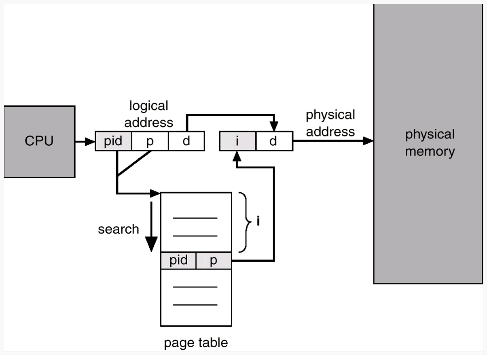


**3）地址变换机构：**



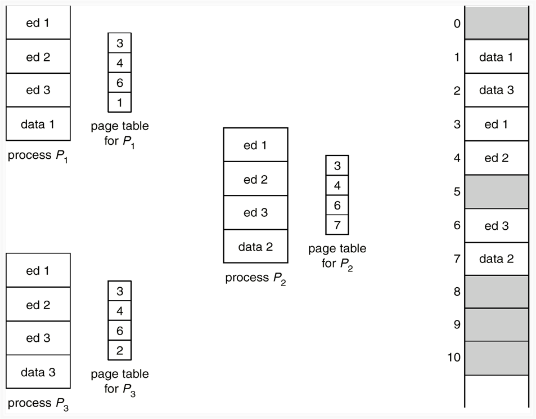
**（2）多级页表：**和二级页表类似。

**（3）反置页表：**按物理块号的顺序排序，内容为隶属的进程pid及其页号p。

（物理内存多少页框，页表就有多少项）

采用反置页表技术，整个系统中**只有一张页表**，每个物理页框在页表中都有一项。

**（4）页面共享：**多个进程的页表共用一些物理页框。（反置页表难以实现，因为反置页表中，其是根据进程pid和逻辑页号p查找，一个表项只能表示一个进程）



**四、基本分段存储管理方式**

**1.分段存储管理方式的引入**

**（1）方便编程：**按逻辑关系划分段：有独立的段名，各段的逻辑地址均从0开始。程序通过分段划分为多个模块，如代码段、数据段、共享段。可以分别编写和编译。

**（2）分段共享：**可以按段为单位来进行共享；分页不易于实现共享。

**（3）分段保护：**可以针对不同类型的段采取不同的保护措施

**（4）动态链接：**进程开始运行时，只装入主模块，运行中需要哪段再装入、链接。

**（5）动态增长：**如数据段根据运行需要可能会增大。

**（6）优点：**没有内碎片，外碎片可通过内存紧凑来消除。便于改变进程占用空间大小。

**（7）缺点：**基本分段管理要求进程全部装入内存。

**2.分段系统的基本原理**

**（1）段式管理的数据结构**

**1）进程段表：**每个进程一张段表，描述组成进程地址空间的各段在内存的起始地址（段基址）及段长。PCB中存放着本进程段表在内存的起始地址及段表长度。

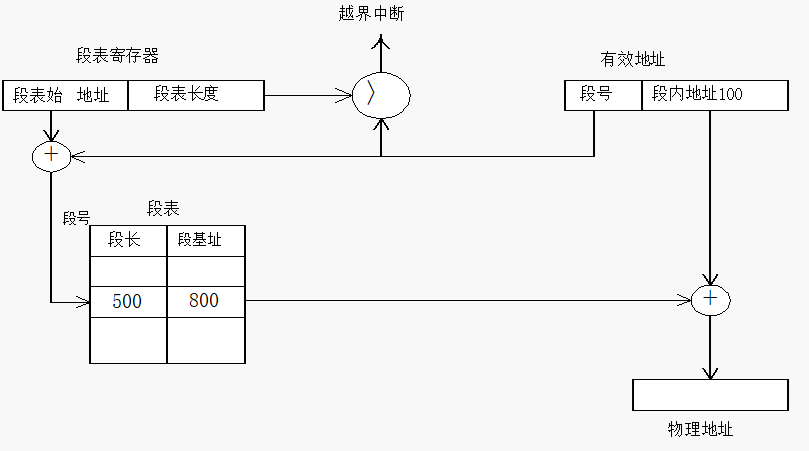
**2）系统段表：**描述系统内所有占用段的使用情况。

**3）空闲段表：**描述内存中所有空闲段，可以结合到系统段表中。

**段表：**由**段基址**和**段长**组成

**（2）逻辑地址结构：**二维结构：段号+段内地址

**（3）地址变换机构：**



**（4）分页和分段的区别：**

1）页是物理单位，而段是逻辑单位。分页是出于系统管理的需要，分段是出于用户应用的需要。

2）页大小是系统固定的，而段大小则通常不固定。

3）分段是二维的，程序员在标识一个地址时，既要给出段名，又需给出段内地址。

4）通常段比页大，因而段表比页表短，可以缩短查找时间，提高访问速度。

5）分段比分页系统更容易共享代码。

**3.信息共享**

分段比分页更容易共享，因为段按逻辑功能分，且比页大。

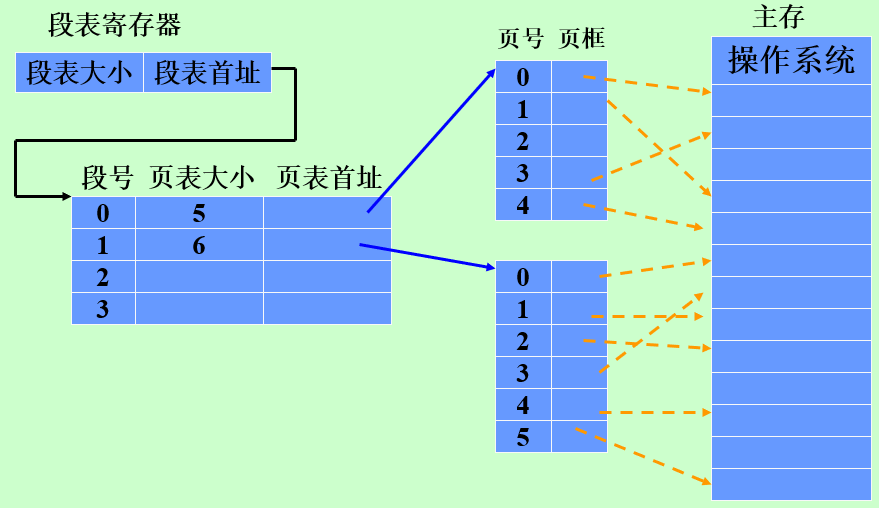
能共享的代码一定是**可重入代码，**也叫**纯代码**（pure code）是允许多个进程同时访问的代码，代码在执行过程中不能有任何改变。可能改变的变量拷贝到进程自己的数据区。

**4.段页式存储管理方式**

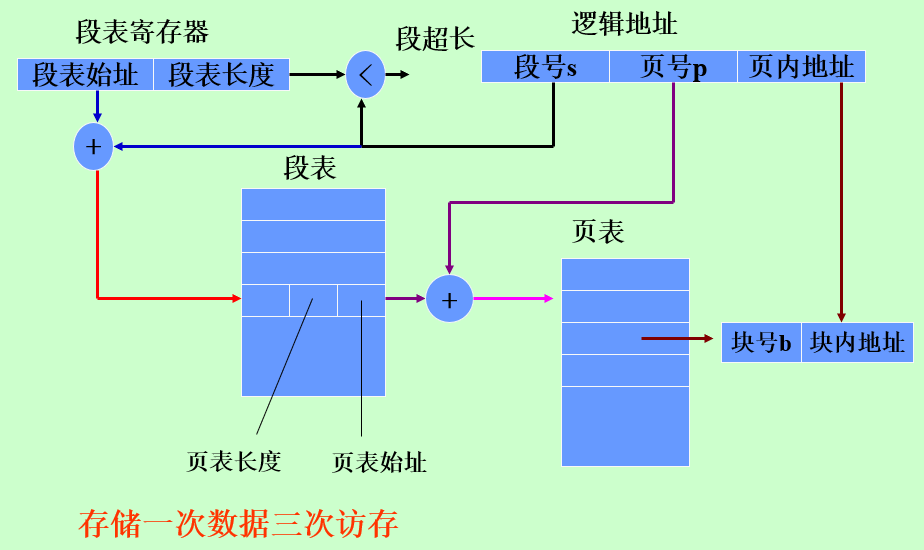
**（1）基本原理：**段内分页管理。先将用户程序分为若干段，每个段再分为若干页。地址变换为三维地址。



