# 计算机操作系统

* 考试时间：7.4
* 第一讲
  + 什么叫操作系统
    - 计算机操作系统是指控制和管理计算机的软、硬件资源，合理组织计算机的工作流程，方便用户使用的程序集合。
  + 操作系统的三个作用 管理者 ……虚拟机
    - 计算机系统软硬件资源的管理者。
    - 为用户提供一台等价的扩展机器或虚拟机。
    - 最重要、最基本、最复杂的系统程序，控制应用程序执行的程序。
  + 操作系统的发展历史 每一代思想 特别是分时系统 （现代的都是分时）定义特点优缺点
    - 第一代：手工操作
      * 1945-1955
      * 使用机器语言
      * 无操作系统
      * 用于数学计算
      * 输入输出：插件版、纸带、卡片
      * 计算机处理能力日益提升，而手工操作效率低下，造成了资源浪费。
    - 第二代：单批道处理系统
      * 1955-1965
      * 用于大型机
      * 使用汇编语言，FORTRAN，作业
      * FMS（Fortran Monitor System），IBSYS（IBM为7094机配备的操作系统）
      * 用于较复杂的科学工程计算
        + 联机批处理
        + 脱机批处理
      * 机时在走来走去中浪费掉
      * 优点：同一批作业自动依次更替，改善了主机CPU和I/O设备的使用效率，提高了吞吐量。
      * 主要问题：CPU和I/O设备使用忙闲不均，取决于作业特性。
        + 计算为主的作业，外设空闲；
        + I/O为主的作业，CPU空闲。
    - 第三代：多批道处理系统
      * 1965-1980
      * 使用集成电路
      * 操作系统：庞大、复杂
      * 多道：内存中同时存放几个作业。
      * 几项新技术：Multiprogramming，Spooling
      * 优点：
        + 资源利用率高（CPU、内存、I/O）
        + 作业吞吐量大
      * 缺点：
        + 用户交互性差
        + 作业平均周转时间长
    - 第四代：分时系统
      * 70年代中期至今
      * 多个用户分享使用同一台计算机。多个程序分时共享硬件和软件资源。
      * 通常按时间片分配：各个程序在CPU上执行的轮换时间。
      * 操作系统：CTSS（M.I.T.）、Multics（computer community）
      * 特征：
        + 同时性

也称多路性。若干用户同时与一台计算机相连，宏观上看各个用户在同时使用计算机，他们是并行的；微观上看各个用户在轮流使用计算机。

* + - * + 交互性

用户通过终端设备（如键盘、鼠标）向系统发出请求，并根据系统的响应结果再向系统发出请求，直至得到满意的结果。

* + - * + 独立性

每个用户使用各自的终端与系统交互，彼此独立、互不干扰

* + - * + 及时性

指用户向系统发出请求后，应该在较短的时间内得到响应。

* + - 新发展：个人计算机、实时系统、网络与分布式系统、移动计算……
  + 什么叫中断
    - 中断：指CPU在收到外部中断信号后，停止原来工作，转去处理该中断事件，完毕后回到原来断点继续工作。
    - 通道：用于控制I/O设备与内存间的数据传输。启动后可独立与CPU运行，实现CPU与I/O的并行。
  + 中断的处理机制
* 第二讲
  + 操作系统的五大功能点 大点重要
    - 处理机管理
      * 进程控制
      * 进程调度
      * 进程同步
      * 进程通信
    - 存储管理
      * 内存分配
      * 内存保护
      * 地址映射
      * 内存扩充
    - 设备管理
      * 缓冲管理
      * 设备分配
      * 设备处理
    - 文件管理
      * 文件存储空间的管理
      * 目录管理
      * 文件的读/写管理和保护
    - 用户接口
      * 命令接口
      * 程序接口
      * 图形接口
  + 操作系统的四个特征 并发共享虚拟异步 细点要知道
    - 并发（concurrency）
      * 两个或多个事件在同一时间间隔内发生
      * 并行：两个或多个事件在同一时刻发生
      * 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）
    - 共享（sharing）
      * 系统中的资源可以供内存中多个并发执行的进程共同使用。
        + 互斥共享方式，临界资源。
        + 同时访问方式。
      * 并发和共享是操作系统的两个最基本的特征，他们互为存在的条件。
    - 虚拟（virtual）
      * 通过某种技术把一个物理实体变为若干个逻辑上的对应物。
      * OS中利用多种虚拟技术
        + 虚拟处理机技术，多道程序并发执行时分时使用一台处理机。
        + 虚拟存储器技术，将一台机器的物理存储器变为虚拟存储器，以便从逻辑上来扩充存储器的容量。
        + 虚拟设备技术，将一台物理I/O设备虚拟为多台逻辑上的I/O设备，并允许每个用户占用一台逻辑上的I/O设备。
        + 把一条物理信道通过分时使用的方法虚拟为多条逻辑信道。
    - 异步（asynchronism）
      * 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：分时系统中，多个进程并发执行，“时走时停”，不可预知每个进程的运行推进快慢。
      * 判据：无论快慢，应该结果相同——通过进程互斥和同步手段来保证。
      * 难以重现系统在某个时刻的状态（包括重现运行中的错误）
  + 每种操作系统特点优缺点 SPOOLING名词解释 偶尔简答
    - 批处理操作系统（多道批处理）
      * SPOOLING技术
        + 实现批处理的主要输入输出手段是Simultaneous Peripheral Operation On-Line，同时外围设备联机操作-假脱机技术。
        + 思想：利用磁盘做缓冲，将输入、计算、输出分别组织成独立的任务流，使I/O和计算真正并行。
        + 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

* + - * 优点
        + 作业流程自动化：资源利用率高
        + 吞吐量大：单位时间内完成的工作总量大
      * 缺点
        + 用户交互性差，调试程序困难

无交互手段：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

* + - 分时操作系统
      * 分时：把计算机的系统资源（尤其是CPU时间）进行时间上的分割，每个时间段称为一个时间片（time slice），每个用户依次轮流使用时间片。
      * 工作方式
        + 一台主机连接了若干个终端
        + 每个终端有一个用户使用
        + 交互式地向系统提出命令请求
        + 系统接受每个用户的命令
        + 用时间片轮转方式处理服务器的请求
        + 通过交互式方式在终端上显示结果
        + 用户根据上步结果发出下道命令
      * 特征
        + 多路性

多个用户同时工作，也称同步性。

* + - * + 独立性

各用户独立操作，互不干扰，感觉不到计算机为其他用户服务。

* + - * + 及时性

系统能及时对用户的操作进行响应。

* + - * + 交互性

分时系统的基本属性。

* + - * + 实现条件

终端设备

轮转算法

会话语言

独占资源，“先进先出”方法

* + - 实时操作系统
      * 能够在制定或者确定的时间内完成系统功能和对外部或内部、同步或异步时间做出相应的系统。
    - 网络操作系统
      * 专用系统
        + 许多实时系统是专用系统，而批处理与分时系统通常是通用系统。
      * 实时控制
        + 实时系统用于控制实时过程，要求对外部事件的迅速响应，具有较强的中断处理机构。
      * 高可靠性
        + 实时系统用于控制重要过程，要求高度可靠，具有较高冗余，如双机系统。
      * 事件驱动和队列驱动
        + 实时系统的工作方式：接受外部消息、分析消息、调用相应处理程序进行处理。
      * 可与通用系统结合成通用实时系统：实时处理前台作业，批处理为后台作业。
    - 网络操作系统
      * 是在通常操作系统功能的基础上提供网络通信和网络服务功能的操作系统。
      * 目标：互相通信，资源共享
      * 功能
        + 网络通信功能

通过网络协议进行高效、可靠的数据传输

* + - * + 网络资源管理

协调各用户使用

* + - * + 网络服务

文件盒设备共享，信息发布

* + - * + 网络管理

安全管理、故障管理、性能管理等

* + - * + 互操作

直接控制对方比交换数据更为困难

* + - 分布式操作系统
      * 以计算机网络为基础，处理和控制的分散（相对于集中式系统），基本特征是处理上的分布，即功能和任务的分布。
      * 分布式操作系统的所有任务可在系统中任何处理机上运行，自动实现全系统范围内的任务分配并自动调度各处理机的工作负载。
      * 透明性
        + 资源共享，分布对用户来讲是不知道的
      * 自治性
        + 处于分布式系统的多个主机处于平等地位，无主从关系。
      * 优点：处理能力增强、速度更快、可靠性增强
    - 多处理机操作系统
      * 为了提高计算机系统性能和可靠性
        + 增加系统的吞吐量：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）
        + 提高系统的可靠性：故障时系统降级运行
      * 类型
        + 紧密耦合：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。
        + 松散耦合：各处理机带有各自的存储器、I/O设备和操作系统，通过通道或通信线路相连。每个处理机上独立运行OS
    - 嵌入式操作系统
      * 以应用为中心、以计算机技术为基础、软件硬件可裁剪、适应应用系统，对功能、可靠性、成本、体积、功耗严格要求的专用计算机系统。
      * 具有某些实时系统的特征
      * 但有限制条件：大小、内存、能源
  + 什么叫响应时间
    - 从终端发出命令到系统处理完这条命令给予回答所需的时间
    - 影响因素
      * 机器处理能力
      * 系统中连接的终端数目
      * 服务请求的分布
      * 调度算法（时间片的选取）
  + 硬实时软实时
    - 硬实时
      * 必须在规定的时刻或时间范围完成任务。
    - 软实时
      * 接受偶尔违反最终时限的情况。
  + 处理器 非对称 对称
    - 非对称式多处理
      * 又称主从模式
      * 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。
    - 对称式多处理
      * OS交替在各个处理器上执行，任务负载较为平均，性能调节容易——傻瓜式。
  + 操作系统结构 强内核 弱内核
    - 强内核
      * 基于传统的集中式操作系统的内核结构，系统调用是通过程序陷入内核实现，内核完成相应的服务后返回应用程序，同时返回结果给用户。
    - 微内核
      * 基本思想
        + 良好的结构化、模块化，最小的公共服务
      * 设计目标
        + 使内核尽可能小、功能尽可能少（基本），把其他所有功能放到核外的用户级来完成。
      * 提供以下基本服务
        + （有限的）进程管理和调度
        + 进程间的通信机制
        + （某些）存储管理
        + 低级I/O操作
* 第三讲
  + 作业的概念 作业流 作业控制块 作业的四个状态 转换关系 后？……运行 退出
    - 作业
      * 用户在一次计算过程中或一次事务处理过程中，要求计算机系统所做工作的总称。
      * 由程序、数据、作业说明书三部分组成。
    - 作业步
      * 一个作业可划分成若干部分，处理作业的各个独立的子任务被称为一个作业步。
    - 作业流
      * 把若干作业一次放在输入设备上，在操作系统控制下一次运行。
    - 作业处理过程



