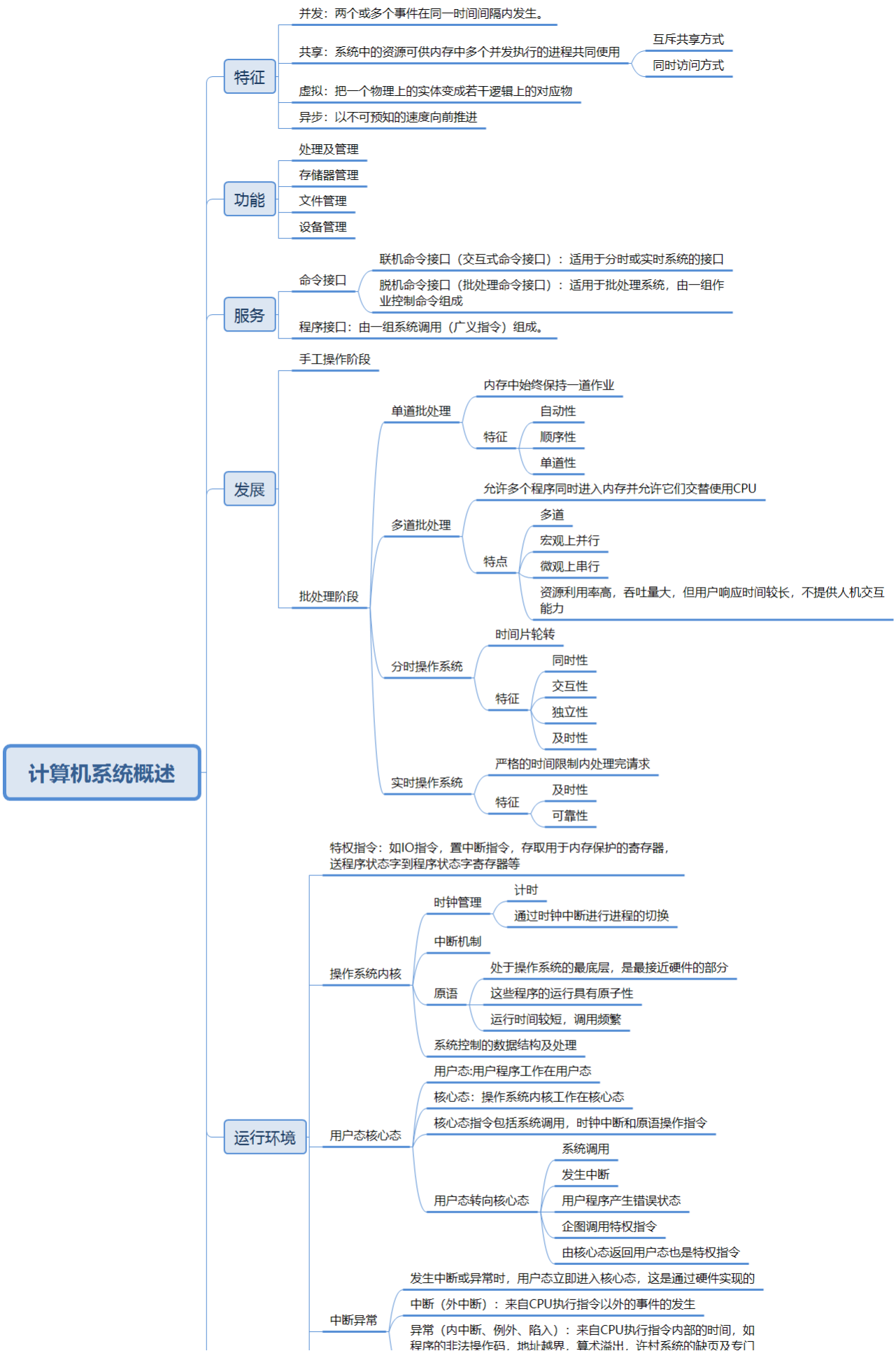
@TOC

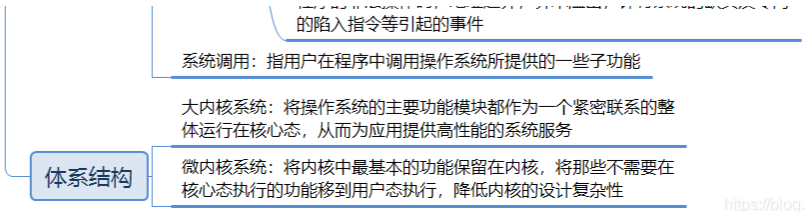
相关课程链接

> 数据结构总结与知识网图 计算机网络知识总结及知识网图 操作系统总结及知识网图 计算机组成原理总结及知识网图

第一章 计算机系统概述

知识网图


计算机系统概述
特征
并发：两个或多个事件在同一时间间隔内发生。
共享：系统中的资源可供内存中多个并发执行的进程共同使用
互斥共享方式
同时访问方式
虚拟：把一个物理上的实体变成若干逻辑上的对应物
异步：以不可预知的速度向前推进
功能
处理及管理
存储器管理
文件管理
设备管理
服务
命令接口
联机命令接口（交互式命令接口）：适用于分时或实时系统的接口
脱机命令接口（批处理命令接口）：适用于批处理系统，由一组作业控制命令组成
程序接口：由一组系统调用（广义指令）组成。
发展
手工操作阶段
批处理阶段
单道批处理
内存中始终保持一道作业
特征
自动性
顺序性
单道性
多道批处理
允许多个程序同时进入内存并允许它们交替使用CPU
特点
多道
宏观上并行
微观上串行
资源利用率高，吞吐量大，但用户响应时间较长，不提供人机交互能力
分时操作系统
时间片轮转
特征
同时性
交互性
独立性
及时性
实时操作系统
严格的时间限制内处理完请求
特征
及时性
可靠性
运行环境
特权指令：如IO指令，置中断指令，存取用于内存保护的寄存器，送程序状态字到程序状态字寄存器等
操作系统内核
时钟管理
计时
通过时钟中断进行进程的切换
中断机制
原语
处于操作系统的最底层，是最接近硬件的部分
这些程序的运行具有原子性
运行时间较短，调用频繁
系统控制的数据结构及处理
用户态核心态
用户态:用户程序工作在用户态
核心态：操作系统内核工作在核心态
核心态指令包括系统调用，时钟中断和原语操作指令
用户态转向核心态
系统调用
发生中断
用户程序产生错误状态
企图调用特权指令
由核心态返回用户态也是特权指令
中断异常
发生中断或异常时，用户态立即进入核心态，这是通过硬件实现的
中断（外中断）：来自CPU执行指令以外的事件的发生
异常（内中断、例外、陷入）：来自CPU执行指令内部的时间，如程序的非法操作码，地址越界，算术溢出，许村系统的缺页及专门