**（2）各种表格：**每个进程一张段表，**每个段一张页表**。段表寄存器给出当前运行进程的段表首地址及段表长度，段表中存放着与每段对应的页表的首地址及页表长度。PCB中存放段表的地址及长度.



**（3）地址变换机构：**



**第五章 虚拟存储器**

**一、虚拟存储器的基本概念**

**1.虚拟存储器的引入**

**（1）常规存储管理的特征：**

**1）一次性：**作业一次性全部装入内存，作业不能大于物理内存的实际尺寸，大于往往采用覆盖技术。

**2）程序的驻留性：**装入内存便驻留内存，阻塞时也在内存。但装入内存的程序和数据不一定马上使用。

**（2）局部性原理：**

**1）概念：**包括时间局部性、空间局部性

**2）具体体现：**程序中存在相当多的循环结构，它们由少量指令组成，而被多次执行。程序中存在相当多对一定数据结构的操作，如数组操作，往往局限在较小范围内。

**（3）虚拟存储器的定义：**

**1）部分装入：**在程序装入时，不需要将其全部装入到内存，而只需将当前需要执行的部分页或段读入到内存，就可让程序开始执行

**2）请求调入**：进程执行过程中，如果需执行的指令或访问的数据未在内存（称为**缺页**或**缺段**），则由处理器通知操作系统将相应页或段调入到内存，然后继续执行进程

**3）置换功能：**如果此时内存已满，操作系统将内存中暂时不使用的页或段调出保存在外存上，从而腾出空间存放将要调入的页或段

**4）虚拟存储器**：具有请求调入和置换功能，能从逻辑上对内存容量加以扩充的一种存储器系统。

**（4）虚拟存储器的特征：**

**1）多次性：**一个进程的程序和数据可分多次装入内存

**2）对换性：**允许将那些暂不使用的段或页面，从内存调至外存的对换区

**3）虚拟性：**进程逻辑地址空间可大于物理地址空间

**（5）引入虚拟存储技术的好处：**虚拟存储器是一种借助于**外存空间**，允许一个进程在其运行过程中**部分地装入内存**。可在较小的可用内存中执行较大的用户进程；可在内存中容纳更多进程并发执行；不必影响编程时的程序结构

**2.虚拟存储器的实现方式**

虚拟存储的实现是建立在**离散分配存储管理**方式的基础上，不可采用连续分配方式，可采用以下实现方式：

**（1）请求分页系统：**纯分页系统+请求调页+页面置换—**页式虚拟存储器**

**1）硬件支持：**请求分页页表：纯分页页表增加若干项；缺页中断；地址变换机构

**2）软件支持：**请求调页的软件；页面置换的软件

**（2）请求分段系统：**纯分段系统+请求调段+分段置换—**段式虚拟存储器**

**（3）段页式虚拟存储**

**二、请求分页存储管理方式**

**1.请求分页中的硬件支持**

**（1）页表机制：**每个进程一张页表，如下图：



**状态位P：**存在位（present bit)，中断位，描述该页在内存还是外存

**修改位M:**该页是否被修改过，又叫脏位

**访问字段A：**在近期被访问的次数，或最近一次访问到现在的时间间隔

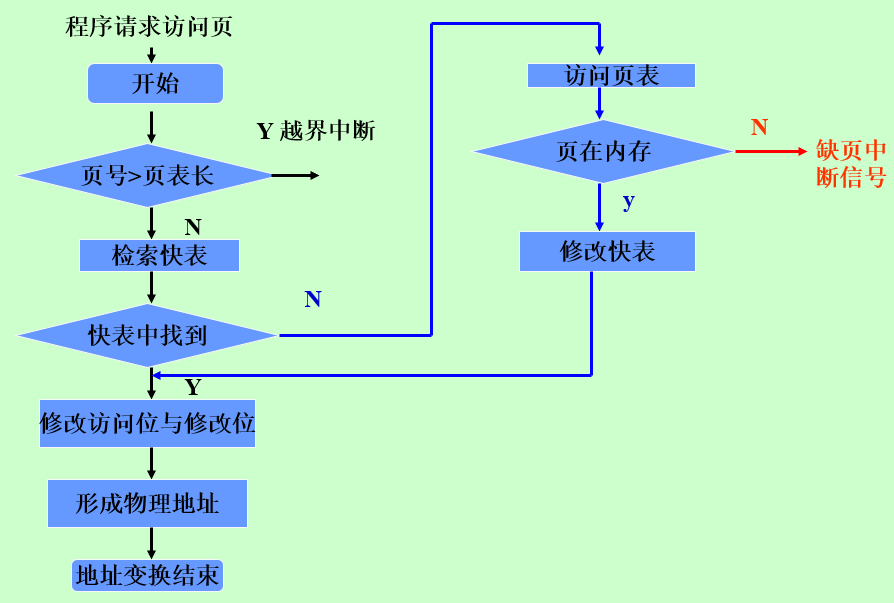
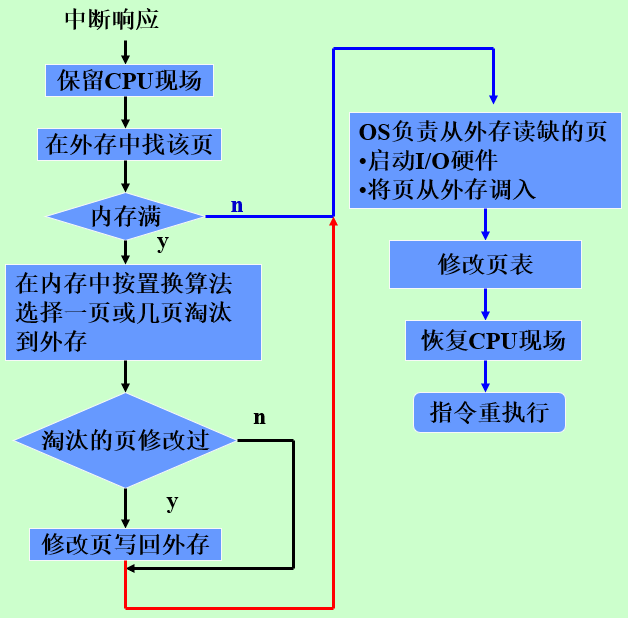
**外存地址：**磁盘上的地址

**（2）缺页中断机构：**每当进程要访问的页不在内存，则自陷，产生缺页中断。

1）缺页中断可能在**指令执行期间产生**并处理，而不一定是在一条指令执行完毕之后。所缺的页面调入之后，重新执行被中断的指令（**从取指开始**）。

2）一条指令的执行可能产生多次缺页中断

**（3）地址变换机构**—在原变换机构的基础上增加了缺页中断：

**2.请求分页中的内存分配**

**（1）最小物理块数的确定：**OS要保证进程运行的最少要分配的物理块数，少了则缺页中断频繁，进程无法运行。

**（2）物理块分配策略：**

**1）固定分配局部置换：**给每个进程分配固定数目的页框，当发生缺页中断时，只考虑从该进程所属的页框中调出旧的页面，换入新的页面。

**2）可变分配全局置换**：预先分配给进程一定数目的页框，**OS控制一定数量的空闲页框**，在进程的执行过程中，发生缺页时，OS就分配给该进程一个空闲的页框，当空闲的页框用完时，OS可根据需要从任意的进程中调出一个页框。

**3）可变分配局部置换：**预分配给进程一定数目的页框，**OS控制一定数量的空闲页框**，在进程的执行过程中，发生缺页时，**首先考虑**从该进程所属的页框中调出旧的页面，若发现该进程频繁发生缺页中断，再分配新的页框给该进程。（前两个策略的折中）

**（3）物理块分类算法：**

**1）平均分配：**将系统中所有可供分配的物理页框，平均分配给每个进程。

**2）按比例分配：**分配给进程i物理页框数=（进程i所需物理页框数÷所有进程所需物理页框数）\*物理内存可用的页框总数

**3）按优先权分配**

**3.页面调入策略**

**（1）何时调入页面：**

**1）预调页：**在发生缺页需要调入某页时，一次调入该页及相邻几个页。成功率50%。

**2）请求调页：**又叫纯调页，只调入发生缺页时所需的页面。

**（2）从何处调入页面：**

1）进程运行前，将其全部页面从文件区复制到交换区，以后总是从交换区调入。执行时调入速度快，要求交换区空间较大。

2）凡不会修改的页面或是未被修改的页面，直接从文件区调入，换出时不必写回磁盘，下次仍从文件区调入。已被修改的页面，被置换时需调出到交换区，以后从交换区调入。节省交换区空间。

3）UNIX方式，未运行过的页面，直接从文件区调入，而曾经运行过的页面，换出时放在对换区，下次调入时，应从对换区调入。在进程结束时，更新文件区内容

**（3）缺页率：**=缺页次数/内存访问次数

**（4）影响缺页率因素：**1）页面大小；2）进程分配物理页框数；3）页面置换算法；4）程序固有属性

**三、页面置换算法（重点！略，见PPT）**

**1.最佳置换算法**

**2.FIFO算法**

**3.最近最久未使用LRU算法**

**4.Clock置换算法**

**5.最少使用置换算法**

**6.页面缓冲算法**

**（1）两个链表:** 发生缺页时，用前面算法选择被置换页，被置换页放入两个链表之一

**空闲页面链表：**页面未被修改，就将其归入到空闲页面链表的末尾

**已修改页面链表：**页面被修改，将其归入到已修改页面链表

**（2）**需要调入新的物理页面时，将新页面内容读入到空闲页面链表的第一项，然后将第一项删除。

**（3）**空闲页面和已修改页面，仍停留在内存中一段时间，如果这些页面被再次访问，该页面可以返还作为进程的内存页。只需较小开销。否则，空闲页面链表的第一项分配出去，启动硬盘调入。

**（4）**当**已修改页面**达到一定数目后，再将它们一起调出到外存，然后将它们归入空闲页面链表，这样能大大减少I/O操作的次数

**四、请求分页系统实现中的一些问题**

**1.缺页率对有效访问时间的影响**

**有效访问时间**=（1-P）\*ma + P\*缺页中断时间

（其中: 设ma为主存的访问时间，约为10~200ns,缺页率为 P）

考虑快表命中率和缺页率，**完整的内存有效访问时间是**：

EAT=a\*(λ+t)+(1-a)\*[(1-f)\*(λ+t+t)+f\*(λ+t+ε+λ+t)]