* + - 作业控制块（JCB）
      * 用来对作业进行描述的数据结构称为作业控制块
      * 作业控制块是批处理作业存在的标志
      * 保存系统对于作业进行管理所需要的全部信息
      * 位于磁盘区域中，和作业一一对应
      * 内容
        + 信息数量及内容因系统而异
      * 生命周期
        + 建立
        + 使用
        + 撤销
    - 批处理作业的状态及转换
      * 进入状态
      * 后备状态
      * 运行状态
      * 退出状态





* + 程序状态类型 管态算态 访管 特权 用户指令
    - 算态（用户态）
      * 用户工作的状态
    - 管态（系统态）
      * 系统程序工作的状态
    - 特权指令
      * 只允许管态下使用的指令
    - 用户程序在算态下运行——只能使用算态指令
    - 操作系统是系统程序在管态下运行——可以使用算态指令，也可以使用特权指令
    - 访管指令
      * 引起访管中断，实现从算态到管态的转变（设定程序状态字为管态），在管态下有操作系统代替用户完成操作，再由管态返回算态。
  + 系统调用 和一般传统调用的区别 简答
    - 系统调用
      * 至少包括一条访管指令在内的、有操作系统代替用户完成的某些特定功能的程序段。
        + 将系统转入管态，只能由汇编语言直接访问
        + 是操作系统提供给编程人员唯一的接口
    - 功能
      * 完成与硬件相关的工作以及控制程序的执行
        + 设备管理
        + 文件控制
        + 进程控制
        + 进程通信
        + 存储管理
        + 系统管理



* + - 与普通过程调用比较
      * 相同点
        + 改变指令流程
        + 重复执行和公用
        + 改变指令流程后需要返回原处
      * 不同点
        + 系统调用

程序中不包含被调用代码，用户程序长度缩短；当OS升级时，调用方不必改变

调用地址和返回地址都是固定的，系统调用指令中不包含调用地址，只包含功能号

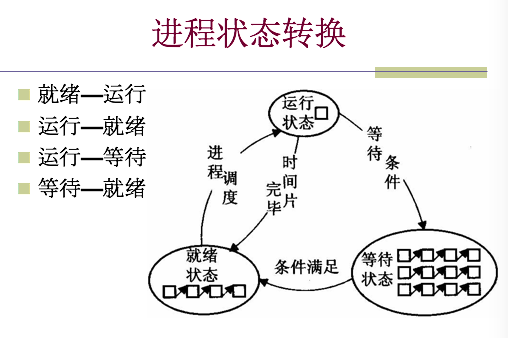
* + - * + call过程调用方式

被调用代码与调用代码在同一程序之内

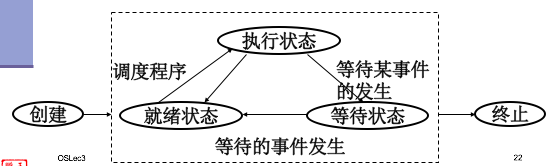
调用地址是固定的，包含在调用语句中；返回地址是不固定的

|  |  |  |
| --- | --- | --- |
| * **两者区别** | * **system calls** | * **call** |
| * **调用方式** | * **动态调用** | * **静态调用** |
| * **运行状态** | * **不同系统状态** | * **相同系统状态** |
| * **进入方式** | * **利用int、trap指令** * **进行系统调用** | * **利用call、jmp指令** * **进入普通过程调用** |

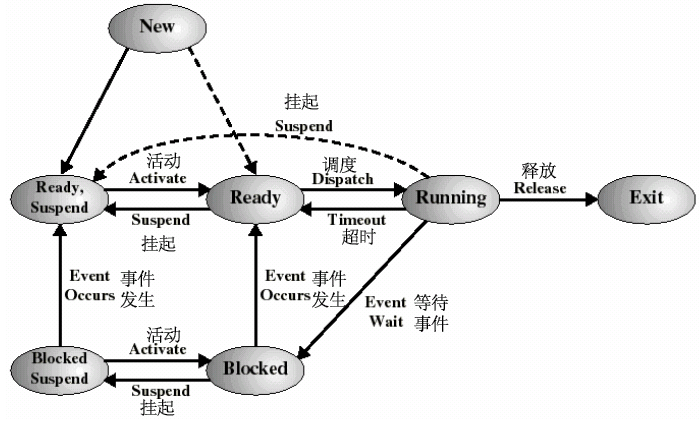
* 第四讲
  + 什么叫进程 三个特点 （封闭性） 基本概念
    - 进程
      * 程序在处理机上执行时所发生的活动
      * 是一个容器，该容器用以聚集相关资源
      * 是具有独立功能的程序关于某个数据集合上的一次运行活动，是系统进行资源分配和调度的独立单位
    - 进程的特征
      * 动态性
        + 进程是程序的一次执行，有着“创建”、“活动”、“撤销”、等过程，具有一定的生命周期，是动态地产生、变化和消亡的。
      * 并发性
        + 进程之间的动作在时间上可以重叠，即系统中有若干进程都已经“开始”但有没有“结果”，称这些进程为并发进程。
      * 独立性
        + 进程是系统调度和资源分配的独立单位，它具有相对独立的功能，拥有自己独立的进程控制快PCB。
      * 异步性
        + 各个并发进程按照各自独立的、不可预知的速度向前推进。
      * 交互性
        + 并发进程之间具有直接或间接的关系，在运行过程中需要进行必要的交互（同步、互斥和数据通信等），已完成特定任务。
  + PCB 状态 唯一标识
    - 系统为了管理进程设置的一个专门的数据结构，用来记录进程的外部特征，描述进程的变化过程。
    - 进程的组成：程序+数据+PCB
    - PCB是系统感知进程存在的唯一标志，进程与PCB是一一对应的。
    - PCB内容
      * 进程描述信息
        + 进程标识符（process ID），唯一，通常是一个整数
        + 进程名，通常基于可执行文件名（不唯一）
        + 用户标识符（user ID）；进程组关系
      * 进程控制信息
        + 当前状态
        + 优先级（priority）
        + 代码执行入口地址
        + 程序的外存地址
        + 运行统计信息（执行时间、页面调度）
        + 进程间同步和通信信息
      * 所拥有的资源和使用情况
        + 内存、外存、I/O设备等
        + 打开文件列表
      * CPU现场保护结构
        + 各寄存器值
  + 进程状态转换 含义 迁移关系 三状态五状态七状态，闭着眼睛画出来
    - 三状态
      * 运行状态（running）
        + 进程占有CPU，并在CPU上运行
      * 就绪状态（ready）
        + 一个进程已经具备运行条件，但由于无CPU暂时不能运行的状态（当调度给其CPU时，立即可以运行）
      * 等待状态（blocked）
        + 也称阻塞状态，阻塞态、封锁态、睡眠态
        + 指进程因等待某种事件的发生而暂时不能运行的状态（即使CPU空闲，该进程也不可运行）



* + - 五状态
      * 创建状态（new）
        + 进程刚创建，但还不能运行，如分配和建立PCB表项、建立资源表格并分配资源，加载程序并建立地址空间表。
      * 结束状态（exit）
        + 进程已结束运行，回收除PCB之外的其他资源，并让其他进程从PCB中收集有关信息。



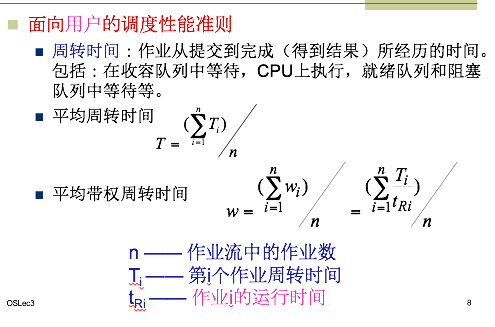
* + - 七状态
      * 挂起状态（suspend）
        + 把一个进程从内存转到外存
      * 激活状态（activate）
        + 把一个进程从外存转到内存