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，合理地组织、调度计算机的工作与资源的分配，进而为用户和其他软件提供方提供方便接口与环境的程序集合。

并发性和并行性的比较

> **并发性**：是指两个或多个事件在同一时刻发生。**并行性**：是指两个或多个事件在同一时间间隔内发生。 在多道程序环境下，并发性是指在一段时间内，宏观上有多个程序同时运行，但在单处理器系统中每个时刻只能有一道程序执行，在微观上这些程序只能分时地交替执行。如果计算机系统中有多个处理器，则这些可以并发执行的程序可被分配到多个处理器上，实现并行执行，即利用每个处理器来处理一个可并发执行的程序。

特权指令和非特权指令

> **特权指令**：是指由特殊权限的指令，若使用不当则会导致系统崩溃。如清内存，分配系统资源，修改虚存的段表或页表，修改用户的访问权限等。为了保证系统安全，特权指令只能用于操作系统或其他系统软件，不能直接提供给用户使用，特权指令必须在核心态执行。 用户态下只能使用**非特权指令**，核心态下可以使用全部指令。在用户态下使用特权指令时，会产生中断以阻止用户使用特权指令。 从用户态转换为核心态的唯一途径是中断或异常。

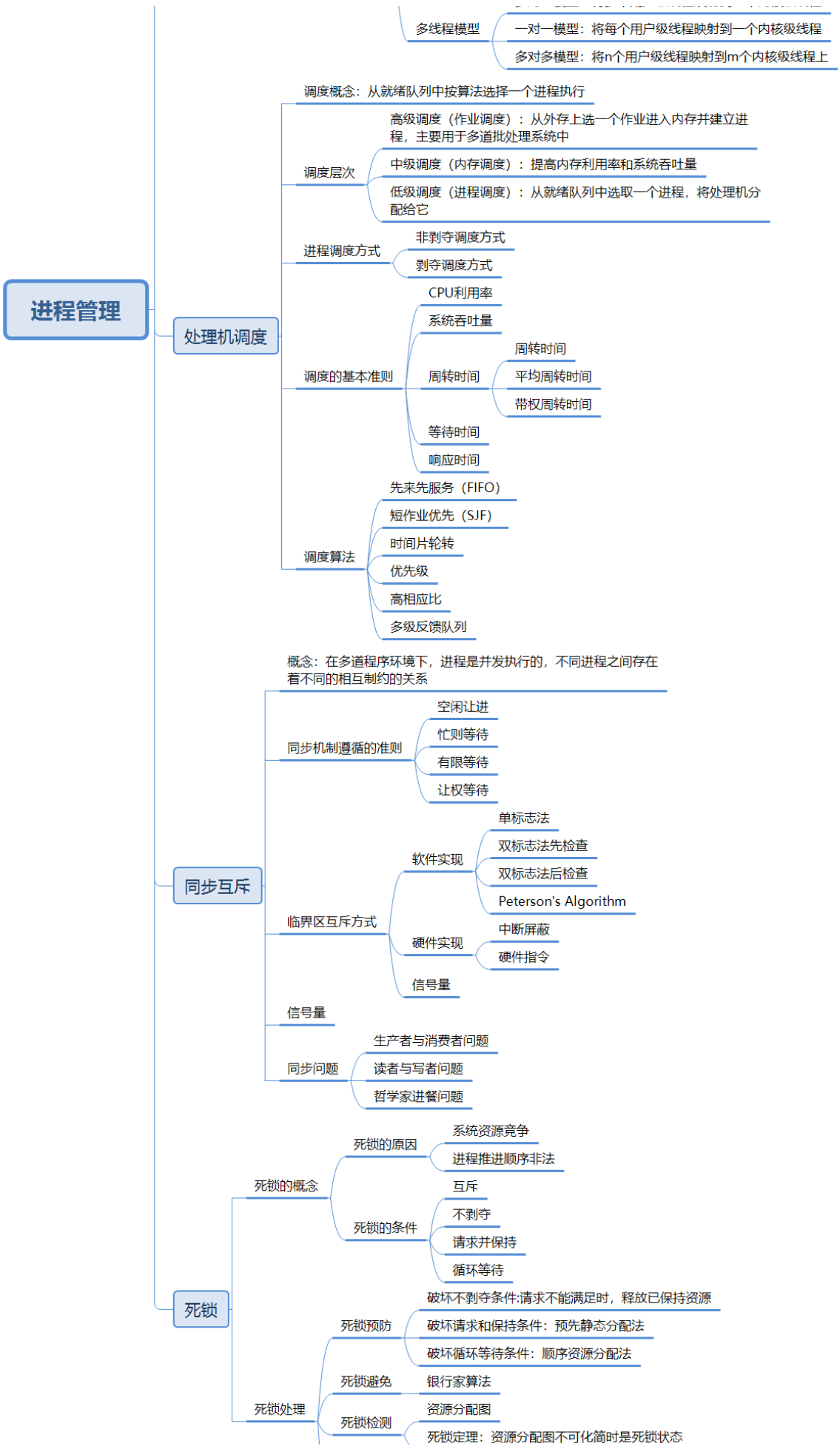
访管指令与访管中断

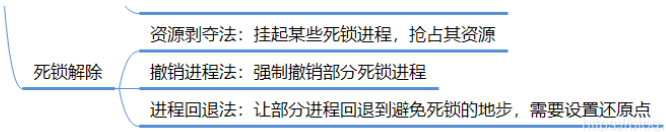
> **访管指令**：是一条可以在用户态下执行的指令。通过使用访管指令，产生一个中断事件（自愿中断），从而将操作系统转换为核心态，称为**访管中断**。 **访管中断**由**访管指令**产生，程序员使用访管指令向操作系统请求服务。