(λ：快表访问时间 a：快表命中率 f：缺页率 ε：缺页中断处理时间 t:内存访问时间)

**2.其他一些问题**

**（1）内存页锁定：**正在**进行I/O操作**的页不应该换出，即要锁定。另一种解决是I/O操作在核心缓冲区进行（不占用用户区页框），操作完成后复制到用户页。

**（2）程序结构：**访问数组所带来得缺页问题（按行存储和按列存储）

**（3）页框大小发展趋势：**采用多种页框大小。针对不同规模的数据结构采用不同的页面大小，如**代码段认为是大规模的**，而**每个线程的栈认为是小规模的**

**五、抖动与工作集**

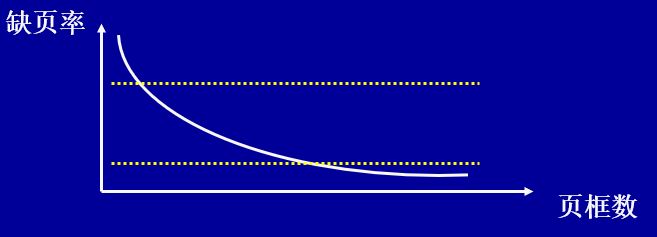
**1.多道程序的度与“抖动”**

随着驻留内存的进程数目增加，或者说**进程并发度**(multiprogramming level)的上升，处理器利用率**先是上升，然后下降**。

随着并发进程数目的增加，每个进程所分到的物理页框数不断减小，缺页率不断上升，使运行进程的**大部分时间都用于页面置换**，而几乎不能完成任何有效的工作，即发生了**抖动**。

**2.工作集**

**（1）缺页率与页框数的关系**



每个进程的缺页率随着所分到的物理页框数的增加而降低。但增加到一定程度时，缺页率不再明显下降。该数目是上述曲线上的拐弯点。

**（2）工作集策略：**

工作集是在某段时间间隔△里，进程实际要访问页面的集合。可用一个二元函数W(t, △)表示（**其中：**t是执行时刻；△是一个虚拟时间段，称为窗口大小(window size)）

引入工作集的目的是依据进程在过去的一段时间内访问的页面来调整进程分配到的物理页框数目。即用过去预测未来。

**（3）利用工作集动态调整多道程序度：**OS跟踪每个进程的工作集，并为进程分配大于其工作集的帧数。如果有空闲帧，可以**启动另一个进程**。如果所有工作集之和大于可用帧（页框）总数，则OS选择**挂起某个进程**。工作集策略防止了抖动，提高了多道程序的并发度。

**3.抖动的预防方法**

（1）CPU调度程序中**引入工作集算法**。保证为每个进程分配的物理页框数能包括该进程当前的局部性。

（2）挂起若干进程

（3）采用**局部置换**策略，可以把抖动局限在较小的范围。但某进程发生抖动，会长时间处于磁盘I/O等待队列中，使其他进程的缺页中断处理时间加长。

**（4）L=S准则：**L：产生缺页之间的平均时；S：平均缺页服务时间

L远大于S，说明缺页率低，但利用率低。

L小于S，缺页率高，缺页的速度超过磁盘的处理能力。

L=S，此时CPU和磁盘的利用率最高。

**六、请求分段存储器管理方式**

**1. 请求分段中的硬件支持**

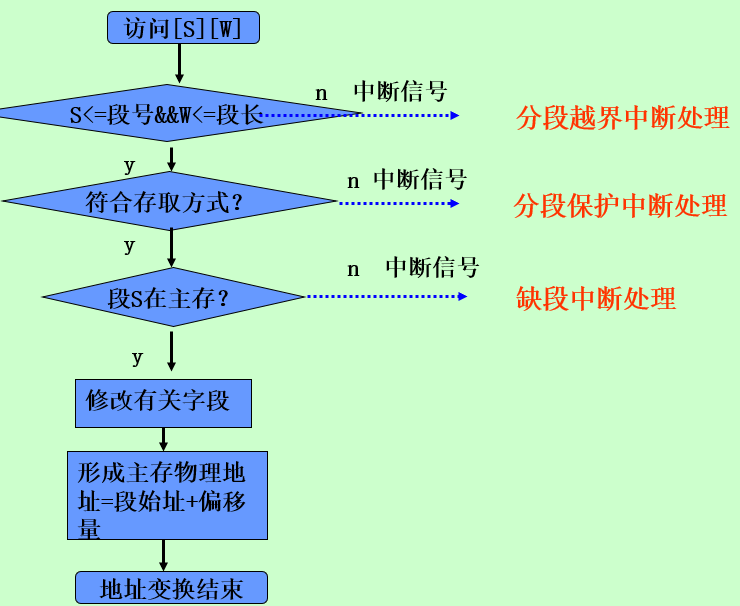
**（1）段表机制：**



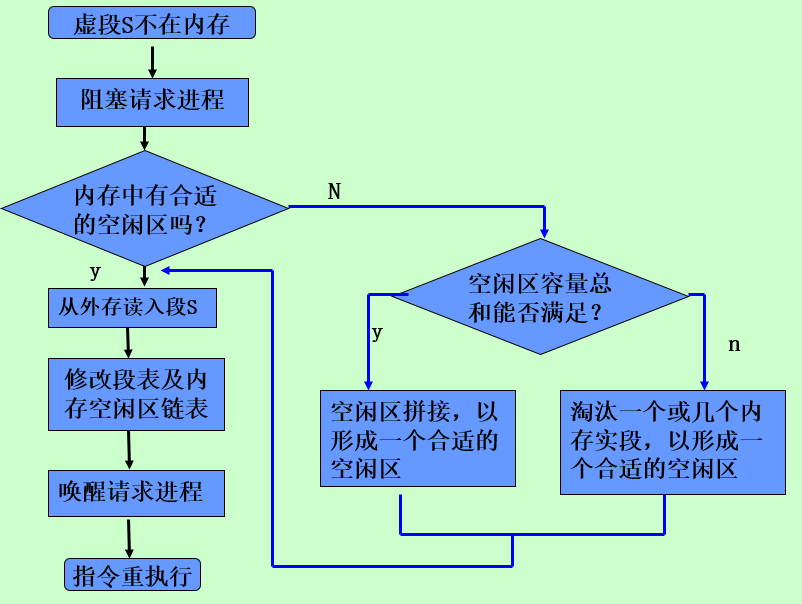
**存取方式：**只执行、只读、读/写。

**增补位：**请求分段系统中特有的，表示本段在执行过程中是否动态增长

**（2）地址变换机构工作流程**



**（3）缺段中断处理**



**2.分段共享与保护**

**（1）共享段表：**为了实现共享段，**系统中配置一张**共享段表，**所有共享段**在该表中都有一项。



**共享进程计数：**仅当该字段为0，才能由系统回收该段占用的内存。

**存取控制：**不同进程对同一共享段有不同的存取权限限制。

**段号：**不同进程，同一个共享段有不同段号。

**（2）共享段的分配与回收：**

**1）共享段的分配：**首次使用共享段时，需要将共享段调入内存，填写共享段表项，置count=1；其它进程再次使用该段时，只需填写共享段表项，将count加1即可。

**2）共享段的回收：**进程不再需要使用某个共享段时，修改共享段表项，将count减1，若count = 0，表明没有进程需要使用该段，系统可将该段占用的内存回收。另外，共享段通常不被交换到外存。

**（3）分段保护**

**1）越界检查**

**2）存取控制检查**

**3）环保护机构（指内存）**：低环具有较高优先权。 OS核心处于0环；重要的实用程序和操作系统服务占据中间环；一般的应用程序在外环。**访问和调用原则：**一个程序可以访问驻留在相同环或低优先权环的数据；一个程序可以调用驻留在相同环或高优先权环的服务