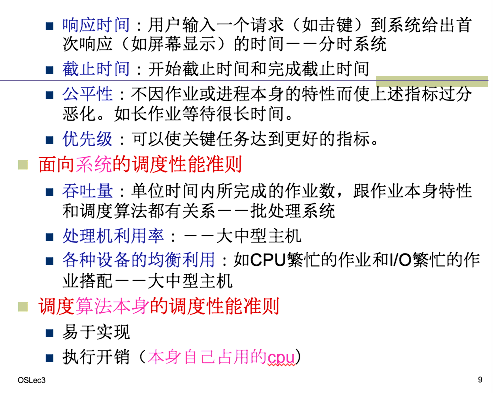


* 第五讲
  + 处理机调度 高级中级低级调度 区别

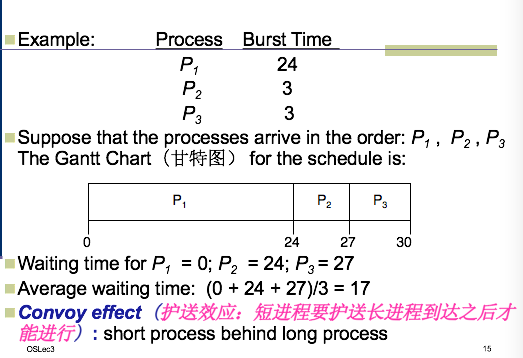


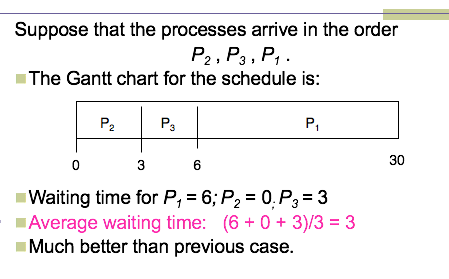
* + 评价指标 优先级 响应时间……基本定义



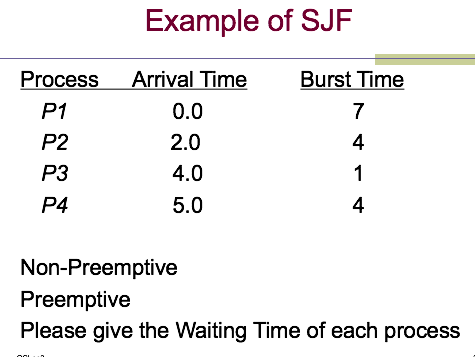


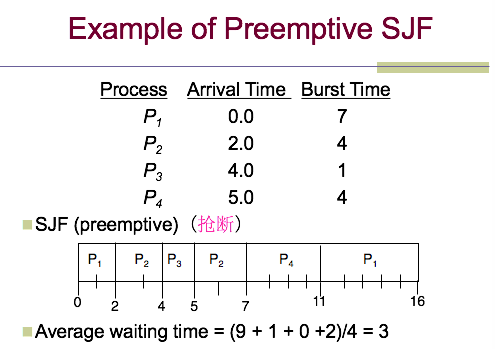
* + 四个调度算法 先来先服务、短作业常考 优点缺点 适用于哪一种作业 最高响应比 先来先服务 短作业 要会计算 给进入时间 估计运行时间 算平均周转时间和在线周转时间？？？其他把思想搞清楚
    - 先来先服务（FCFS）
      * 特点
        + 比较有利于长作业，不利于短作业
        + 有利于CPU繁忙的作业（长时间占用CPU），而不利于I/O繁忙的作业。
        + 最简单的算法

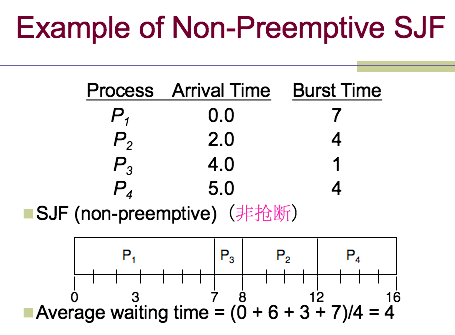




* + - 短作业优先（SJF）
      * 对预计执行时间短 的作业（进程）优先分派处理机。通常后来的短作业不抢先正在执行的作业。
      * 平均周转时间最小
      * 优点
        + 比FCFS改善平均周转时间和平均带权周转时间
        + 提高系统的吞吐量
      * 缺点
        + 对长作业非常不利，可能长时间得不到执行（饥饿现象）
        + 未能依据作业的紧迫程度来划分执行的优先级
        + 难以准确估计作业（进程）的执行时间，影响调度性能