第二章 进程管理

知识网图


进程与线程
进程的概念
在多道程序环境下，引入进程，多道程序并发执行将失去封闭性，具有间断性和不可再现性的特征
组成
程序段
数据段
PCB
特征
动态性：最基本的特征，动态地产生、变化和消亡
并发性：多个进程实体同时存在于内存中
独立性：进程实体是一个能独立运行】独立获得资源和独立接受调度的基本单位
异步性：进程按各自独立的，不可预知的速度向前推进
结构性：每个进程都配置一个PCB对其进行描述
进程的状态及转换
基本状态
运行态
就绪态
阻塞态
创建态
结束态
状态转换
就绪到运行：获得CPU时间片
运行到就绪：时间片用完
运行到阻塞：请求IO或系统调用
阻塞到就绪：IO或中断结束，阻塞进程重新插入就绪队列
进程控制
进程的创建
进程的终止
进程的阻塞和唤醒
进程切换
进程的组织
PCB
是进程存在的唯一标志
信息
进程描述信息
进程控制和管理信息
资源分配清单
处理机相关信息
组织方式
线性方式
链接方式
索引方式
程序段：能被进程调度程序调度到CPU执行的重新代码段
数据段：进程对应的重新加工处理的原始数据，程序执行时产生的中间或最终结果
进程通信
低级通信：PC操作
高级通信
共享存储区：需要同步互斥对共享空间进行控制
信息传递
直接通信方式：直接吧消息发送给接收进程，放入消息缓冲队列上，接收进程从消息缓冲队列中取得消息
间接通信方式：发送进程把消息发送到某个中间实体，接收进程从中间实体取得消息
管道通信
读进程和写进程之间有个管道文件（pipe文件），以字符流的形式读写管道信息
管道机制：互斥，同步，确认对方存在
线程与多线程
引入线程的目的：为了减少程序在并发执行时所付出的时空开销，提高操作系统的并发性能
线程的属性
线程不拥有系统资源，有唯一的标识符和线程控制块
不同的线程可以执行相同的程序
同一进程的各个线程共享该进程所拥有的资源
线程是处理机的独立调度单位
线程的实现方式
用户级线程：有关线程管理的所有工作都由应用程序完成，内核意识不到线程的存在
内核级线程：线程管理的所有工作由内核完成，应用程序没有进行线程管理的代码，只有一个到内核级线程的编程接口
多对一模型：将多个用户级线程映射到一个内核级线程


进程管理
处理机调度
调度概念：从就绪队列中按算法选择一个进程执行
调度层次
高级调度（作业调度）：从外存上选一个作业进入内存并建立进程，主要用于多道批处理系统中
中级调度（内存调度）：提高内存利用率和系统吞吐量
低级调度（进程调度）：从就绪队列中选取一个进程，将处理机分配给它
进程调度方式
非剥夺调度方式
剥夺调度方式
调度的基本准则
CPU利用率
系统吞吐量
周转时间
周转时间
平均周转时间
带权周转时间
等待时间
响应时间
调度算法
先来先服务（FIFO）
短作业优先（SJF）
时间片轮转
优先级
高相应比
多级反馈队列
同步互斥
概念：在多道程序环境下，进程是并发执行的，不同进程之间存在着不同的相互制约的关系
同步机制遵循的准则
空闲让进
忙则等待
有限等待
让权等待
临界区互斥方式
软件实现
单标志法
双标志法先检查
双标志法后检查
Peterson's Algorithm
硬件实现
中断屏蔽
硬件指令
信号量
信号量
同步问题
生产者与消费者问题
读者与写者问题
哲学家进餐问题
死锁
死锁的概念
死锁的原因
系统资源竞争
进程推进顺序非法
死锁的条件
互斥
不剥夺
请求并保持
循环等待
死锁处理
死锁预防
破坏不剥夺条件:请求不能满足时，释放已保持资源
破坏请求和保持条件：预先静态分配法
破坏循环等待条件：顺序资源分配法
死锁避免
银行家算法
死锁检测
资源分配图
死锁定理：资源分配图不可化简时是死锁状态



进程与程序的区别

> 1）进程是暂时的，程序是永久的； 2）进程是动态的，程序是静态的； 3）进程至少由代码、数据和PCB组成，程序仅由代码和数据组成； 4）程序代码经过多次创建可对应不同的进程，而同一系统的进程（或线程）可由系统调度的方法被不同的进程（或线程）多次使用。

进程的定义

> 1）进程是程序的一次执行过程。 2）进程是一个程序及其数据在处理机上顺序执行时所发生的活动。 3）进程是具有独立功能的程序在一个数据集合上运行的过程，是系统进行资源分配和调度的一个独立单位。

进程创建的执行过程

> 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**。 2）为进程**分配资源**，为新进程的**程序**和**数据**及**用户栈**分配必要的内存空间。 3）**初始化PC**B，主要包括初始化**标志信息**、初始化**处理机状态信息**和初始化**处理机控制信息**以及设置**进程的优先级**等。 4）若进程就绪队列能够接纳新进程，则将新进程插入**就绪队列**，等待被调度运行。

进程终止的执行过程

> 1）根据被终止**进程的标识符**，检索**PCB**，从中读出该进程的**状态**。 2）若被终止进程处于**执行状态**，立即**终止**该进程的执行，将**处理机资源**分配给其他进程。 3）若该进程还有子孙进程，则应将其所有子孙进程终止。 4）将该进程所拥有的全部资源，或归还给其父进程，或归还给操作系统。

进程阻塞原语的执行过程

> 1）找到将要被阻塞进程的**标识号**对应的**PCB**。 2）若该进程为**运行态**，则**保护其现**场，将其状态转为**阻塞态**，停止运行。 3）把该PCB**插入**相应事件的等待队列，将**处理机资源调度**给其他就绪进程。

进程唤醒原语执行过程

> 1）在该事件的**等待队列**中找到相应进程的**PCB**。 2）将其从等待队列中移出，并置其状态为**就绪态**。 3）把该PCB插入**就绪队列**，等待调度程序调度。

进程切换的过程

> 1)保存**处理机上下文**，包括程序计数器及其他寄存器。 2）更新**PCB**信息。 3）把进程的PCB移入相应的队列，如就绪、在某事件阻塞等队列。 4）选择另一个进程执行，并更新PCB。 5）更新内存管理的数据结构。 6）恢复处理机上下文。

三级调度的联系

> 1）**作业调度**为进程活动做准备，**进程调度**使进程正常活动，**中级调度**将暂时不能运行的进程挂起，**中级调度**处于**作业调度**和**进程调度**之间。 2）**作业调度**次数少，**中级调度**次数略多，**进程调度**频率最高。 3）**进程调度**是最基本的，不可或缺的。

管程的组成部分

> 1）管程的**名称**； 2）局部于管程内部的**共享结构数据**说明； 3）对该数据结构进行操作的一组**过程（或函数）**； 4）对局部于管程内部的**共享数据**设置**初始值的语句。**