**第六章 输入输出系统**

**一、I/O系统的功能、模型和接口**

**1.I/O系统功能**

**（1）隐藏物理设备的细节**

**（2）设备无关性：**使用逻辑操作和逻辑设备名掩盖设备的物理细节。

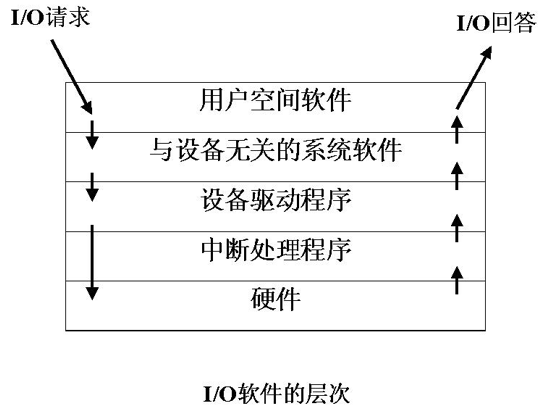
**（3）提高处理机和I/O设备的利用率：**软硬结合，如中断技术、缓冲技术、设备共享和假脱机技术等等。

**（4）对I/O设备进行控制：**轮询、中断、DMA、通道

**（5）确保设备的正确共享：**独占式（打印机）、共享式（磁盘）

**（6）错误处理：**应尽可能在**接近硬件**的层面上进行错误处理，只有低层软件不能处理的错误才向高层报告。

**2.I/0系统层次结构**



**（1）用户级软件（指用户空间的I/O软件）：**用户与设备管理模块的接口，负责解释用户的应用请求，并将这种请求转化为具体的输入/输出操作。

**（2）设备独立性软件(设备无关层）：**负责实现与设备驱动程序的统一接口，设备命名，**将逻辑设备名转换为物理设备名**，实现设备的分配和回收，缓冲区管理，为设备管理和数据传送提供必要的存储空间。

**（3）设备驱动程序：**与硬件直接相关，负责具体实现系统对设备发出的操作命令。设置设备寄存器，检查设备状态。**不同设备有自己的驱动程序**，但提供了一组标准接口。**每种OS都有自己的设备驱动程序接口**，故一个特定设备可能有多种设备驱动程序。

**（4）中断处理程序：**中断处理程序是设备驱动程序中的一部分

**3.I/O系统接口（I/0系统与高层之间的接口）**

**（1）块设备接口：**信息的存取以数据块为单位，如磁盘（DMA方式），速率高，可寻址

**（2）流设备接口：**信息的存取以字符为单位，如打印机（中断），速率低，不可寻址

**（3）网络通信接口**

**二、设备驱动程序**

**1.设备驱动程序功能**

（1）将上层软件发来的抽象要求转换为具体的要求，发送给设备控制器。

（2）检查用户I/O请求合法性，了解设备状态，传递有关参数，设置设备的工作方式。

（3）发出I/O命令，设备空闲则启动。设备忙，则将请求者的请求块挂在设备队列上。

（4）及时响应通道或控制器发来的中断请求，调用相应的中断处理程序。

（5）对于设置有通道的，自动构成通道程序

**2.设备驱动程序特点**

（1）设备驱动程序是在**I/O请求进程**与**设备控制器**之间的桥梁，中转数据和控制。

（2）设备驱动程序与I/O设备特性、控制方式与硬件密切相关，一般由厂商提供。

（3）**向上屏蔽设备细节：**不同类型设备通常其设备驱动程序接口不同，**同类设备的接口相同**。因此，同类设备的不同型号，只要更换设备驱动程序则可由OS使用。

（4）设备驱动程序与硬件密切相关，其中一部分必须用汇编编写，很多设备驱动程序的基本部分已经固化在ROM中。

（5）驱动程序**允许可重入**，一个正在运行的驱动程序可以被再次调用。

**3.设备处理方式：**

（1）为每类设备设置一个进程

（2）作为系统进程执行：整个系统设置一个I/O进程，负责对各类设备的I/O操作；也可设置一个输入进程和一个输出进程（“守护进程”）

（3）不设专门设备处理进程，各类设备有自己设备驱动程序，供用户或系统进程调用。

**4.处理过程：**

（1）将抽象要求转化为具体要求。如将逻辑盘块号转换为物理地址。

（2）检查I/O请求的合法性。

（3）读出和检查设备的状态

（4）传送参数

（5）设置工作方式

（6）启动I/O设备

**三、与设备无关的I/O软件**

**1.与设备无关软件（设备无关性）的基本概念：**

指除了直接与设备打交道的底层软件之外，其它部分软件并不依赖于硬件。可以提高软件的设计效率。应用程序使用逻辑设备名调用设备；OS实际执行时，使用物理设备名。

**设备独立性的好处：**（1）设备分配时的灵活性（2）实现I/0重定向：实现I/O操作的设备可以更换，而不必改变应用程序

**2.与设备无关的软件**

是I/O系统的最高层软件。其中包括了执行所有设备公有操作的软件，有：

**（1）设备驱动程序的统一接口**

**（2）对独占设备分配与回收：**防止独占设备争夺，系统统一分配，禁止用户直接访问

**（3）差错控制：**只处理设备驱动程序无法处理的错误。

**（4）缓冲管理。**

**（5）提供独立于设备的逻辑块：**不同设备信息交换单位是不同的，设备独立性软件负责隐蔽这些差异，向逻辑设备和高层软件**提供统一的逻辑数据块**。

**3.设备分配**

**（1）设备分配中的数据结构：**

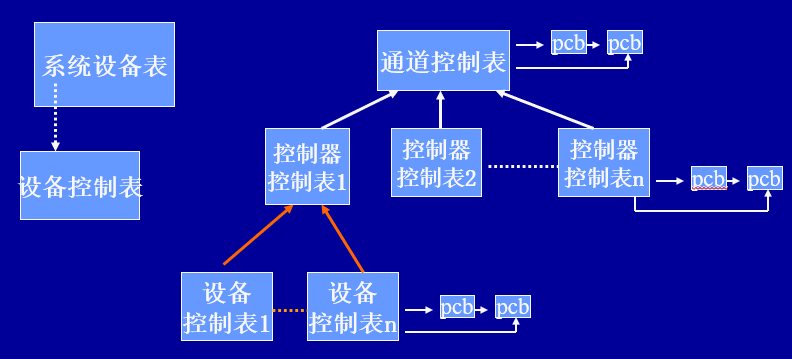
**1）设备控制表DCT：**每个设备一张，描述设备特性和状态,反映设备和控制器的连接情况。

**2）系统设备表SDT：**系统内一张，反映系统中设备资源的状态，记录所有设备的状态及其设备控制表DCT的入口（指针）。

**3）控制器控制表COCT：**每个设备控制器一张，描述I/O控制器的配置和状态。

**4）通道控制表CHCT：**每个通道一张，描述通道工作状态。

**5）表格之间关系：**



**（2）设备分配时应考虑的因素：**

**1）考虑设备固有属性：**独享设备；共享设备；虚拟设备

**2）设备分配算法：**FCFS;基于优先级

**3）设备分配中的安全性**

**安全分配：**进程发出I/O请求后阻塞，完成后唤醒，即进程运行时不占有任何设备资源，摒弃了“请求保持”条件。但I/0设备与CPU是串行工作

**不安全分配：**进程发出I/O请求后仍继续运行，又可发出I/O请求，仅当进程所请求的设备已被另一进程占用时，进程才进入阻塞状态。满足了多个设备并行工作，但是可能造成死锁。

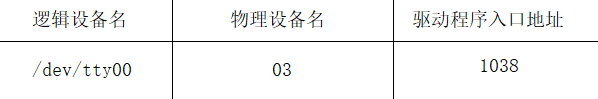
**（3）独占设备的分配程序**

1）进程**以物理设备名请求**设备时，分配方式为：设备 -> 控制器 -> 通道

2）进程**以逻辑设备名请求**设备时，系统要在同类设备中查找是否有空闲的，仅当所有该类设备都忙时，才会阻塞进程。

**4.逻辑设备名到物理设备名映射的实现**

**（1）设备逻辑表LUT：**当进程用逻辑设备名请求I/O设备时，系统为它分配相应的物理设备，并在LUT中建立一个表目



**（2）LUT的设置可采用两种方式：整个系统一张**，不能有相同逻辑设备名。**每个用户一张**，放入PCB中，不同用户逻辑设备名可重名。

**四、用户层的I/O软件（假脱机Spooling系统）**

**1.什么是Spooling**

**（1）概念：**

假脱机技术是将独享设备**模拟**为具有共享特征的虚拟设备。在多道程序系统中，利用一道**程序模拟脱机输入**时的外围控制机，把低速I/O设备上的数据传送到磁盘上；再用另一道**程序模拟脱机输出**时的外围控制机，把数据从磁盘传送到慢速的设备上。这样便可在主机的直接控制下，实现脱机输入输出操作。此时的**外围操作与CPU对数据的处理同时进行**，这种在联机情况下实现的同时外围操作称SPOOLing，也叫假脱机。

**（2）原理：**

**1）实际I/O：**SPOOLing程序和外设进行数据交换，可以称为“实际I/O”。一方面，SPOOLing程序预先从外设输入数据并加以缓冲（输入井），在以后需要的时候输入到应用程序；另一方面，SPOOLing程序接受应用程序的输出数据并加以缓冲（输出井），在以后适当的时候输出到外设。