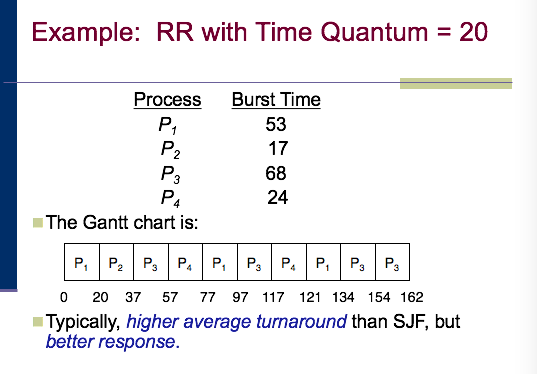


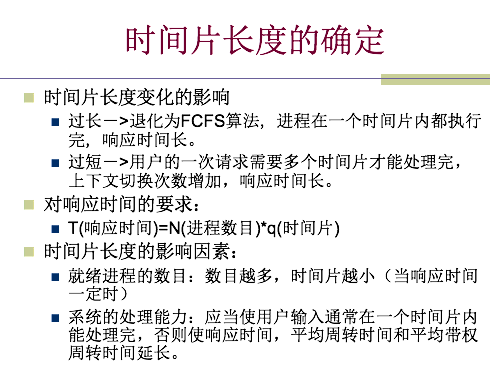




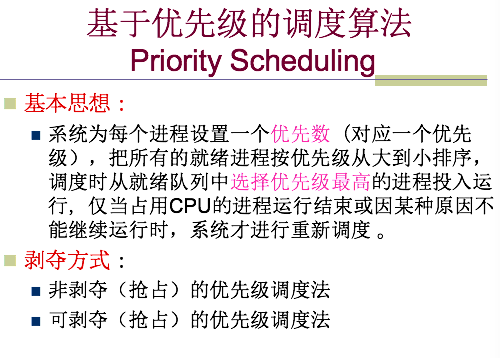
* + - 时间片轮转算法

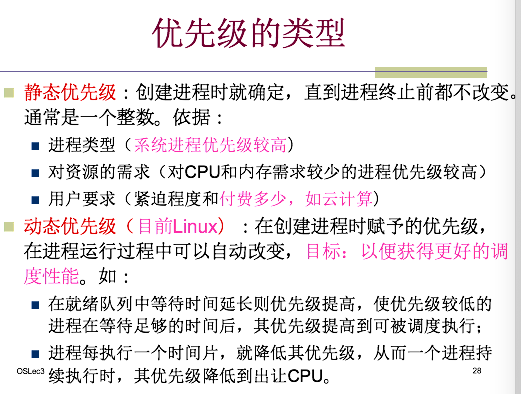




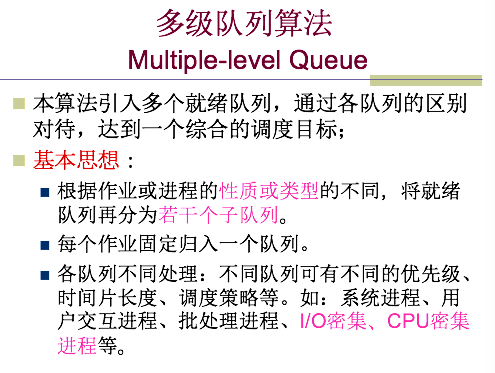


* + - 基于优先级的调度算法

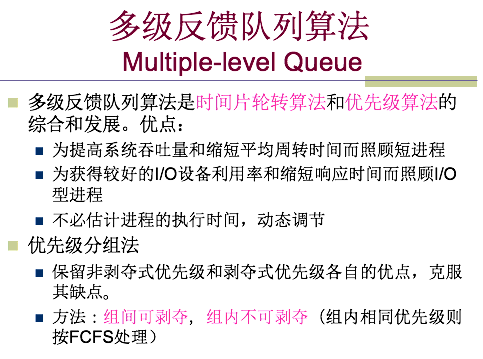


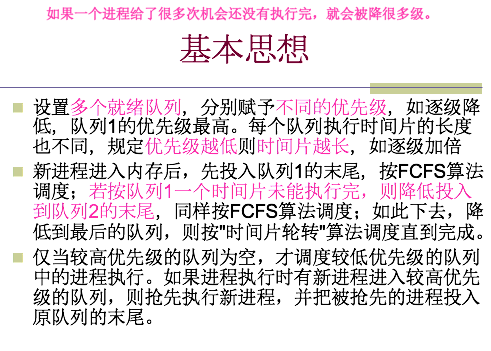


* + - 多级队列算法

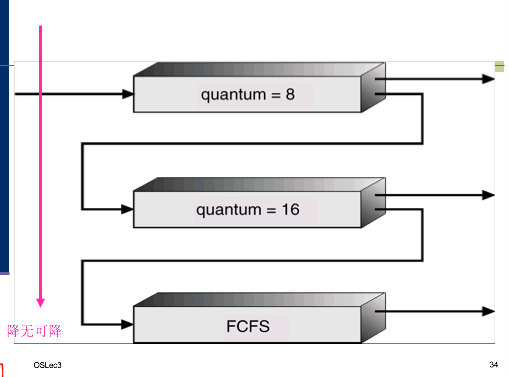


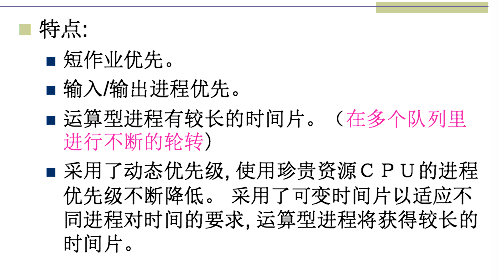
* + - 多级反馈队列算法



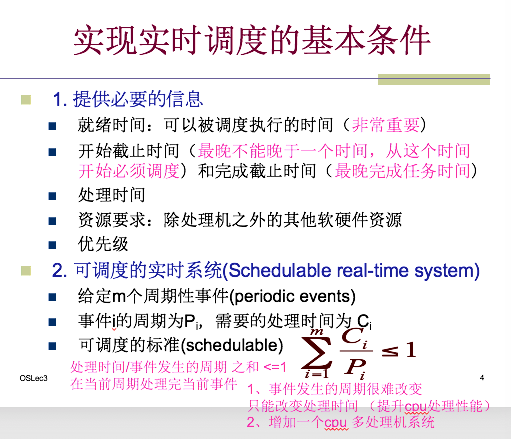


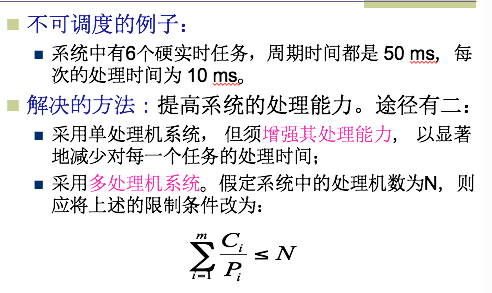


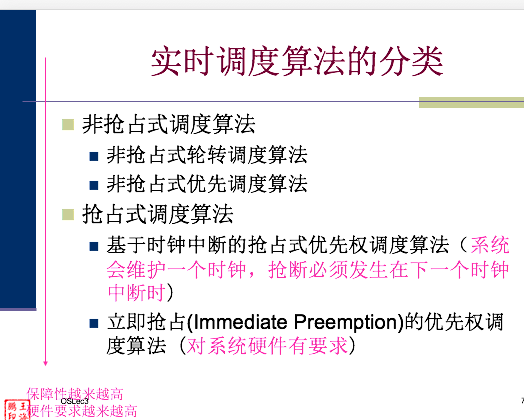




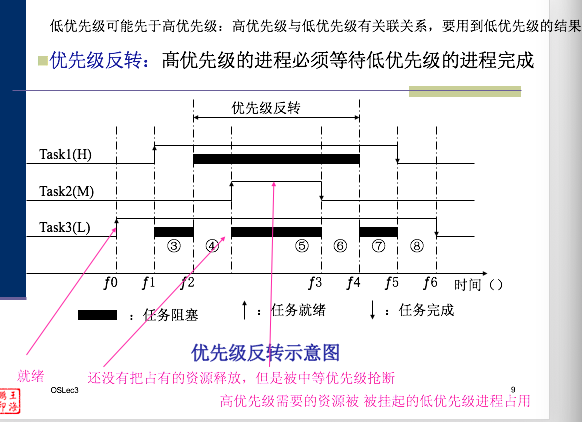
* + - 批处理系统（不管用户的输入输出）
      * FCFS
      * Shorted job first（非剥夺）
      * Shortest remaining time next（剥夺）
    - 交互式系统
      * round robin
      * priority scheduling（时间轮换）
      * round robin with multiple feedback（CTSS多级反馈）
      * lottery scheduling（抽奖抓阄 公平性比较好）
    - 实时系统
      * 硬实时：必须满足绝对的截止时间
      * 软实时：不希望错失截止时间，但可以容忍
* 第六讲
  + 实时调度 比较简单 什么叫实时系统 硬实时 软实时
    - 实时系统
      * 能够实现在指定或者确定的时间内完成系统功能和外部或内部、同步或异步时间做出响应的系统。
      * 在实施计算中，系统的正确性不仅仅依赖于计算的逻辑结果，而且依赖于结果产生的时间。
    - 实施任务
      * 周期性实施任务——硬实时任务
      * 非周期性实时任务——软实时任务
  + 可调度性要求 公式插书
    - 实现实时调度的基本条件
      * 提供必要的信息
        + 就绪时间：可以被调度执行的时间
        + 开始截止时间和完成截止时间
        + 处理时间
        + 资源要求：除处理机之外的其他软硬件资源
        + 优先级
      * 可调度的实时系统
        + 给定m个周期性事件
        + 事件i的周期为Pi，需要的处理时间为Ci
        + 可调度的标准
      * 采用抢占式调度机制
        + 当一个优先权更高的任务到达时，允许将当前任务暂时挂起，而令高优先权的任务立即投入运行，可满足该实时任务对截止时间的要求。
      * 具有快速切换机制
        + 对外部中断的快速响应能力
        + 快速的任务分派能力







* + 优先级继承 优先级反转 什么意思
    - 优先级反转
      * 现象
        + 高优先级的进程必须等待低优先级的进程完成
        + 高优先权进程被低优先级进程阻塞，反而使具有中优先级的进程先于高优先级的进程执行，导致系统的崩溃。
      * 原因
        + 多任务共享资源
      * 影响
        + 造成任务调度的不确定性
        + 严重时可能导致系统崩溃



* + - 优先级继承（解决优先级反转问题）
      * 当高优先级任务等待低优先级任务时，低优先级任务继承高优先级任务的优先级
      * 当低优先级释放资源时，立即把其优先权降低到原来的优先权
  + 多处理器 对称 非对称
    - 与但处理机调度的区别
      * 注重整体运行效率（而不是个别处理机的利用率）
      * 更多样的调度算法
      * 多处理机访问OS数据结构是的互斥（对于共享内存系统）
      * 调度单位广泛采用线程
    - 对称式多处理系统
      * 按控制方式，SMP算法可分为
        + 集中控制

静态分配

每个CPU设立一个就绪队列，进程从开始执行到完成，都在同一个CPU上。

优点：调度算法开销小。

缺点：容易出现忙闲不均。

动态分配

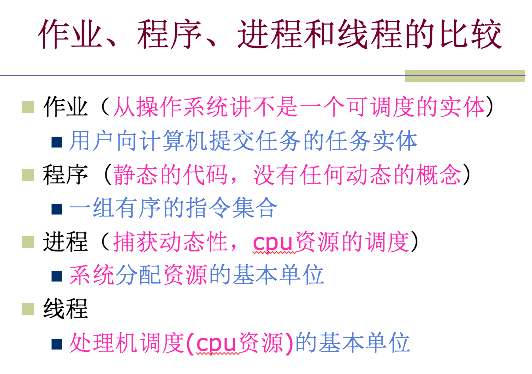
各个CPU采用一个公共就绪队列，队首进程每次分派到当前空闲的CPU上执行。

* + - * + 分散控制

自调度

各个CPU采用一个公共就绪队列，没各处理机都可以从队列中选择恰当进程来执行（每个CPU有自己的调度原则）。需要对就绪队列的数据结构进行互斥访问控制

* + Unix都算了 不看了
* 第七讲
  + 线程 进程 程序 概念 定义 区别
    - 进程
      * 资源分配单位（存储器、文件）和CPU调度（分派）单位。
      * 进程是用于自己资源的单元体
      * 进程是被调度分派在处理器上运行的单元体。
      * 缺点：时间空间开销大，先知并发度的提高。
    - 线程
      * 作为CPU调度单位，而进程只作为其他资源分配单位。
      * 只拥有必不可少的资源，如：线程状态、寄存器上下文和栈。
      * 同样具有就绪、阻塞和执行三种基本状态。
    - 线程的概念
      * 线程是进程内一个相对独立的、可调度的执行单元。
      * 进程中的一个运行实体，是一个CPU调度单位
      * 资源的拥有者还是进程
    - 多线程机制
      * 一个进程可以有多个线程，这些线程共享进程资源，驻留在相同的地址空间，共享数据和文件。
      * 一个线程修改了一个数据项，其他线程可以读取和使用此结果数据。一个线程打开并读一个文件时统一进程中的其他线程也可以同时读此文件。
      * 这些线程运行在同一进程的相同的地址空间内。
    - 引入线程的好处
      * 创建一个新线程花费时间少
      * 两个线程的切换花费时间少
      * 因为同一进程内的线程共享内存和文件，因此他们之间相互通信无需调用内核。
      * 适合多处机及系统。
    - 进程和线程的比较
      * 调度
        + 线程上下文切换比进程上下文切换要快得多。
        + 线程的创建时间比进程短，终止时间比进程短。
        + 同进程内的线程切换时间比进程短。
      * 拥有资源
        + 进程间相互独立，统一进程的各线程间资源共享——某进程内的线程在其他进程不可见。
        + 由于同进程内线程间共享内存和文件资源，可直接进行不通过内核的通信
      * 系统开销
        + 线程减小并发执行的时间和空间的开销。
      * 并发性
        + 在系统中建立更多地线程来提高并发程度。
  + 核心进程 内核线程 轻量级线程 每种线程优缺点
    - 核心级线程（内核线程）
      * 由操作系统内核进行管理。操作系统内核给应用程序提供相应的系统调用和应用程序接口API，以使用户程序可以创建、执行、撤销线程。
      * 优点
        + 对多处理器，核心可以同时调度同一进程的多个线程。
        + 阻塞是在线程一级完成。
        + 核心历程是多线程的。
      * 缺点
        + 在同一进程内的线程切换调用内核，导致速度下降。
    - 用户级线程（用户线程）
      * 管理过程全部由用户程序完成，操作系统内核心只对进程进行管理。
      * 优点
        + 线程切换不调用核心
        + 调度是应用程序特定的：可以选择合适的算法
        + ULT可运行在任何操作系统之上（只需线程库）
      * 缺点
        + 大多数系统调用是阻塞的，因此核心阻塞进程，故进程中所有线程将被阻塞
        + 核心只将处理器分配给进程，统一进程中的两个线程不能同时运行于两个处理器上。
    - 内核控制线程（轻量级线程LWP）
      * 每一个进程都可拥有多个LWP，可通过系统调用来获得内核提供的服务，这样，当一个用户级线程运行时，只要将它连接到一个LWP上，此时它便具有了内核支持线程的所有属性。
  + 线程状态 和进程状态差不多
    - 执行状态、就绪状态、阻塞状态