同步与互斥

> **同步**也称直接制约关系，是指为完成某种任务而建立的两个或多个进程，这些进程因为需要在某些位置上协调它们的工作次序而等待、传递信息所产生的制约关系。如管道通信，一个进程写，一个进程读，它们是相互制约的。 **互斥**也称间接制约关系，比如多个进程同时请求打印机（无SPOOLing技术时）。

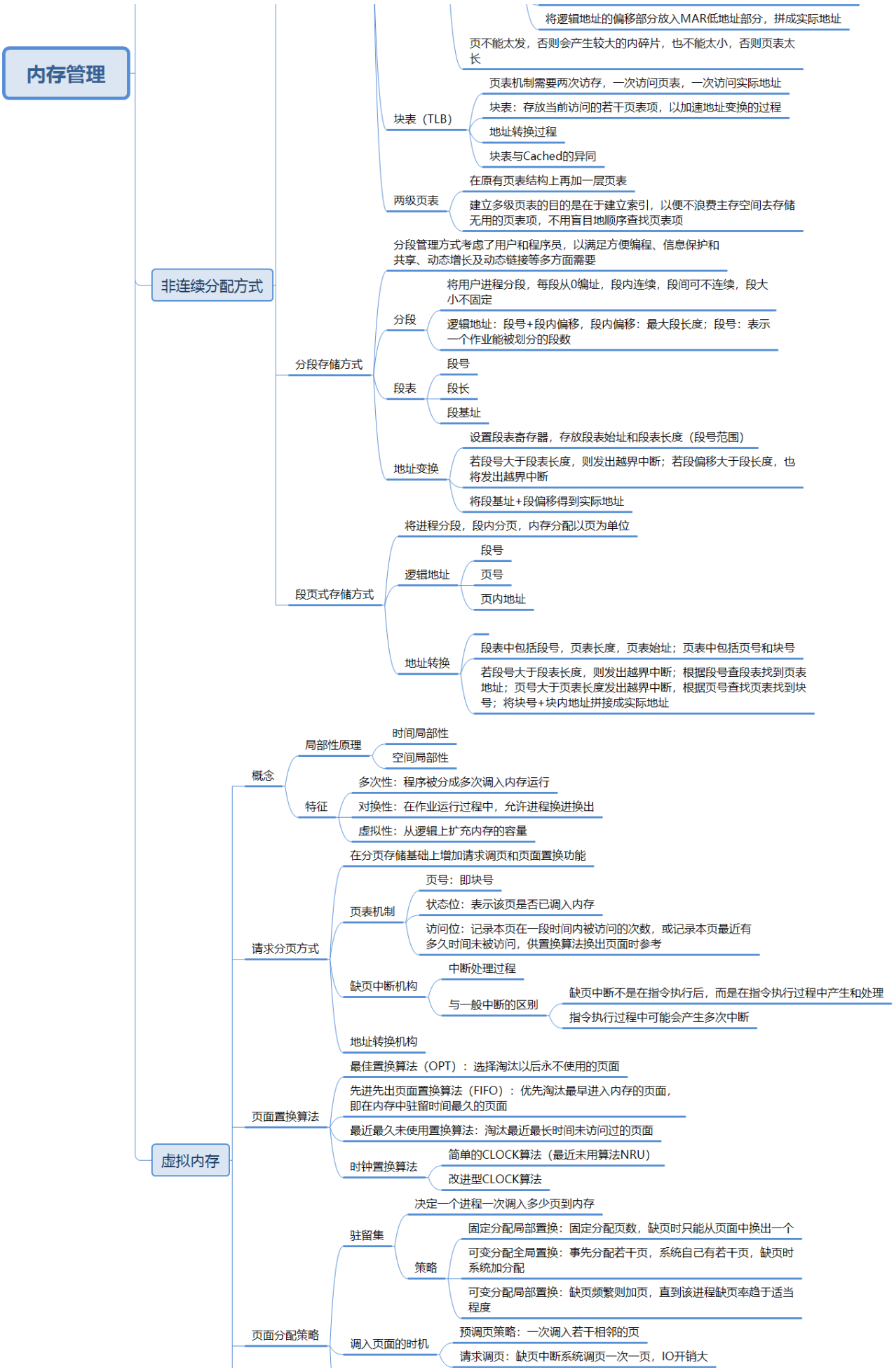
银行家算法的工作原理

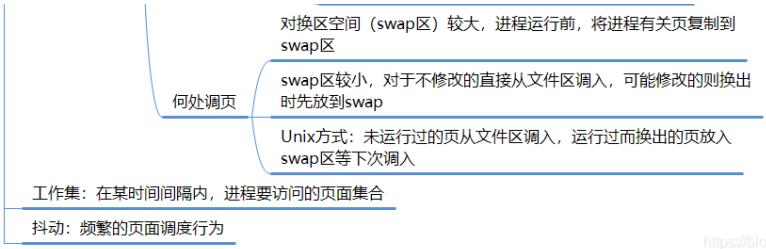
> **银行家算法**的主要思想是避免系统进入**不安全状态。**在每次进行资源分配时，首先检查系统是否有足够的**资源满足要求**，若有则先进行分配，并对分配后的**新状态进行安全性检查。**若新状态安全，则正式分配上述资源，否则拒绝上述资源分配。保证**系统始终处于安全状态**，从而**避免**了死锁现象的发生。