**2）虚拟I/O：**应用程序进行I/O操作时，只是和SPOOLing程序交换数据，可以称为“虚拟I/O”。这时候的虚拟I/O实际上是SPOOLing程序从缓冲池中读出数据（输入）或把数据送入缓冲池（输出），而不是跟实际的外设进行I/O操作。

**2.Spooling系统的组成**

**（1）输入井和输出井：**在磁盘上，一般以文件形式进行管理，称为井文件。

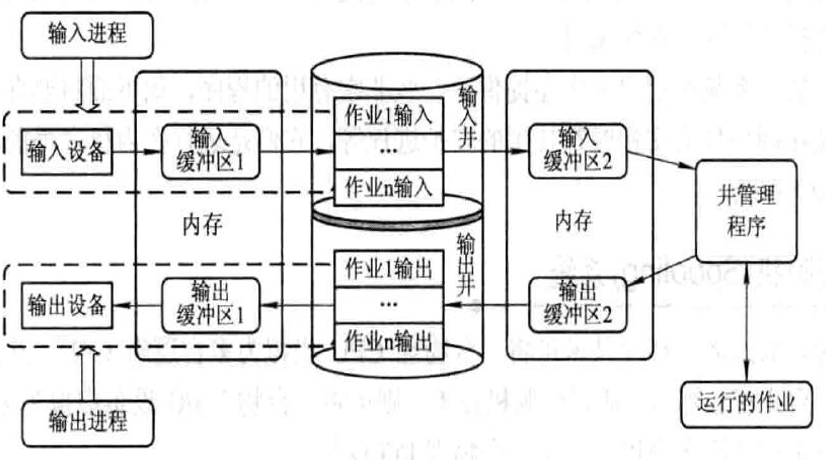
**（2）输入缓冲区和输出缓冲区：**在**内存**。缓和CPU和磁盘速度不匹配的矛盾。