* 第八讲
  + 进程间关系 什么叫同步 互斥
    - 进程同步
      * 指进程之间的一种协调配合关系，它表现在进程的执行顺序的规定上。
      * 概念：相互协调的几个进程在某些确定点上协调他们的工作，一个进程到达了这些点之后，除非另一进程已完成了某些操作，否则就需要停下来等待这些操作的完成。
    - 进程互斥
      * 两个或两个以上的进程由于不能同时使用同一资源，只能一个进程使用完了另一个进程才能使用的现象。
      * 互斥关系也是一种协调关系，从广义上讲它也属于同步关系的范畴。
  + 什么叫临界资源 临界区
    - 临界资源（一类资源）
      * 一次仅允许一个进程访问的资源
      * 资源状态为临街：0或1
    - 临界区（一段代码）
      * 临界段，在每个程序中，访问临界资源的那段程序。
      * 对临界段的设计有如下原则
        + 每次至多只允许一个进程处于临界段中
        + 对于请求进入临界段的多个进程，在有限时间内只让一个进入。
        + 进程只应在临界段中停留有限时间。
  + 临界区的访问的基本原则 忙则等待 空闲让进 优先等待 让全等待 每个解释含义 非常重要
    - 同步机制应遵循的准则
      * 空闲让进
        + 当临界资源处于空闲状态，允许一个请求进入临界区的进程立即进入临界区，从而有效地利用资源。
      * 忙则等待
        + 已经有进程进入临界区时，意味着相应的临界资源正在被访问，所以其他准备进入临界区的进程必须等待，来保证多进程互斥。
      * 有限等待
        + 对要求访问临界资源的进程，应该保证该进程能在有效的时间内进入临界区，防止死等状态。
      * 让权等待
        + 当进程不能进入临界区，应该立即释放处理机，防止进程忙等待。
  + 软件实现硬件实现就算了 不给大家增加负担了
* 第九讲 （重要 15分）
  + 同步互斥 肯定要考 编程题 也不排除简答 什么叫信号量
    - 信号量
      * 信号量是一个数据结构，由两个变量构成
        + 整型变量V
        + 指针变量S
      * 初始化指定一个非负整数值，表示空闲资源总数（又称为“资源信号量”）
      * 若为非负值表示当前的空闲资源数，若为负值其绝对值表示当前等待临界区的进程数。
      * 信号量的值只能被P、V操作原语进行改变。
    - P（S）：表示申请一个资源
    - V（S）：表示释放一个资源
* 第十讲
  + 进程间通信 低级高级通信
    - 进程间通信
      * 是指进程之间可直接以较高的效率传递较多数据的信息交换方式。
      * 低级通信
        + 只能传递状态和控制信息等，进行简单的信号交换。
        + 优点

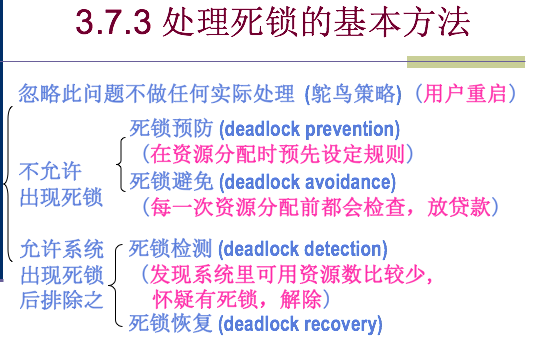
速度快

* + - * + 缺点

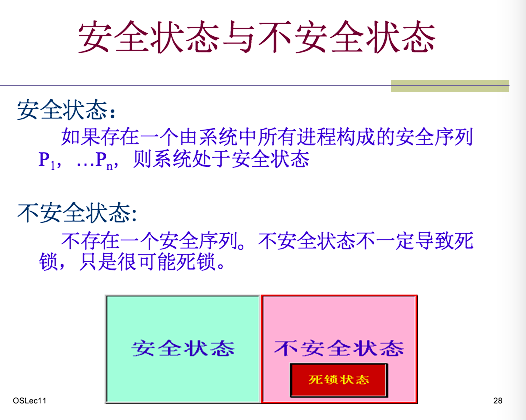
传送信息量小：效率低，每次通信传递的信息量固定，若传递较多信息则需要进行多次通信。

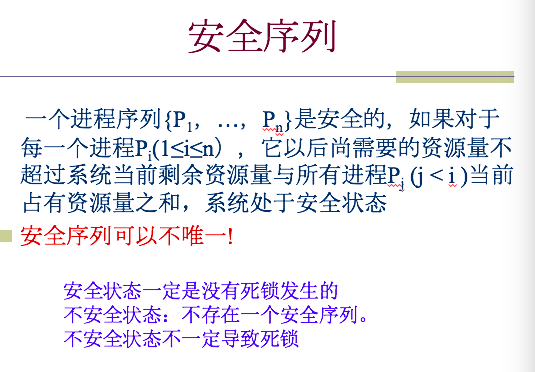
变成复杂：：用户直接实现通信细节，易出错。

* + - * 高级通信
        + 高效、大量的数据传递
  + 通信类型 共享消息邮箱管道 思想 优缺点
    - 共享存储器系统
      * 通过数据区的共享，写入与读出达到通信的目的。
      * 基于共享数据结构的通信方式
        + 诸进程公用某些数据结构，进程通过他们交换信息。如生产者-消费者问题中的有界缓冲区。
      * 基于共享存储区的通信方式
        + 高级通信，在存储器中划出一块共享存储区，进程在通信前，向系统申请共享存储区中的一个分区，并指定该分区的关键字，若系统已经给其他进程分配了这样的分区，则将该分区的描述符返回给申请者。接着，申请者把获得的共享存储区连接到本进程上，此后可读写该分区。
      * 以上两种方式的同步和互斥都要由进程自己负责。
    - 消息传递系统
      * 直接通信方式：消息缓冲
        + 采用进程的消息缓冲队列
        + 消息发送者将消息直接放在接受者的消息缓冲队列
      * 间接通信方式：邮箱通信
        + 利用中间者——信箱、邮局来传递信件
        + 发送进程将消息发送到信箱中，接收进程从信箱中取出消息。
      * 前者需要两进程都存在，后者不需要。
    - 管道通信（共享文件方式）
      * 用以连接读、写进程的共享文件
* 第十一讲
  + 死锁 内容多 什么叫死锁 名词解释
    - 死锁
      * 一组进程中，每个进程都无限等待被该组进程中另一进程所占有的资源，因而永远无法得到资源，这种现象称为进程死锁，这一组进程就成为死锁进程。
  + 死锁的四个必要条件 互斥占有等待不可剥夺…… 解释
    - 产生死锁的四个必要条件
      * 互斥
        + 指进程对所分配到的资源进行排他性使用，即在一段时间内某资源只能由一个进程占有。如果此时还有其他进程申请该资源，则他只能阻塞，直至占有该资源的进程释放。
      * 占有等待
        + 进程已经保持了至少一个资源，但又提出了新的资源要求，而该资源又已被其他进程占有，此时请求进程阻塞，但又对已经获得的其他资源保持不放。
      * 不可剥夺
        + 进程已获得的资源，在未使用完之前，不能被剥夺，只能在使用完时由自己释放。
      * 环路等待
        + 在发生死锁时，必然存在一个进程-资源的封闭的环形链：P0正在等待P1占用的资源，P1正在等待P2占用的资源，……，Pn正在等待已被P0占用的资源
  + 死锁预防避免检测与恢复 什么叫 解释
    - 死锁预防
      * 在系统设计时确定资源分配算法，保证不发生死锁。
      * 破坏死锁的四个必要条件
    - 死锁避免
      * 在系统运行过程中，对进程发出的每一个系统能够满足的资源申请进行动态检查，并根据检查结果决定是否分配资源，若分配后系统可能发生死锁，则不予分配，否则予以分配。
    - 死锁检测
      * 允许死锁发生，操作系统不断监视系统进展情况，判断死锁是否发生。
      * 一旦死锁发生则采取专门的措施，接触死锁并以最小的代价恢复操作系统的正常运行。
      * 检测时机
        + 当进程等待时检测死锁（缺点是系统开销大）
        + 定时检测
        + 系统资源利用率下降时检测思死锁。
    - 死锁jiechu
      * 以最小的代价恢复系统的运行
      * 方法
        + 重新启动（整个操作系统）
        + 撤销进程
        + 剥夺资源
        + 进程回退（和剥夺资源类似）

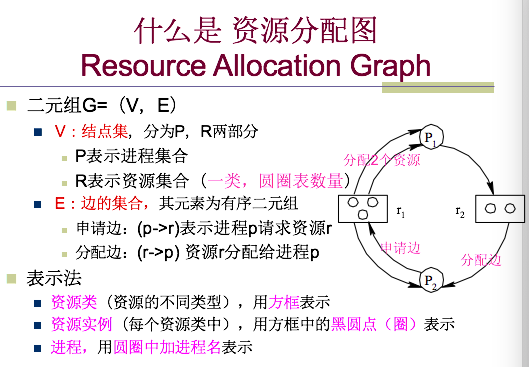


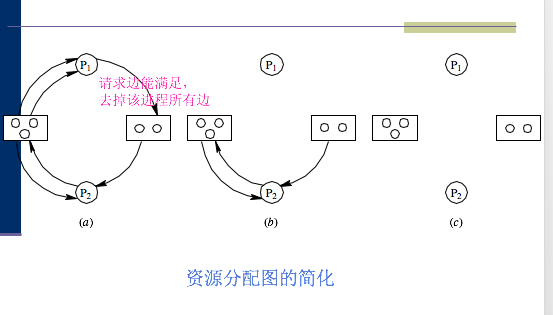
* + 什么叫安全状态安全序列 不安全状态
    - 安全状态
      * 指系统能按某种进程顺序（P1，P2，P3，……，Pn）来为每个进程Pi分配其所需资源，直至满足每个进程对资源的最大需求，使每个进程都可顺利的完成。如果系统无法找到这样一个安全序列，则称系统处于不安全状态（称<P1，P2，……，Pn>序列为安全序列，不唯一）