第三章 内存管理

知识网图

- 概念
  - 内存管理功能
    - 内存空间的分配与回收
    - 地址转换
    - 内存空间的扩充
    - 存储保护
  - 程序装入与链接
    - 步骤
      - 编译：由编译程序将用户源代码编译成若干目标模块
      - 链接：由链接程序将编译后形成的一组目标模块及所需的库函数链接在一起，形成一个完整的装入模块
      - 装入：由装入程序将装入模块装入内存运行
    - 链接方式
      - 静态链接：在程序运行前直接链接成一个完整的可执行程序，后面不可再拆分
      - 装入时动态链接：将编译后的目标模块，在装入时边装入边链接
      - 运行时动态链接：程序执行需要该模块时才链接
    - 装入
      - 绝对装入：编译时产生的逻辑地址和实际地址相同，无需重定位，只适用于单道程序环境
      - 可重定位装入（静态重定位）
        - 各目标模块起始地址从0开始，装入时一次重定位到目标模块
        - 必须一次分配全部内存，作业无法在内存移动，也无法继续申请空间
      - 运行时装入（动态重定位）
        - 装入内存时不进行地址转换，运行时才进行地址转换，装入地址都是相对地址
        - 需要重定位寄存器，只需装入部分代码即可运行，可分配到不连续的内存区，可继续申请空间，便于共享
  - 地址空间
    - 逻辑地址（相对地址）
      - 从0号单元开始编址，不同进程有相同的逻辑地址空间，最后映射到内存不同位置
      - 应用程序员只需要知道逻辑地址，系统程序员只会涉及内存管理机制
    - 物理地址
      - 内存中物理单元的集合，是地址转换的最终地址，进程在运行时执行指令和访问数据，最后都要通过物理地址从主存中存取
  - 内存保护
    - 保护操作系统进程，用户进程之间互不影响
    - 通过重定位寄存器和界地址寄存器来保护内存
- 覆盖与交换
  - 多道程序环境下用来扩充内存的两种方式
  - 覆盖：将用户空间划分为固定区域和若干覆盖区域，经常活跃部分放在固定区，将要访问的放在覆盖区
  - 交换
    - 将等待状态的进程换出，将辅存中准备好的进程换入
  - 二者区别
    - 交换是不同进程之间进行，覆盖是同一进程中
    - 覆盖技术已成为历史，交换技术能在使用
- 连续分配方式
  - 单一连续分配
    - 内存中只有一道程序
    - 简单，无外部碎片，但只适用于单用户、单任务的操作系统，有内部碎片，存储器利用率低
  - 固定分区分配
    - 将用户内存分为若干固定大小的区域，每个分区只装入一道作业
  - 动态分区分配
    - 是一种动态划分内存的分区方法
    - 会产生外部碎片，需要紧凑方式来解决
    - 分配策略
      - 首次适应算法（FF）：按顺序递增的次序链接，顺序找到第一个满足的空闲分区
      - 最佳适应算法（BF）按容量递增，找第一个满足的
      - 最坏适应算法（WF）：按容量递减，找第一个满足的
      - 邻近适应算法（NF）：在FF基础上，从上次查找结束位置开始查找
  - 分页存储方式
    - 分页管理方式是从计算机的角度考虑设计的，目的在于提高内存利用率，提升计算机的性能
    - 把主存空间划分为大小相等且固定的块，块作为主存的基本单位
    - 不会产生外部碎片，但最后一块可能会产生内部碎片
    - 基本概念
      - 页面
      - 页面大小
      - 块
      - 逻辑地址结构（页号+页内地址）
      - 页表
    - 地址转换
      - 将逻辑地址转换为内存中的物理地址
      - 需要一个页表寄存器，存放页表实址和页表长度
      - 转换过程
        - 页表若大于页表长度，则发出越界中断
        - 页号*页表项大小+页表实址，找到对应页表项，找到块号放入MAR


内存管理
非连续分配方式
将逻辑地址的偏移部分放入MAR低地址部分，拼成实际地址
页不能太发，否则会产生较大的内碎片，也不能太小，否则页表太长
块表（TLB）
页表机制需要两次访存，一次访问页表，一次访问实际地址
块表：存放当前访问的若干页表项，以加速地址变换的过程
地址转换过程
块表与Cached的异同
两级页表
在原有页表结构上再加一层页表
建立多级页表的目的是在于建立索引，以便不浪费主存空间去存储无用的页表项，不用盲目地顺序查找页表项
分段存储方式
分段管理方式考虑了用户和程序员，以满足方便编程、信息保护和共享、动态增长及动态链接等多方面需要
分段
将用户进程分段，每段从0编址，段内连续，段间可不连续，段大小不固定
逻辑地址：段号+段内偏移，段内偏移：最大段长度；段号：表示一个作业能被划分的段数
段表
段号
段长
段基址
地址变换
设置段表寄存器，存放段表始址和段表长度（段号范围）
若段号大于段表长度，则发出越界中断；若段偏移大于段长度，也将发出越界中断
将段基址+段偏移得到实际地址
段页式存储方式
将进程分段，段内分页，内存分配以页为单位
逻辑地址
段号
页号
页内地址
地址转换
段表中包括段号，页表长度，页表始址；页表中包括页号和块号
若段号大于段表长度，则发出越界中断；根据段号查段表找到页表地址；页号大于页表长度发出越界中断，根据页号查找页表找到块号；将块号+块内地址拼接成实际地址
虚拟内存
概念
局部性原理
时间局部性
空间局部性
特征
多次性：程序被分成多次调入内存运行
对换性：在作业运行过程中，允许进程换进换出
虚拟性：从逻辑上扩充内存的容量
请求分页方式
在分页存储基础上增加请求调页和页面置换功能
页表机制
页号：即块号
状态位：表示该页是否已调入内存
访问位：记录本页在一段时间内被访问的次数，或记录本页最近有多久时间未被访问，供置换算法换出页面时参考
缺页中断机构
中断处理过程
与一般中断的区别
缺页中断不是在指令执行后，而是在指令执行过程中产生和处理
指令执行过程中可能会产生多次中断
地址转换机构
页面置换算法
最佳置换算法（OPT）：选择淘汰以后永不使用的页面
先进先出页面置换算法（FIFO）：优先淘汰最早进入内存的页面，即在内存中驻留时间最久的页面
最近最久未使用置换算法：淘汰最近最长时间未访问过的页面
时钟置换算法
简单的CLOCK算法（最近未用算法NRU）
改进型CLOCK算法
页面分配策略
驻留集
决定一个进程一次调入多少页到内存
策略
固定分配局部置换：固定分配页数，缺页时只能从页面中换出一个
可变分配全局置换：事先分配若干页，系统自己有若干页，缺页时系统加分配
可变分配局部置换：缺页频繁则加页，直到该进程缺页率趋于适当程度
调入页面的时机
预调页策略：一次调入若干相邻的页
请求调页：缺页中断系统调页一次一页，IO开销大



用户源程序变为可在内存中执行的程序所需的步骤

> **编译**：由**编译程序将**用户源代码**编译成若干**目标模块**。 **链接**：由**链接模块**将编译后形成的一组**目标模块**及所需的**库函数**链接在一起，形成一个完整的装入模块。 **装入**：由**装入程序**将装入模块装入**内存**运行。

内存保护的两种方法

> 1）在CPU中设置一对**上下限寄存器**，存放用户作业在主存中的下限和上限地址，每当CPU要访问一个地址时，分别和两个寄存器的值相比，判断有无越界。 2）采用**重定位寄存器（或基址寄存器）**和**界地址寄存器**来实现保护。**重定位寄存器**含最小的物理地址值，**基地址寄存器**含逻辑地址的最大值。每个逻辑地址值必须小于界地址寄存器；内存管理机构动态地将逻辑地址与界地址寄存器进行比较，若未发生地址越界，则加上重定位寄存器的值后映射成为物理地址，再送交内存单元。