**（3）输入进程和输出进程：**

1）输入进程也称为预输入进程，将用户的输入数据从输入设备传送到输入缓冲区，再放到输入井。

2）输出进程也称为缓输出进程，将用户要输出的数据从输出缓冲区传送到输出井。当输出设备空闲时，再由输出井经内存缓冲区送输出设备。

**（4）井管理程序：**用于控制**作业与磁盘井**之间的信息交换。当作业执行过程中向某外设发出I/O请求，由OS调用井管理程序，由它控制从输入井读入信息或输出到输出井上。



**3.Spooling系统特点：**

**（1）提高了I/O操作速度：**虚拟I/O比实际I/O速度快，缩短应用程序执行时间

**（2）实现对独享设备的共享：**未为任何进程分配设备，是在输入井和输出井中为进程分配了空闲盘块

**（3）实现了虚拟设备：**每个使用独占设备的进程都认为是自己独占了一个设备

**（4）守护进程：**无事可做时它们是处于睡眠状态。有工作时要被唤醒。凡需要将独占设备虚拟为多个逻辑设备时，都需要一个守护进程和一个假脱机文件队列。**守护进程是允许使用该独占设备的唯一进程。**

**五、缓冲管理**

**1.引入缓冲区的原因**

（1）缓和CPU和I/O设备间速度不匹配的矛盾

（2）协调传输数据大小不一致的设备。

（3）减少对CPU的中断频率

（4）提高CPU和I/O设备之间的并行性

**2.单缓冲**

一个缓冲区，CPU和外设轮流使用，一方处理完之后接着等待对方处理。

**3.双缓冲**

设备输入时，先将数据送入第一缓冲区，装满后再转向第二缓冲区。此时OS可以从第一缓冲区移出数据，送入用户进程所在内存。CPU和外设都可以连续处理而无需等待对方。

**4.循环缓冲**

环形缓冲(circular buffer)：多个缓冲区，CPU和外设的处理速度可以相差较大。

**5.缓冲池（供所有进程使用）**

**（1）缓冲池的组成：**既可用于输入，又可用于输出，缓冲池中有三类缓冲区：

**1）空闲缓冲区队列emq**

**2）输入缓冲区队列inq：**由装满输入数据的缓冲区链成的队列。

**3）输出缓冲区队列outq：**由装满输出数据的缓冲区链成的队列。

**（2）对缓冲池的操作：**

**1）收容输入：**设备输入数据

**2）提取输入：**计算进程读入数据

**3）收容输出：**计算进程输出数据

**4）提取输出：**向设备输出数据。

**六、磁盘存储器管理**

**1.磁盘性能简述**

**（1）地址格式：**驱动器号、磁道号（柱面号）、磁头号（记录面号）、扇区号

**（2）磁盘的类型：**固定头磁盘；移动头磁盘

**（3）磁盘访问时间：**

**1）寻道时间：**与寻道距离成正比（影响磁盘访问时间的主要参数，占比最大）

**2）旋转延迟时间**

**3）传输时间**

**2.磁盘调度算法（重点！略，见PPT）**

**3.磁盘高速缓存**

**（1）磁盘高速缓存形式：**指利用内存空间暂存从磁盘中读出的盘块信息。是一组逻辑上属于磁盘，物理上驻留内存的盘块。

**（2）数据交付方式：**某进程访问磁盘数据，OS核心首先查看磁盘高速缓存，若有则拷贝。否则访问磁盘，将所需数据交付请求进程，同时在高速缓存中复制一份，若以后需要可直接交付。**两种交付方式：**

1）将高速缓存中的数据直接传送到请求进程的内存区。

2）交付给请求进程的是指向所需数据的指针。

**（3）置换算法：**类似于请求调页系统。将磁盘中的数据读入高速缓存时，若高速缓存中已装满数据，则需要置换。

**（4）数据一致性：**系统发生故障，可能会造成内存高速缓存中的数据与磁盘数据不一致。解决方案：1）备用电源；2）OS定期回写机制（UNIX方法，定期回写dirty块）

**4.提高磁盘I/O速度的其它方法**

**（1）提前读：**在将当前请求盘块读入磁盘高速缓存时，可将下一个盘块的数据同时预先读入磁盘高速缓存，从而减少磁盘I/O。

**（2）延迟写：**缓冲区A本应立即写入磁盘，OS不马上写，而是将其置于空闲缓冲区队尾。这样，若缓冲区A中数据在后期又被访问，可在此处找到，减少磁盘I/O。当缓存区A要被使用时，再将其写入磁盘，腾出缓冲区空间。

**（3）优化物理块分布：**同一文件所占用的盘块应尽量安排在同一条磁道或相邻磁道上

**（4）虚拟盘：**利用内存空间去仿真磁盘，又称为RAM盘。可接收所有标准磁盘操作，只不过操作在内存中进行，并且由OS控制，对用户透明。但虚拟盘是易失的。

**第七章 文件系统**

**一、文件和文件系统**

**1.文化、记录和数据项**

**（1）数据项：**

**1）基本数据项：**可命名的最小数据单位，原子数据，有数据类型。

**2）组合数据项：**由若干基本数据项组合而成

**（2）记录：**一组相关数据项集合，用于描述一个对象在某方面的属性。

**（3）文件：**由创建者定义，具有文件名的一组相关元素的集合。文件名是文件的符号名。**有结构文件**由若干相关记录组成；**无结构文件**被看成是一个字符流。**文件包括两部分：**

**1）文件体：**文件本身信息

**2）文件属性（文件控制块->目录文件）：**文件名、文件类型、长度、物理位置、存取控制、创建人、创建时间、修改时间等等

**2.文件类型**

**（1）用途：**系统文件、用户文件、库文件

**（2）文件中的数据形式：**源文件、目标文件（.obj）、可执行代码文件（.exe）。

**（3）存取控制属性：**只执行文件、只读文件、读写文件

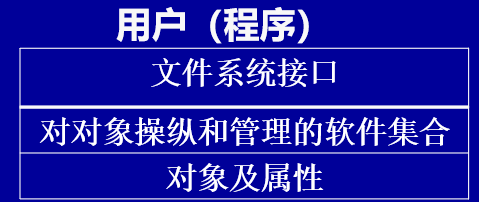
**（4）组织形式和处理方式：**

**普通文件：**由ASCII码或二进制码组成的文件

**目录文件：**由文件目录组成

**特殊文件：**指系统中的各类I/O设备。为了便于统一管理，系统将所有I/O设备都视为文件，只是对它们的操作由驱动程序完成。

**3.文件系统层次结构**



**（1）对象及属性：**文件系统管理的对象有：文件、目录、磁盘存储空间。

**（2）对对象操纵和管理的软件集合：**文件系统核心部分，分为以下四层：

**1）I/O控制层：**最底层，由设备驱动程序构成

**2）基本文件系统层：**主要处理内存与磁盘之间的数据块交换

**3）基本I/O管理层：**完成与磁盘I/O有关的事务，如将文件逻辑块号转换为物理块号，管理磁盘中的空闲盘块，I/O缓冲的指定等。

**4）逻辑文件系统：**处理与记录和文件有关的操作，如允许用户和应用程序使用符号文件名访问文件及记录，实现文件和记录的保护等。

**（3）文件系统的接口：**以接口的形式提供了一组对文件和记录操作的方法和手段。两种类型的接口：命令接口及程序接口

**4.文件操作：**

**（1）基本文件操作：**

**1）创建文件：**分配外存空间，在文件系统的目录中建立一个目录项。

**2）删除文件：**在目录中找到要删除文件的目录项并删除，同时回收空间。

**3）读文件：**应用进程调用系统调用，系统查找该文件的目录项，确定外存地址，目录项中有读写指针。

**4）写文件：**类似读文件。

**5）设置读写位置：**前面的读写操作每次从文件的起始位置读写。本操作用于设置读写指针，从需要位置开始。即**将顺序存取改为随机存取**。

**（2）文件的打开与关闭**

**1）打开：**系统将指定文件的目录项复制到内存中打开文件表中为其建立一个表目，并将该表目的编号（索引号）返回给进程。避免多次访问外存获取文件属性信息。

**2）关闭：**将内存中对应的文件目录项复制到外存目录表中，从内存打开文件表中删除对应的目录项。

**（3）其他操作（以系统调用形式提供用户）**

**1）关于文件属性的操作：**改变文件名、改变文件所有者、改变文件的访问权限等。

**2）有关目录操作的：**创建目录、删除目录等。

**3）实现文件共享的操作**

**二、文件的逻辑结构**

**文件的逻辑结构（File Logical Structure）:**也称文件的组织（File Organization），是指从用户观点出发讨论文件组织结构，是用户可直接处理的数据及结构，独立于文件的物理特性。

**1.文件逻辑结构类型**

**（1）按文件是否有结构分：**

**1）有结构文件：**记录式文件，有定长记录和变长记录两种形式。

**2）无结构文件：**文件体为字节流，不划分记录，顺序访问。

**（2）结构文件按照组织方式分类：**

**1）顺序文件：**文件中的记录按照某种顺序排列，适合于定长记录文件

**2）索引文件：**记录长度可变，建立一张索引表，每个记录一个表项，加快检索。

**3）索引顺序文件：**建立索引表，一组记录一个表项（以上两种方式结合）

**2.顺序文件**

**（1）逻辑记录的排序**

**1）串结构：**记录顺序与关键字无关，存入时间决定顺序。

**2）顺序结构：**记录按关键字排序，检索效率高。

**（2）对顺序文件的读写**

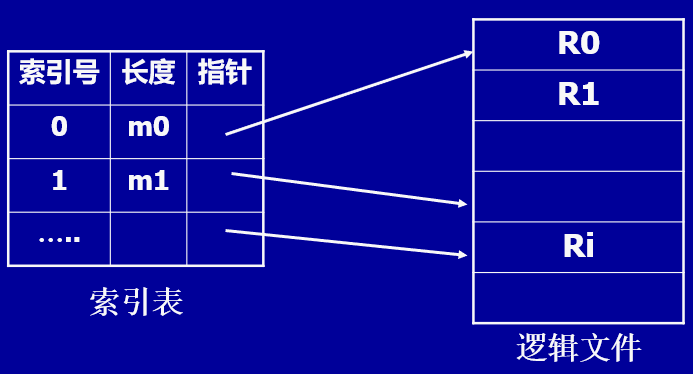
**1）定长记录：**读/写指针每次读/写一个记录后ptr=ptr+L使指针指向下一个记录。

**2）变长记录：**上述的L变为变长即可

**（3）优缺点：**处理数据速度快，但对单个记录处理困难，插入或删除尤其如此。（磁带）

**3.索引文件**

另外建立一张索引表，每个记录在表中对应一个索引项，索引项按照记录中的某个关键字域排序。索引文件的记录项通常较小，查找速度快，便于随机访问。



**4.索引顺序文件**

将顺序文件中的所有记录分为若干组；为顺序文件建立一张索引表，每组的第一个记录在索引表中有对应表项。

查找任意记录时，先据关键字查索引表（此时可采用各种查找算法），找到所在组的第一个记录，之后顺序查找该组。



**5.直接文件和哈希文件**

**（1）直接文件：**直接文件是根据记录的键值直接就可获得记录的物理地址。

**（2）哈希文件：**一种直接文件，通过哈希函数进行键值到物理地址的转换。

**三、文件的物理结构（外存分配方式）**

**文件的物理结构：**又称文件的存储结构，文件在外存上的组织形式，与存储介质的存储性能及所采用的外存分配方式有关。

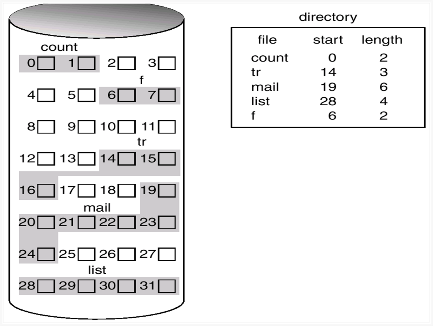
**簇：**又称为部分或**盘块**。文件的存储空间通常由多个独立的簇组成，而每个簇包含若干个连续的外存存储单位（如扇区sector）。

**（1）簇大小的确定：**大到能容纳整个文件，小到一个外存存储块

**（2）簇大小确定方法：**簇大小可变；簇大小固定

**1.连续分配**

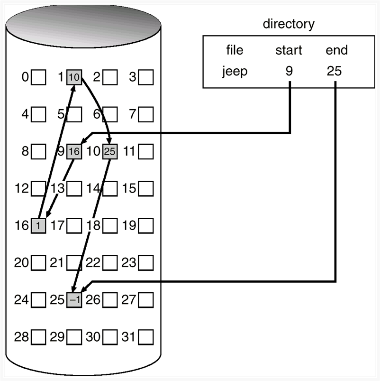
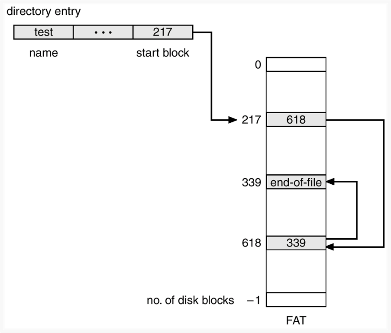
每个文件分配一组相邻接的盘块，也称物理顺序文件



**2.链接分配**

**（1）隐式链接：**分配给文件的盘块不连续，在每个簇中有指向下一个簇的指针。文件目录项中只存放文件数据所存储的第一块的盘块号。有的文件系统可能还会在目录项中存放文件的最后一块的簇号。

**（2）显示链接：**指针单独存放在一张表中，称文件分配表（FAT），文件目录项中存放文件首块的地址，FAT表中的序号与物理块号对应。

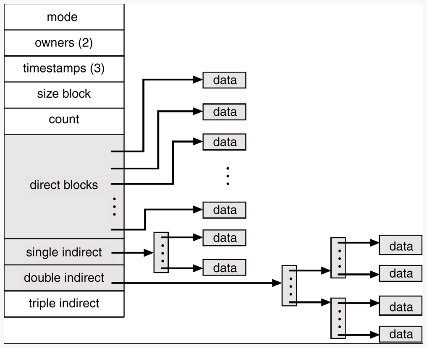
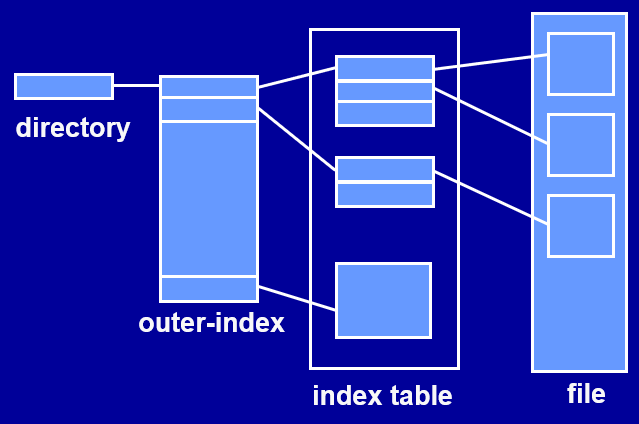
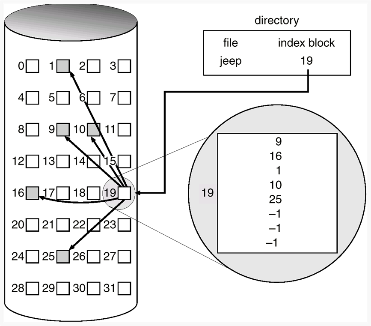
 

**3.索引分配**

**（1）单级索引：**为每个文件分配一个索引块。记录分配给该文件的所有盘块号，文件目录项中记录该索引块的盘块号

**（2）多级索引：**对于大文件，一个块装不下所有索引，需要多个块。索引块之间的关系：①链接；②再创建索引，即索引块的索引—多级索引

**（3）混合方式：**如：BSD UNIX ，在一个文件索引结点中有如下内容：直接地址、一级间接索引、二级间接索引、三级间接索引等等



**四、目录管理**

**目录**是由文件说明（属性）组成的用于文件检索的一个特殊文件。它是一种数据结构，用于标识系统中的所有文件及其物理地址。也称文件系统的元数据文件。目录管理要求实现：**（1）按名存取（用户取的名字）；（2）快速检索：**合理组织目录结构；**（3）文件共享**；**（4）文件重名的解决**