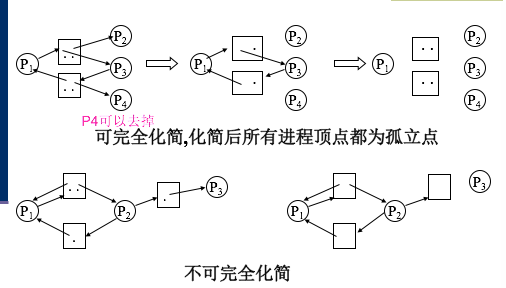




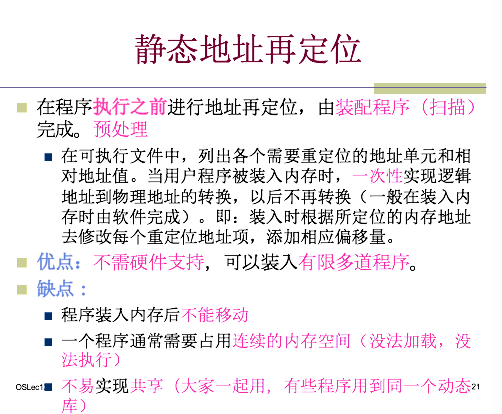
* + 银行家算法是一个计算题 考的概率高 好好准备
* 第十二讲
  + 死锁避免 给一个当前状态 请求 分析每个请求是否能满足 最后请求结束状态
  + 分配图 资源分配图 计算题 几年出一次

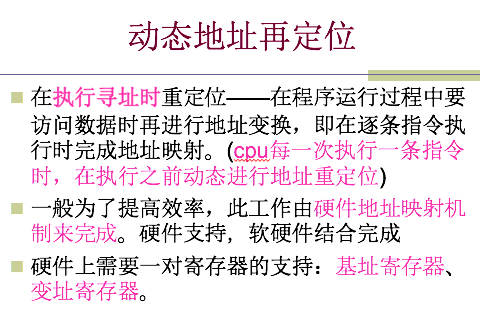


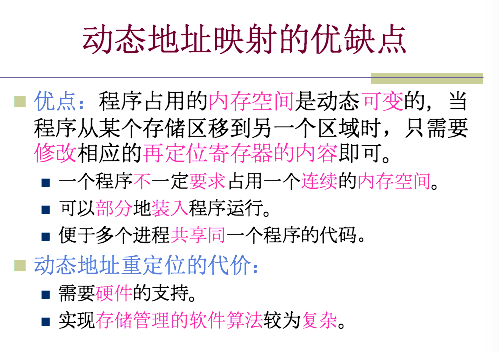




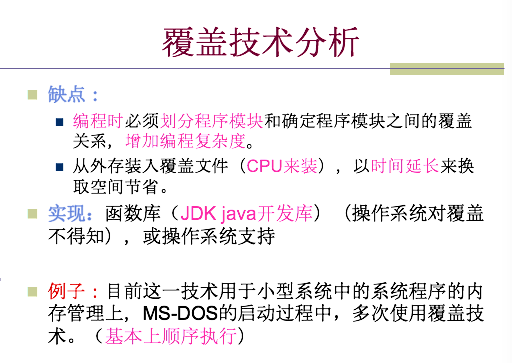
* 第十三讲
  + 存储 什么叫逻辑地址 物理地址 存储的几个功能
    - 物理地址（绝对地址、实地址）
      * 内存中存储单元的地址。物理地址可直接寻址。
    - 逻辑地址
      * 用户的程序经过汇编或编译后的目标代码的地址。
    - 存储管理的四大功能
      * 存储空间的管理、分配和回收
        + 记录内存的使用情况——设置相应的内存分配表（内存分配回收的依据）
        + 静态存储分配；动态存储分配
        + 分配和回收算法及相应的数据结构
      * 地址再定位（地址变换、地址映射）
        + 可执行文件生成中的链接技术
        + 程序加载（装入）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
      * 存储共享和保护
        + 两个或多个进程共用内存中相同区域
        + 代码和数据共享
        + 地址空间访问权限
        + 保护过程——防止地址越界、当值操作越权
      * 存储器扩充
        + 存储器的逻辑组织和物理组织，虚拟存储
        + 覆盖
        + 交换
  + 地址分配 静态动态 几个概念搞清楚 静态动态之间区别



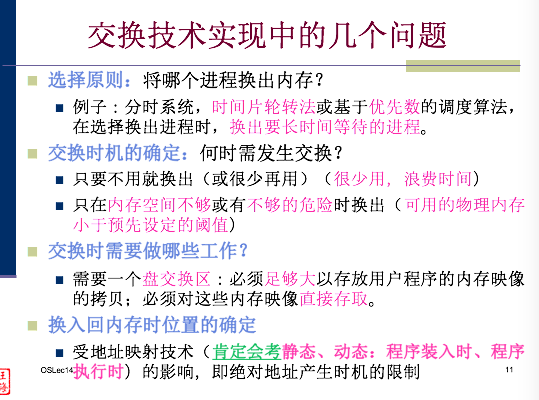


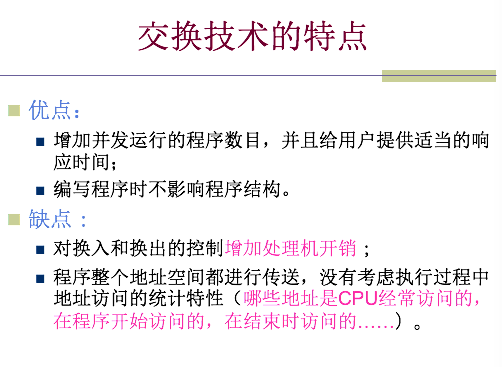


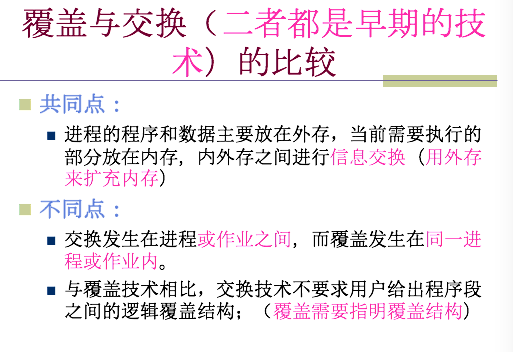
* + 分区单一知道一下思想就行不深究
  + 碎片 什么叫碎片 外碎片 内碎片
    - 碎片
      * 存在于已分配的分区之间的一些不能充分利用的空白区
    - 外碎片
      * 没有被分配出去
    - 内碎片
      * 已经被分配出去
      * 明确知道属于哪个进程
* 第十四讲
  + 覆盖 交换 什么叫 原理 比较
    - 覆盖技术
      * 一个作业的若干程序段，或几个作业的某些部分共享一个存储空间。（不把全部程序放进内存，而是把某一段需要的程序放入，用完之后用新的程序段覆盖）
      * 原理
        + 一个程序的几个代码段或数据段，按照时间先后（执行时间）来占用公共的内存空间
        + 把程序划分为若干个功能上相对独立的程序段，按照其自身的逻辑结构将那些不会同时执行的程序段共享同一块内存区域
        + 程序段先保存在磁盘上，当有关程序段的前一部分执行结束，把后续程序段调入内存，覆盖前面的程序段。
        + 一般要求作业各模块之间有明确的调用结构，程序员要向系统指明覆盖结构，然后由操作系统完成自动覆盖。



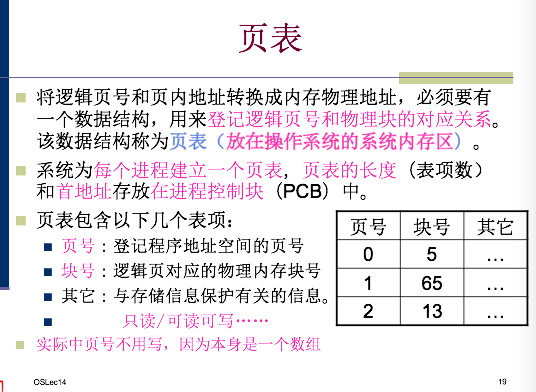
* + - 交换技术
      * 系统将内存中某些进程暂时移到外存，把外存中某些进程换进内存，占据前者所占用的区域（以作业为单位）
      * 原理
        + 暂停执行内存中的进程，将整个进程的地址保存到外存的交换区中，而将外存中由阻塞变为就绪的进程的地址空间读入到内存中，并将该进程送到就绪队列。
        + 交换单位为整个进程的地址空间
        + 与分区存储管理配合使用
        + 又称作“对换”或者“滚进滚出”