地址转换过程（不具有快表）

> 1）计算页号P(P=A/L)与页内偏移量W(W=A%L). 2）比较页号P和页表长度M，若页号P大于页表长度M,则产生越界中断，否则继续执行。 3）页表中页号P对于的页表项地址=页表始址F+页号P*页表项长度，取出该页表内容b，即为物理块号。 4）计算E=b*L+W，得到物理地址E，进行访存。

具有快表的地址转换过程

> 1）CPU给出逻辑地址，根据硬件进行地址转换，将页号送入**高速缓存寄存器**，并将此页号与快表中所有页号进行比较。 2）若找到匹配的页号，说明所要访问的页表在快表中，直接从块表中取出该页对应的页框号，与页内偏移量拼接构成物理地址，则存取数据只需访存一次即可。 3）若未找到与一般中断的匹配的页号，则需要访问主存中的页表，读出页表项后，应同时将其存入快表。若快表已满，则按照一定的算法对旧的页表进行替换。

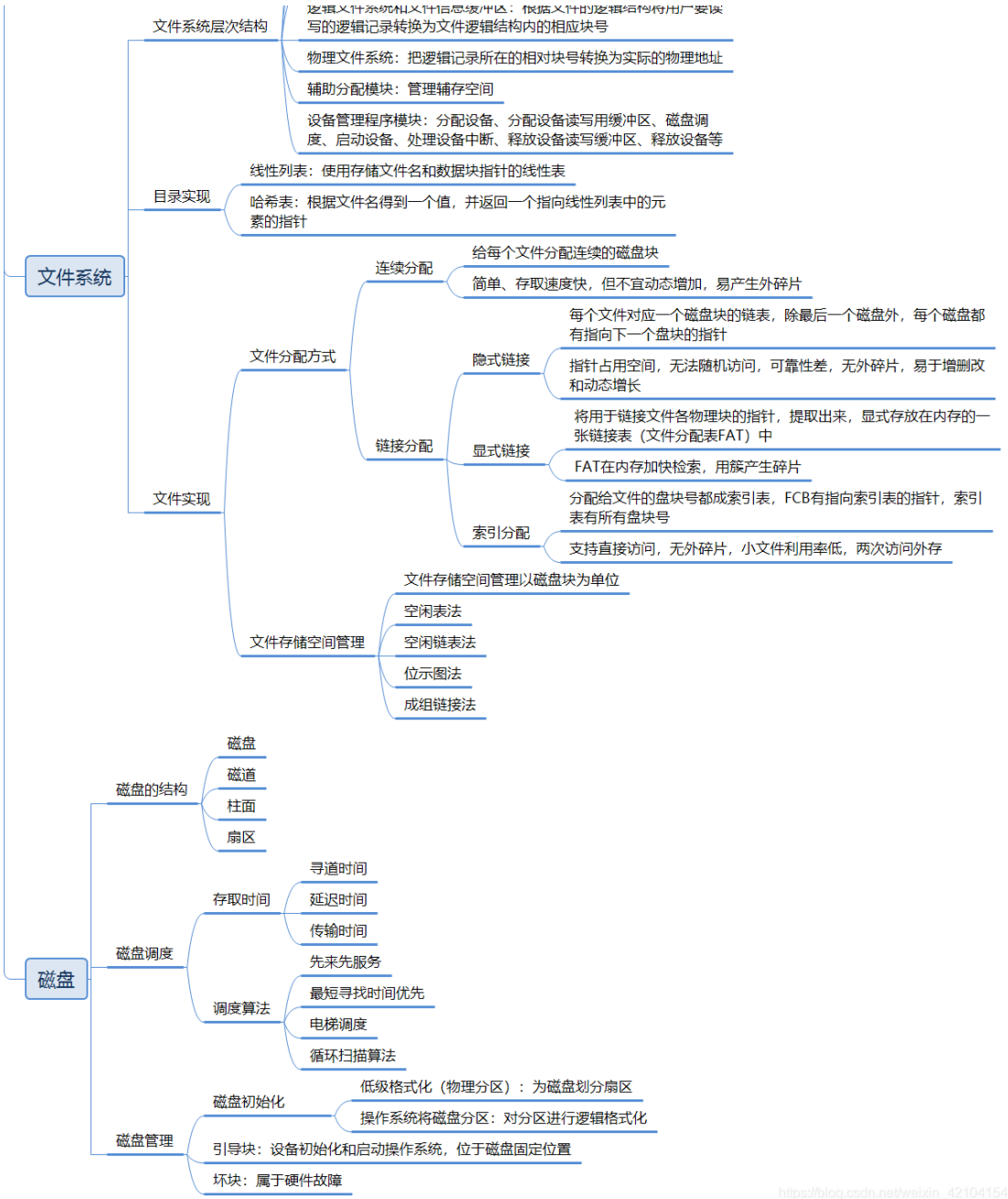
请求分页地址转换过程

> **请求分页系统**中的**地址转换机构**，是在**分页系统**的**地址转换机构**的基础上，为实现虚拟内存，又增加某些功能而形成的。在进行**地址转换**时，先**检索快表**。 若在**快表**中找到要访问的页，则**修改页表项中的访问位**，然后利用页表项中给出的物理块号和页内地址形成物理地址。 若在快表中未找到该页的页表项，则从内存中去查找页表，再对比页表项中的状态位P，查看该页是否已调入内存，若未调入则产生缺页中断，请求从外存把该页调入内存。