**1.文件控制块和索引结点**

**（1）文件控制块：**FCB是文件存在的标识，FCB记录了系统对于文件进行管理所需要的全部信息。一个文件控制块保存在文件目录中，作为一个**目录项**。包含**基本信息**（文件名、别名数目、物理位置、逻辑结构）；**存取控制信息**（存取权限）；**使用信息**（建立修改时间、打开进程数、修改状态等）

**（2）索引结点（I-node）：**

**1）索引结点的引入：**将FCB拆成两部分：①文件目录部分：文件名字（字符）、索引节点编号（指针）② I节点：记录文件描述信息的部分（除去文件名）的数据结构

**2）磁盘索引结点：**存放在磁盘上的I-node，磁盘上每一个文件都有唯一的I-node

**3）内存索引节点（活动I－node）：**当文件打开后，将磁盘上I-node拷贝到内存，以便使用。除了上述内容，增加了以下内容：

**I-node编号：**内存I-node标识

**状态：**I-node是否上锁或被修改等

**访问计数：**多进程共享

**文件所在的逻辑设备号**

**（3）内存中的文件结构：**在安装文件卷时文件系统中的以下信息从外存缓冲到内存中:

**1）内存分区表：**包含每个已安装的分区的信息；（C盘、D盘）

**2）内存目录项信息：**包括每个最近刚访问过的文件目录项；

**3）系统打开文件表：**包括系统每个打开文件的FCB及其它信息（共享进程计数）；

**4）进程打开文件表：**包括指向系统打开文件表的指针及其它信息（读写指针）。

**2.目录结构**

**（1）单级目录结构：**整个目录组织是一个线性结构，系统中的所有文件都建立在一张目录表中，每个文件占用一个目录项。它主要用于单用户操作系统。（文件不能重名）

**（2）两级目录结构：**在主目录或根目录（第一级目录）下，每个用户对应一个用户文件目录（第二级目录），在用户目录下是该用户的文件，而不再有下级目录。适用于多用户系统，各用户可有自己的专用目录。（文件可重名，检索速度快，不同用户实现文件共享）

**（3）多级目录：**或称为树状目录。文件数目较多时，便于系统和用户将文件分散管理。



**3.目录查询技术**

目录查询-〉FCB(I-node)-〉文件的物理地址（盘块号）-〉文件的物理位置（柱面、磁头号、扇区号）

**（1）线性查询：**以线性表组织文件名及其指向数据块的指针.

**（2）哈希表实现目录：**目录为线性列表，但采用的是哈希数据结构。检索时，哈希表根据文件名得到一个值，并返回一个指向线性列表中元素的指针，降低目录的搜索时间。

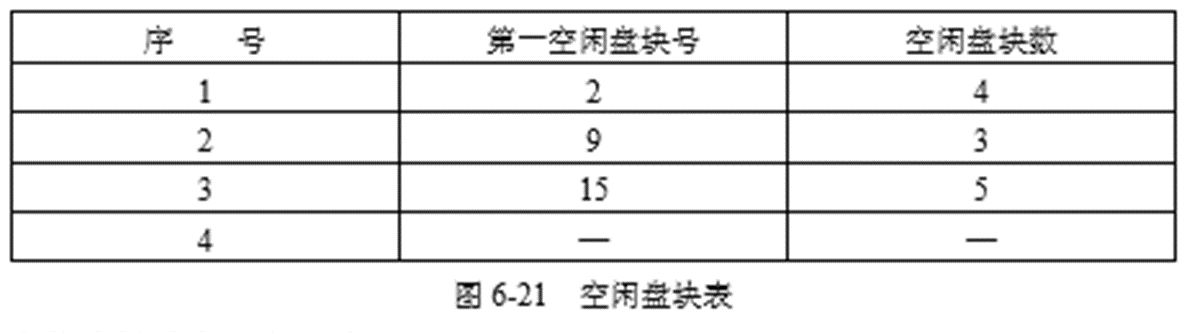
**冲突：**当两个文件哈希到相同位置时使每个哈希条目为链表而不是单个值

**五、文件存储空间的管理(磁盘空间管理）**

外存空闲空间管理的数据结构通常称为**磁盘分配表**(disk allocation table)，分配的基本单位是**簇**。

**1.空闲表法和空闲链表法**

**（1）空闲表法：**



**（2）空闲链表法：**每个空闲簇中有指向下一个空闲簇的指针，所有空闲簇构成一个链表。有两种形式：

**1）空闲簇链接：**以空闲盘块为单位拉成一条链，为文件分配空间时要多次操作。

**2）空闲区链接：**每个盘区除了有指向下一个盘区的指针，还应指明本盘区大小。

**2.位示图法**

**（1）位示图：**每一位表示一个簇，取值0和1分别表示空闲和占用。



**（2）盘块的分配**

1）顺序扫描位示图，找到一个或一组空盘块。

2）转换为对应的盘块号，公式如下：块号为b，第i行，第j列，则b=n\*(i-1)+j（n为每行的位数。）

3）修改位示图。

**（3）盘块的回收：**已知盘块号b，得对应的行列号，之后修改位示图对应位的值。

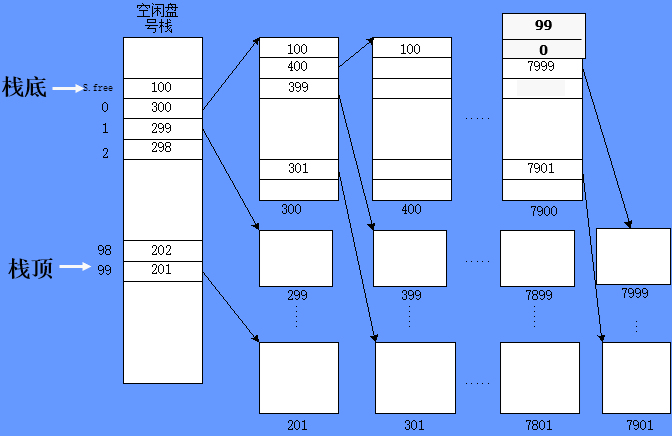
i=(b-1) DIV n + 1，j=(b-1) MOD n + 1

（**注：**上述公式中盘块号和行列号都从1开始。若盘块号和行列号都从0开始，**则**b=n\*i+j、i=b div n、j=b mod n）

**3.成组链法**

UNIX系统采用成组链接法，是空闲表法和空闲链表法的结合

空闲盘块栈用来存放当前可用的一组空闲盘块号。每一组含有的盘块总数和该组所有的空闲盘块号记入前一组的第一个盘块中。

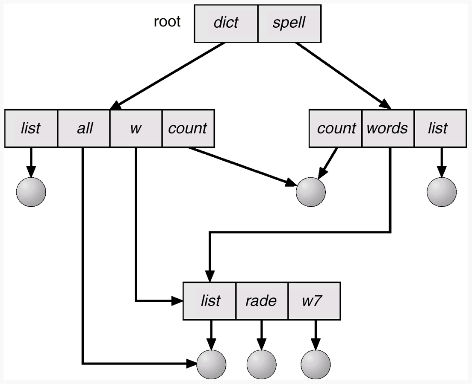


**六、文件共享**

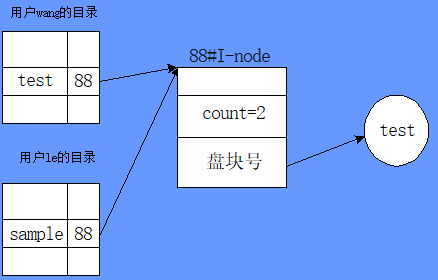
**文件共享**指系统允许多个用户（进程）共享同一个文件。系统中只保留该共享文件的一份副本。文件共享不仅要实现单机系统的共享，多机系统的共享，还要实现网络范围的共享。

**1.基于有向无循环图实现文件共享**

**（1）有向无循环图：**允许一个文件有多个父目录，即多个属于不同用户的目录指向同一个文件，实现多用户共享。此时访问共享文件的路径不唯一。



**（2）利用索引节点实现文件共享：**也称为**硬链接（hard link）**。文件的物理地址及其它属性信息不放在文件目录中，而是放在索引结点中(磁盘索引结点），在目录中设置指向相应索引结点的指针。通过**多个文件名链接(link)到同一个索引结点**，可建立同一个文件的多个彼此平等的别名。别名的数目记录在索引结点的链接计数中，**若其减至0，则文件被删除**。



**2.利用符号链实现文件共享**

由系统创建一个LINK类型的新文件，并给个文件名，其内容是到另一个目录或文件路径的链接，该方法称为符号链接。建立符号链接文件，并不影响原文件，实际上它们各是一个文件。可以建立任意的别名关系，甚至原文件是在其他计算机上。