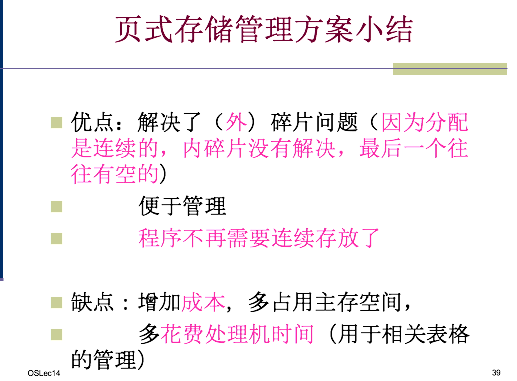




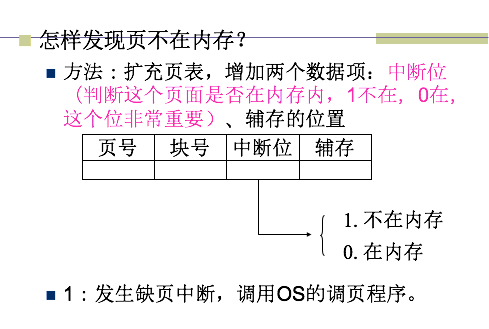
* + 页式存储管理 页怎么分配 页地址 逻辑怎么分配 什么叫页表 怎么做转换 给出逻辑地址 映射成物理地址 有可能会越界

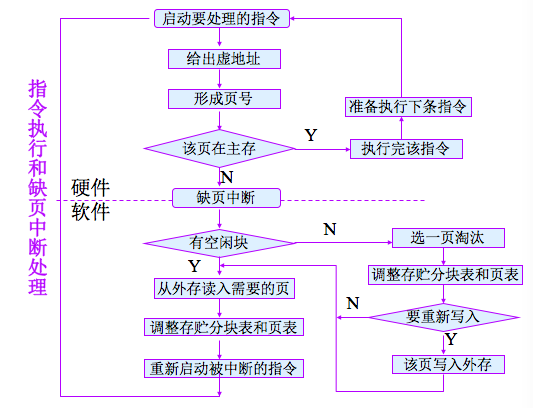


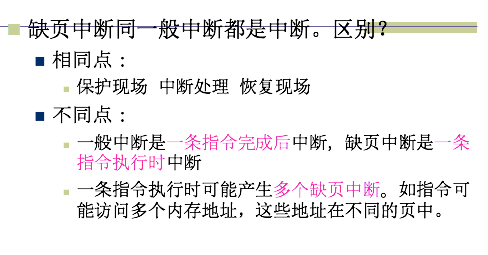
* + 地址是一维还是二维
  + 页式优缺点



* 第十五讲
  + 虚拟存储是什么
    - 虚拟存储解决内存小、作业大、作业多的矛盾
    - 原理：程序执行时的局部性原理；虚拟存贮技术
    - 实现方式：操作系统统一管理各级存储器；内存中只存放当前要执行的程序部分，其余的保存在外存上，内、外存统一编址，扩大寻址空间。
  + 缺页中断 处理过程 和一般中断区别
    - 处理过程
      * 在地址映射过程中，在页表中发现所要访问的页不在内存，则产生缺页中断。
      * 操作系统接到此中断信号后，就调出缺页中断处理程序，根据页表中给出的外存地址，准备将该页调入内存。
      * 此时应将缺页的进程挂起（调页完成唤醒）
      * 如果内存中有空闲块，则分配一个块，将要调入的页装入该块，并修改页表中相应页表项目的驻留位及相应的内存块号。
      * 若此时内存中没有空闲块，则要淘汰某页（若淘汰页在内存期间被修改过，还需要将其写回外存）