第四章 文件管理

知识网图


文件管理
概念
文件：是以计算机硬盘为载体的存储在计算机上的信息集合，用户是以文件为单位进行输入输出
文件结构
数据项：文件系统中最低级的数据组织形式
基本数据项
组合数据项
记录：一组相关的数据项的集合
文件：由创建者所定义的一组相关信息的集合
有结构文件
无结构文件
文件操作
创建、读写、重定位、删除、截断
由系统调用实现
文件打开与关闭
文件逻辑结构
文件的逻辑结构是从用户观点出发看到的文件组织形式，文件的物理结构是从实现观点出发看到的文件在外存上的存储组织形式
无结构文件：将数据按顺序组织成记录并积累、保存，是有序相关信息项的集合
有结构文件
顺序文件
串结构：记录之间的顺序与关键字无关，通常由时间决定
顺序结构：文件中的所有记录按关键字顺序排列
索引文件
定长记录文件：Ai=i*L
不定长记录文件可建立一张索引表加快检索
索引顺序文件
顺序和索引两种组织形式的结合
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，在索引表中为每组中得第一个记录建立一个索引项
直接文件或散列文件：给定记录的键值或通过散列函数转换的键值直接决定记录的物理地址
目录结构
文件目录管理要求
按名存取
快速检索
权限控制
允许重名
FCB：存放控制文件需要的各种信息的数据结构
基本信息
存取控制信息
使用信息
索引节点
FCB包含的信息量过大，将文件目录项简化为文件名和索引节点指针，索引节点指针指向索引节点
索引目录
单级目录
系统只有一个目录表，每个文件对应一个目录项
简单，查找速度慢，不能重名，无法共享，仅适用于单用户环境
二级目录
主文件目录、用户文件目录
提高检索速度，不通透用户的文件可重名；不同用户可访问同一共享文件；用户之间无法合作共享
多级（树形）目录
进程对各文件的访问都是相对于当前目录的
查询速度快，结构清晰，管理有效，但增加访问磁盘次数
无环图目录
为了实现文件共享，在树形目录的基础上增加些指向同一结点的有向边，构成有向无环图
实现文件共享，但更加管理复杂
文件共享
基于索引结点的共享方式（硬链接）：由文件名和索引节点指针构成各用户的文件目录，通过指向同一索引节点达到共享目的
基于符号链接（软链接）的共享方式：共享文件时，只有文件的拥有者才拥有指向其索引结点的指针，而共享该文件的其他用户只有该文件的路径名
文件保护
访问类型
读
写
执行
添加
删除
列表清单
访问控制
访问控制列表：规定每个用户名及其所允许的访问类型
口令：系统创建文件时将口令添加至FCP上，访问时需要提供口令
明码：对文件加密，访问时需要密钥
用户调用接口：为用户提供与文件及目录有关的调用
文件目录系统：管理文件目录
存取控制验证模块：实现文件保护



文件共享方式

> **1）基于索引结点的共享方式（硬链接）** 该共享方式，将文件的**物理地址**及**其他的文件属性**等信息不放入目录项中，而放在**索引结点**中。在文件目录中只设置文件名及指向相应索引结点的指针。在索引结点中有一个链接计数count，用于表示链接到本索引结点上的用户目录项的数目。只有当count为0时，表示没有用户使用该文件，系统将负责删除该文件。 **2）利用符号链实现文件共享（软链接）** 在利用符号链方式实现文件共享时，只有**文件的拥有者**才拥有**指向其索引结点的指针**。而共享该文件的其他用户只有**指向该文件的路径名**，并不拥有指向其索引结点的指针。 当文件拥有者把一个共享文件删除后，其他共享用户通过符号链访问时将访问失败，并将符号链删除。 若当文件拥有者删除共享文件，而在其他共享的用户使用符号链访问该文件之前，又有用户在同一路径下创建另一个具有同一名称的文件，则符号链仍然有效，但访问的文件发生变化，导致错误。 硬链接和软链接都是文件系统中的**静态共享方法**，在文件系统中还存在着**动态共享方法**，即两个进程同时对同一个文件进行操作。

文件分配方式

> 文件分配对应于文件的物理结构，是指如何为文件分配磁盘。常见有以下三种方式： **1）连续分配** 为每个文件分配连续的磁盘块，磁盘地址定义了磁盘上的一个**线性排序**。这种排序使作业访问磁盘时需要的寻道数和寻道时间最小。 特点：实现简单，存取速度快，但文件长度不宜动态增加。 **2）链接分配** 链接分配采取**离散分配**的方式，消除外部碎片，提高磁盘空间的利用率，便于文件的增删改。 **隐式链接**：每个文件对应一个磁盘块的链表；除最后一个盘块外，每个盘块都有指向下一个盘块的**指针**，目录包括文件的第一块的指针和最后一块的指针。 **显式链接**：将用于链接文件各物理快的指针，从每个物理块的块末尾提取出来，显示地存放在内存的一张**链接表**中，即**文件分配表FAT**每个表项中存放对应块的下一块链接指针，即下一个盘块号。 FAT表在系统启动时就会被读入内存，查找FAT的过程是在内存中进行的。 **3）索引分配** 将**每个文件所有的盘块号**集中存放在**索引表**中，每个文件都有其索引块，这是一个磁盘块地址的数组。创建文件时，索引块的所有指针都设为空。 索引分配支持直接访问，且没有外部碎片，但由于索引块的分配，增加了系统存储空间的开销