删除链接并不影响原文件。如果文件条目本身被删除，则释放文件空间，并使链接指针无效。文件主删除文件时不管这些链接，直到使用到该符号链接时，再将其删除。

**3.符号链接（软链接）和索引结点链接（硬链接）**

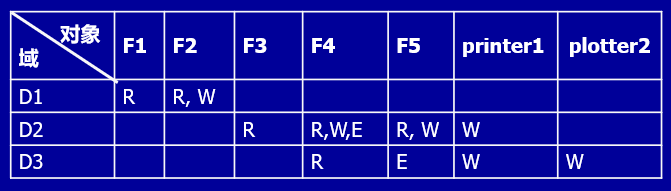
建立符号链接(软链接)时，引用计数值直接复制； 建立硬链接时，引用计数值加1。 删除文件时，删除操作对于符号链接是不可见的，这并不影响文件系统，当以后再通过符号链接访问时，发现文件不存在，直接删除符号链接；

但是对于硬链接则不可以直接删除，引用计数值减1，若值不为0，则不能删除此文件，因为还有其他硬链接指向此文件。

**七、文件保护**

**文件保护**包括两个方面：文件不受物理损坏；避免文件被非法访问

**1.基本的访问矩阵**



**对象：**可以是硬件资源，如磁盘驱动器，打印机；也可以软件资源，如文件，程序。

**域：**是抽象概念，可用各种方式实现，如一个用户是一个域，一个进程也是一个域

**2.访问矩阵的实现**

**（1）访问控制表ACL：**为访问矩阵中**每一列**建立一张ACL。若对象是文件或目录，则ACL中给出每个用户及其所允许的访问类型。当用户请求访问特定文件时，OS检查该文件的ACL，决定是否允许访问。**文件的ACL存放在文件控制块中。**

**（2）访问权限表：**为访问矩阵的**每一行**建立一张表，即访问权限表。说明了一个域（用户、进程）对不同对象的访问权限。



**（3）访问权限表和访问控制表的使用：**

1）仅当访问权限表安全时，由它保护的对象才安全，因此该表不允许用户直接访问，**存放在核心区的一个专用区域中**。

2）目前大多数系统都同时采用访问控制表和访问权限表。为每个对象配置一张访问控制表。若某进程第一次访问某对象时，**先检查访问控制表**，若无权限，则发生异常。若允许访问，则为该**进程建立一张访问权限表，将它连接到该进程**。进程不再需要访问某对象时，则撤销相应的访问权限表。

**附加内容**

1.短作业优先SJF其平均等待时间最小

2.静态分配资源破坏了“循环等待”和“占有并等待”两个条件

3.源程序、可执行文件、库文件、txt文件都是无结构文件。数据库表文件是有结构文件

4.一个程序可以创建多个进程，一个进程同一时刻只可以对应一个程序

5.合理使用P,V操作可以解决一切互斥问题，但不可以解决死锁问题

6.虚拟存储的容量受内存容量、外存容量和寻址位数影响，其中影响最大的是内存容量

（虚拟存储的容量只受指令系统的寻址空间限制）

7.逻辑文件的组织形式是由用户按对信息处理的要求决定的

8.Belady现象：在采用FIFO算法时，有时会出现分配的物理页面数增加，缺页率时而增大、时而减小

9.目态到管态：硬件完成（中断）；管态到目态：修改程序状态字

10.进程调度不需要硬件支持

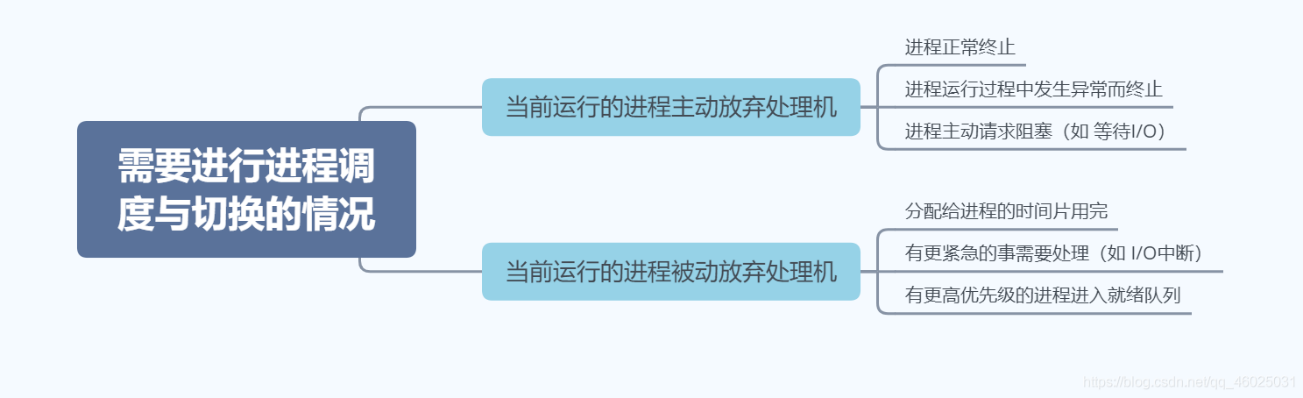
11.页式地址：线性地址；段式地址：二维地址；段页式地址：三维地址

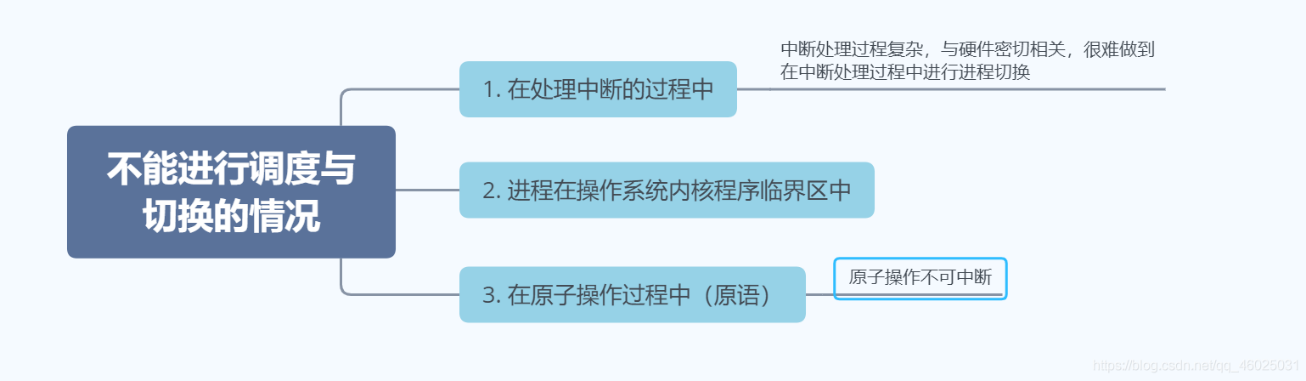
12.引入进程的目的：提高资源的利用率和正确描述程序的执行情况

13.请求分页系统的硬件支持：（1）页表机制；（2）缺页中断机构；（3）地址变换机构

14.请求分页系统的软件支持：（1）请求调页的软件；（2）页面置换的软件；

15.进程切换相关：





进程在操作系统内核程序临界区中不能进行调度与切换。（√）

进程处于临界区时不能进行处理机调度。（×）

16.使用I/0缓冲技术的先决条件是：设备带宽低于CPU带宽

17. read系统调用的参数应该包含文件的名称。

该句话是错的，因为要先open得到一个文件句柄，之后有关文件的系统调用都用这个句柄。

18．容易保留较大空闲分区：首次适应、最佳适应

不容易保留较大空闲分区：最差适应（基本不留下小空闲分区），循环首次适应

19.以打印机为例描述SPOOLing系统的工作过程:

当用户进程请求打印输出时，SPOOLing系统中输出进程，在输出井中为之申请一个空闲磁盘块，将要打印的数据送入其中，输出进程再为用户申请一张空白的用户请求打印表，将用户的打印要求填入其中，再将该表挂到请求打印队列上。若还有进程请求打印，输出进程仍这样处理。当打印机空闲时，打印进程将从打印请求队列的队首取出一张请求打印表，根据表中要求将要打印的数据从输出井传送到内存缓冲区，由打印机打印输出。

20.为什么要有设备驱动程序？用户进程怎样通过设备驱动程序控制设备工作？

（1）为了实现设备独立性，系统设置了设备独立性软件层，在其之下是驱动程序，它是与硬件紧密相关的软件，是I/0进程与设备控制器之间的通信程序，负责接收上层软件命令，屏蔽了具体设备细节。负责管理设备控制器

（2）设备驱动程序将用户进程抽象时I/O请求转换为具体要求，读出和检查设备状态，传送必要的参数，设置设备工作方式，启动I/0设备。

21.什么是进程的顺序性和并发性？

（1）顺序性包括两层含义：内部顺序性：对于一个进程来说，它的所有指令是按顺序执行的；外部顺序性：对于多个进程来说，进程是依次执行的

（2）并发性：多个进程同时存在内存，宏观上并发执行