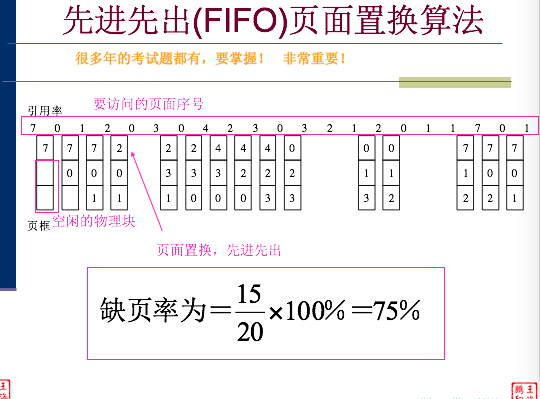


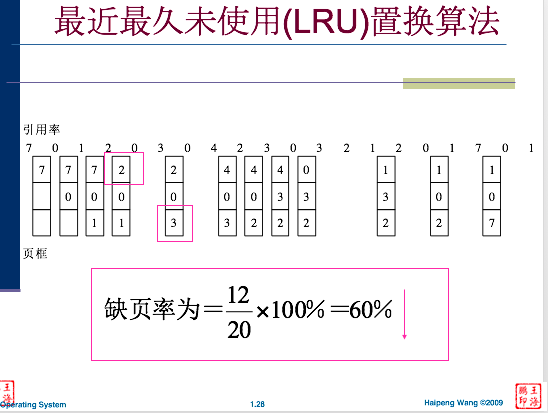


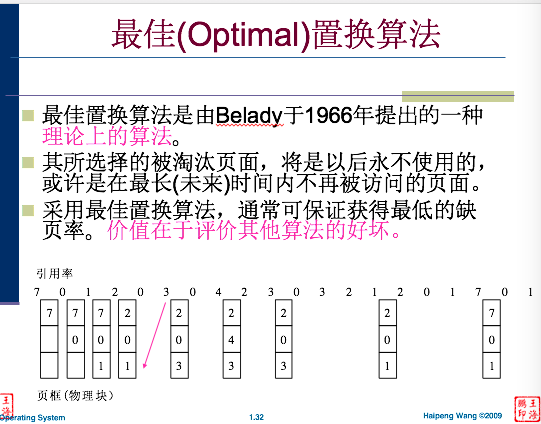


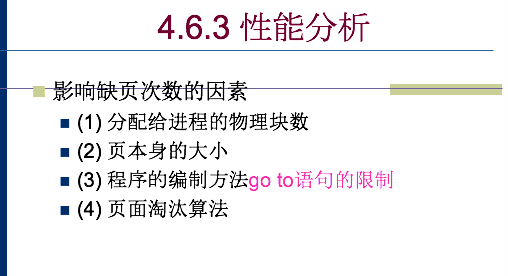
* + 页面置换算法 算置换率 给页面访问序列 算缺页率 先来先服务 最简……



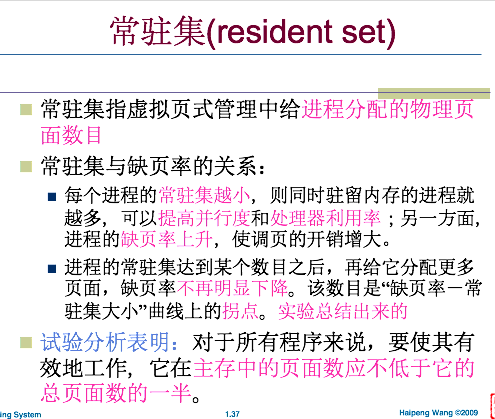


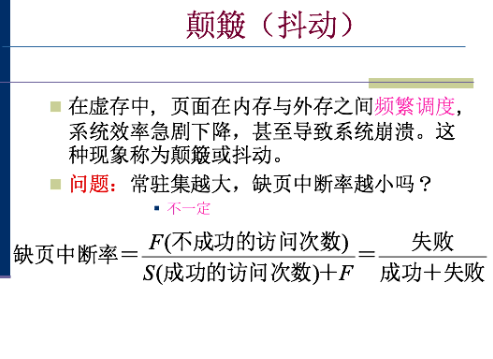


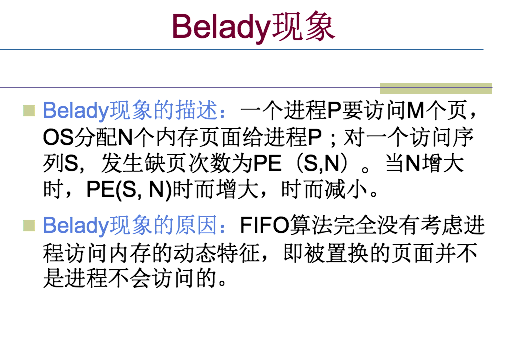




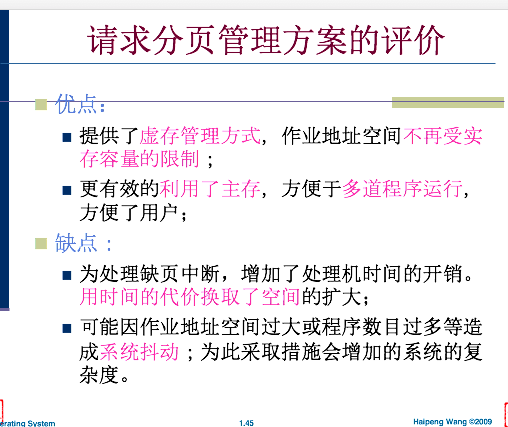
* + 性能 常驻集什么叫 颠簸 抖动 ……现象







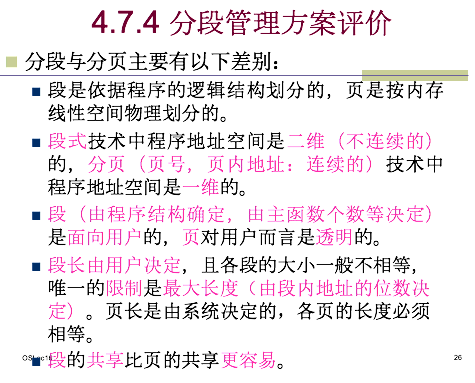


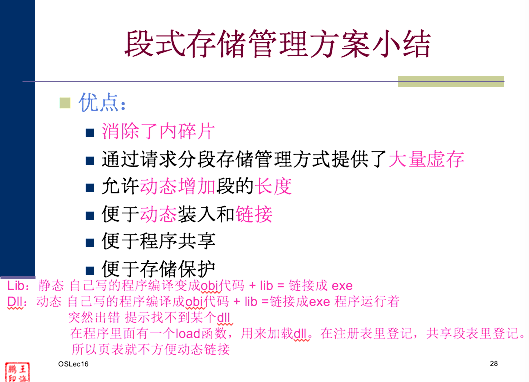


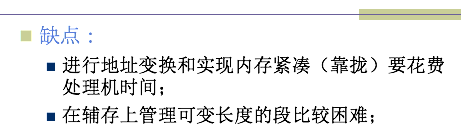
* 第十六讲
  + 段式存储
  + 段表 结构 地址映射过程



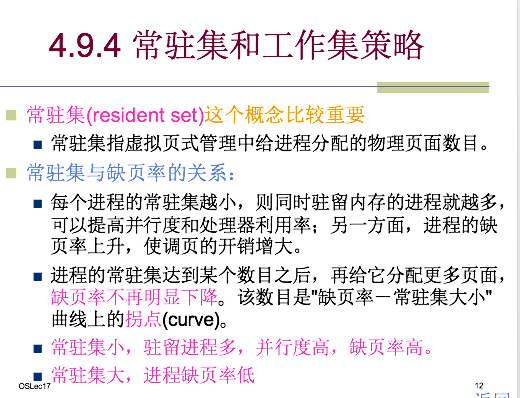
* + 快表 什么叫快表
  + 段式存储管理优缺点



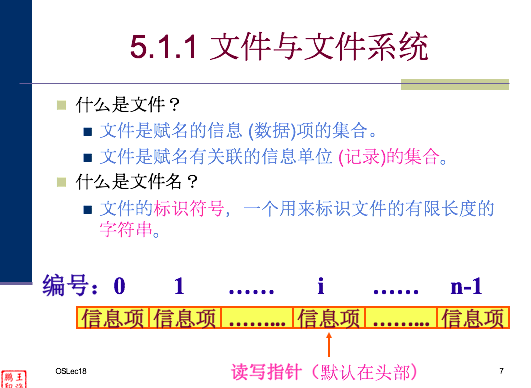




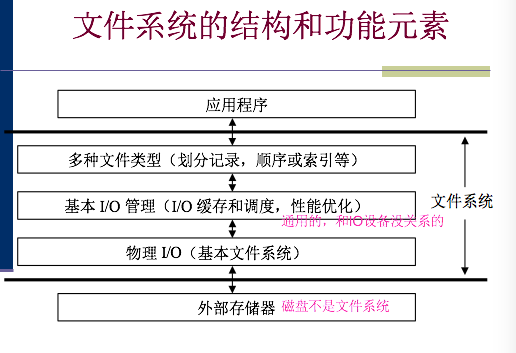
* 第十七讲
  + 分段 段页式 思想 优缺点
  + 比较简单



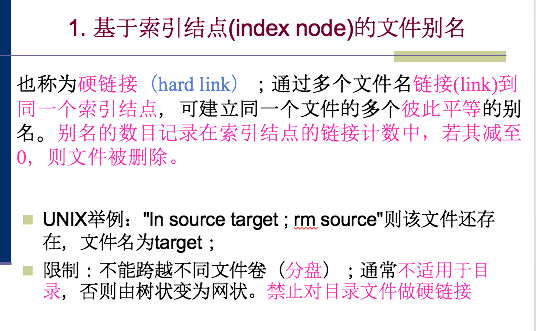
* 第十八讲
  + 文件 什么叫文件 文件系统 文件目录 目录文件 搞明白

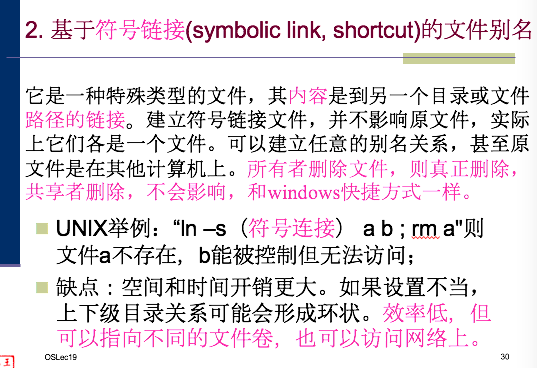


* + 文件类型 物理类型 文件系统主要目录结构 图

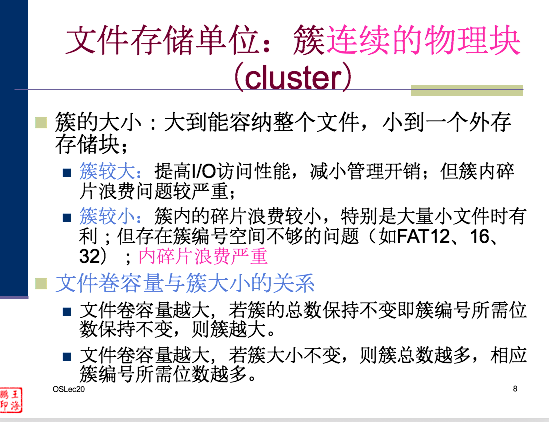


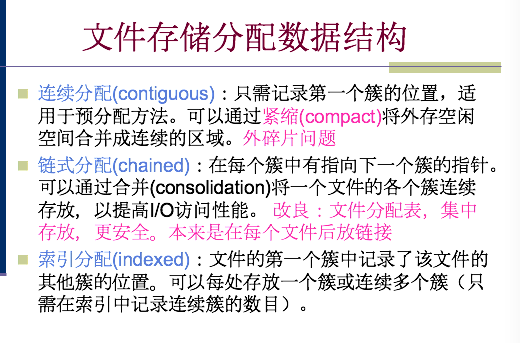
* + 文件系统的逻辑结构比较简单 了解一下
  + 物理结构 索引连接连续结构 特点 搞明白
  + 三种文件物理结构的比较
* 第十九讲
  + 文件目录
  + 目录文件
  + 文件系统的特点
  + 索引节点
  + 硬链接 符号链接 搞明白



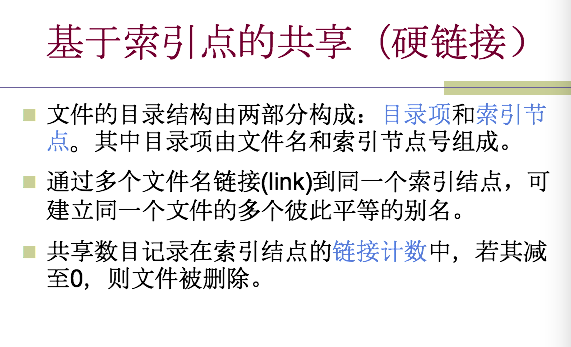


* 第二十讲
  + 比较简单 磁盘 时延
  + 簇 什么叫

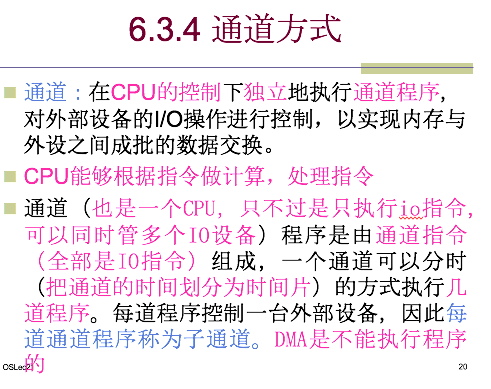


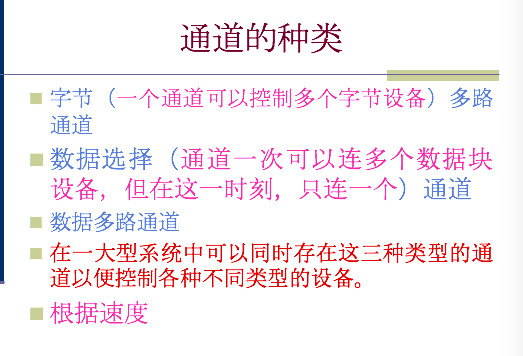


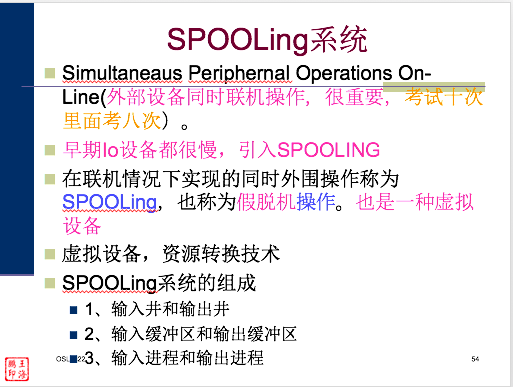




* + 成组链表 了解一下就行了
  + 硬链接很重要
* 第二十一讲
  + IO 字符设备 块设备 独占设备 虚拟设备 控制块
  + IO编址
  + 四种IO控制方式 很重要 特点 优缺点 区别 还有SPOOLING五种









* 第二十二讲
  + IO软件 缓冲 磁道时间
  + 调度算法 计算题 时间
  + 优化策略 知道一下就好
* 讲难点
  + 图书馆
    - 互斥问题
      * 座位
      * 登记本
      * 定义信号量
      * seat =100 mutext=1
      * p（seat）
      * p（mutext）
      * 凡是阻塞的 直接定义信号量
      * 不等待的 定义为全局变量 对全局变量的访问要保护 定义一个互斥信号量 保护它。
  + 过桥问题
    - 互斥 定义一个 信号量
    - 同方向同时过 类似读者写者问题 读可以并发读，写可以并发写 定义两个信号量
      * 记录从东向西的人数
      * int ： D = 0
      * int： W = 0
      * S SW = 1