解决索引块太小无法支持大文件的策略

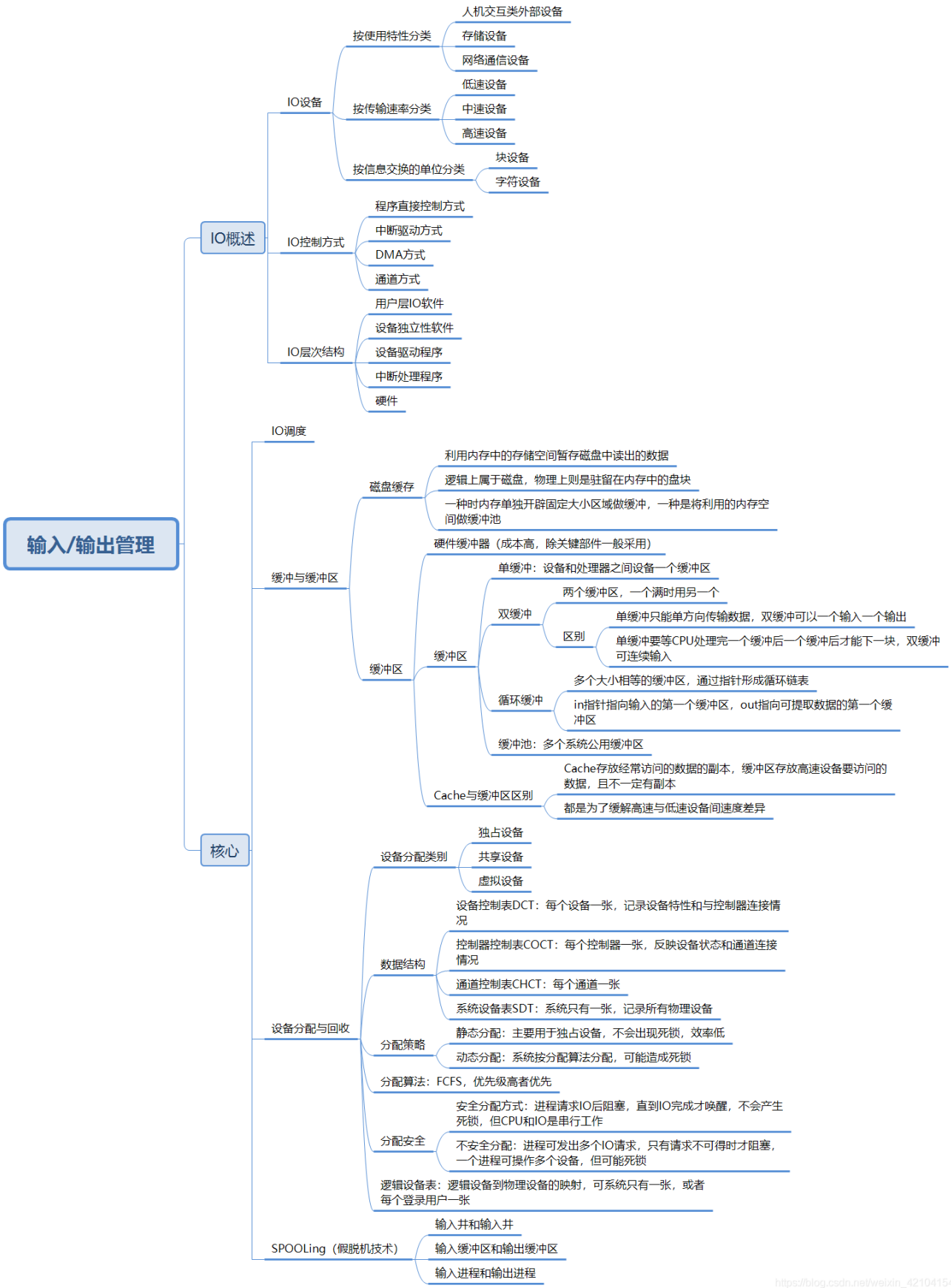
> **1）链接方案** 一个索引块通常为一个磁盘块，因此它本身可以直接被读写。为了处理大文件，可以**将多个索引块链接**起来。 **2）多层索引** 将第一次索引块指向第二层索引块，第二层索引块在指向文件块。这种方法根据最大文件大小，可以继续到第三层或第四层。 **3）混合索引** 将多种索引分配方式相结合的分配方式。

文件存储空间管理

> **1）空闲表法** 属于**连续分配方式**，为每个文件分配一块**连续的内存空间**。系统为外存上的所有空闲区建立一张**空闲盘块表**，每个空闲区对应于一个**空闲表项**，其中包括表象序号、该空闲区第一个盘块号、该区的空闲盘块数等信息，再将所有空闲区按其起始盘块号递增的次序排列。 **2）空闲链表法 空闲盘块链**：将磁盘上的所有**空闲空间**以**盘块**为单位拉成一条链。 **空闲盘区链**：将磁盘上的所有**空闲空间**以**盘区**为单位拉成一条链。 **3）位示图法** 利用**二进制的一位**来表示磁盘中一个**盘块的使用情况**，磁盘上所有的盘块都有一个二进制位与之对应。0表示对应的盘块空闲，1表示对应的盘块已分配。 **4）成组链接法** 将顺序的n个空闲扇区地址保存在第一个空闲扇区内，其后一个空闲扇区内则保存另一个顺序空闲扇区的地址，如此连续，直至所有空闲扇区均予以链接。系统只需保存一个指向第一个空闲扇区的指针。

第五章 输入输出管理

知识网图



程序直接控制方式

> 计算机从外部设备读取数据到存储器，每次读一个字的数据。对读入的每个字，CPU需要对外设状态进行循环检查，直到确定该字已经在I/O控制器的数据寄存器中。在程序直接控制方式中，由于CPU的高速性和I/O设备的低速性，导致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，造成了CPU资源的极大浪费。 CPU之所以要不断测试I/O设备的状态，就是因为在CPU中未采用中断机构，使I/O设备无法向CPU报告它已完成了一个字符的输入操作。

中断驱动方式

> 允许I/O设备主动打断CPU的运行并请求服务，从而“解放”CPU，使得其向I/O控制器发送读命令后可以继续做其他有用的工作。 中断驱动方式比程序直接控制方式有效，但由于数据中的每个字在I/O控制器之间的传输必须经过CPU，导致中断驱动方式仍然会消耗较多的CPU事件。

DMA方式

> 在I/O设备和内存之间开辟直接的数据交换通路。基本单位是数据块，所传送的数据，是从设备直接的送入内存的，或者相反。仅在传送一个或多个数据块的开始和结束时，才需CPU干预，整个数据的传送是在DMA控制器的控制下完成的。

通道控制方式

> I/O通道是指专门负责输入/输出的处理机。I/O通道方式是DMA方式的发展，可以进一步减少CPU的干预，即把一对数据块的读（写）为单位的干预，减少为对一组数据块的读（或写）及有关控制和管理为单位的干预。同时，又可以实现CPU、通道、I/O设备三者的并行操作，更有效地提高整个系统的资源利用率。

I/O子系统的层次结构

> **1)用户层I/O软件** 实现与用户交互的接口，用户可直接调用在用户层提供的、与I/O操作有关的库函数，对设备进行操作。 **2）设备独立性软件** 用于实现用户程序与设备驱动器的统一接口、设备命令、设备保护及设备分配与释放等，同时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** 与硬件直接有关，负责具体实现系统对设备发出的操作指令，驱动I/O设备工作的驱动程序 **4）中断处理程序** 用于保存被中断进程的CPU环境，转入相应的中断处理程序进行处理，处理完并恢复被中断进程的现场后，返回被中断进程。 **5）硬件设备**

设备控制器（适配器）的主要功能：

> 1）接受和识别CPU或通道发来的命令，如磁盘控制器能接收读写查找等命令 2）实现数据交换，包括设备和控制器之间的数据传输；通过数据总线或通道，控制器和主存之间的数据传输 3）发现和记录设备及自身的状态信息，供CPU处理使用 4）设备地址识别

引入缓冲区的主要目的

> 1)缓和CPU与I/O设备之间的速度不匹配。 2）减少对CPU的中断频率，放宽对CPU中断响应时间的限制。 3）解决基本数据单元大小（即数据粒度）不匹配问题。 4）提高CPU和I/O设备之间的并行性 **其采用方法如下：** 1）采用硬件缓冲器，但成本高 2）采用缓冲区（位于内存区域）

知识网图

链接: https://download.csdn.net/download/weixin_42104154/16486701.