操作系统复习

~~没都不看的OS复习整理~~

1.概论

操作系统的基本功能：

- 提供操作界面
- 控制程序运行
- 管理系统资源
- 配置系统参数
- 监控系统状态
- 工具软件集合

定义

操作系统是一个大型的程序系统，负责计算机系统软、硬件资源的分配；控制和协调并发活动；提供用户接口，使用户获得良好的工作环境。

即：

- 管理和调度资源
- 为用户提供接口

特性

- 并发性：同时处理多个任务的能力
- 共享性：为多个并发任务提供资源共享
- 不确定性：具有处理随机事件的能力

功能

1. 进程管理
2. 存储管理
3. 设备管理
4. 文件管理

硬件发展的四个阶段

1. 手工操作
    - 电子管时代
2. 单道批处理系统（联机批处理/脱机批处理）
    - 晶体管时代

   还有**集成电路时代**和**大规模集成电路时代**
3. 多道批处理系统
    - 多道程序设计技术
    - 宏观上并行，微观上串行
4. 分时操作系统

中断技术

CPU收到外部(中断)信号后，停止当前工作，转去处理该外部事件，处理完毕后回到原来工作的中断处(断点)继续原来的工作

通道技术

专门处理外设与内存之间的数据传输的处理机

分时技术

主机以很短的“时间片”为单位，把CPU循环地轮流分配给每个作业使用，直到全部作业被运行完

特点：

- 时间片：较短的时间间隔
- 响应及时：主机独占时间片

分时系统的特点

- 多路调制性
- 独占性
- 交互性

分时系统的衍化：

- 实时操作系统：硬实时（限时完成），软实时（尽可能快的完成）
- 微机操作系统（PC机）
- 多处理机系统：

  特点：具有并行处理能力；紧耦合，存在瓶颈，可扩展性差；不支持大规模并行计算，不支持分布处理。
- 网络操作系统
- 分布式操作系统：

  分布的多个通用资源部件，经过网络互联，由操作系统对资源进行全局统一的管理和调度

  特点：
  - 可扩展性、增加性能、高可靠性
  - 特殊的计算机网络
  - 主机自治，又互相协调，运行分布式程序

UNIX是**第一个**实用化的分时操作系统。

2.操作系统结构与硬件支持

操作系统的逻辑结构

整体式结构

以**模块**为基本单位构建

优点：模块的设计、编码和调试独立；模块间可以自由调用

缺点：错误易扩散；开发维护困难；可伸缩性差

层次结构

功能模块按调用次序排成若干层，各层单向依赖或单向调用

优点：

- 结构清晰，避免循环调用
- 整体问题局部化，系统的正确性容易保证
- 有利于操作系统的维护、扩充、移植

微内核结构

- 微内核：足够小，提供OS最基本的核心功能和服务
- 核外服务器：由若干服务器或进程共同构成

处理机的态

支持操作系统的最基本硬件结构：**CPU/内存/中断/时钟**

CPU态

CPU的工作状态，对资源和指令使用权限的描述。硬件要求在处理器中包含有一个**模式位。**

态的分类：

- 核态：管理程序/OS内核，能访问所有资源和执行所有指令
- 用户态（目态）
- 管态：介于上述两态之间

用户态向核态转换

- 用户请求OS提供服务
- 发生中断
- 用户进程产生错误（内部中断）
- 用户态企图执行特权指令

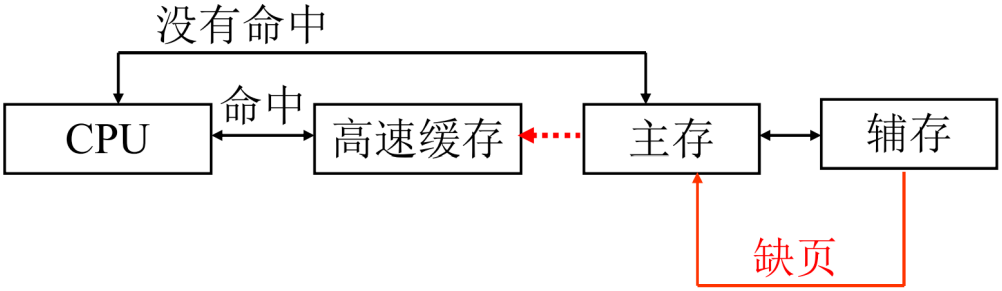
核态向用户态转换的情形

- 中断返回(IRET)

存储器(内存)

分类：主存(直接和CPU交换信息)/辅存，半导体/磁/光存储器，RAM(随机存取，掉电失忆)/ROM(只读，永久存)

CPU读取指令或数据时的访问顺序：



中断机制

定义：指CPU对突发的外部事件的反应过程或机制

目的：

- 实现并发活动
- 实现实时处理
- 故障自动处理

中断源

引起系统中断的事件

中断类型

- 强迫中断：程序没有预期，如I/O、外部中断

  自愿中断

- 外中断：CPU外部事件引起
  - 不可屏蔽中断：中断原因紧要，CPU必须响应
  - 可屏蔽中断

  内中断

断点：程序中断的地方，将要执行的下一指令的地址，一般由CS:IP获得

现场：程序正确运行所依赖的信息集合

现场保护和恢复：通过栈实现

中断响应过程

1. 识别中断源
2. 保护断点和现场
3. 装入中断服务程序CS:IP
4. 进入中断服务程序
5. 恢复现场和断点
6. 中断返回：IRET

中断响应的实质

- 交换指令执行地址
- 交换CPU的态
- 工作
  - 现场保护和恢复
  - 参数传递（通信）

3.用户界面

操作系统的启动

实模式：程序按照8086寻址方法访问 `0~FFFFFh(1MB)` 空间，寻址方式为物理地址(20位)

1M空间的划分：

- 前面640K:基本内存
- 中间128K：显卡显存

- 末尾256K：BIOS

系统BIOS：

- 基本输入/输出系统
- 位置 `F0000-FFFFF`
- 功能：
  - **CMOS设置**
  - **基本I/O设备中断服务** (中断类型号 `10H~1FH`，其中INT 13H为软盘I/O调用 AH=02H读扇区

    CHS扇区定位机制：柱面-磁头-扇区-逻辑扇区
  - **POST(上电自检)**

    按下 `PowerOn` 或者 `Reset` 键执行 `FFFF0` 单元的指令

    `JUMP POST; POST 加电自检`
  - **系统自举/加载OS**

    开机时**将OS载入内存**并运行为用户建立用户环境

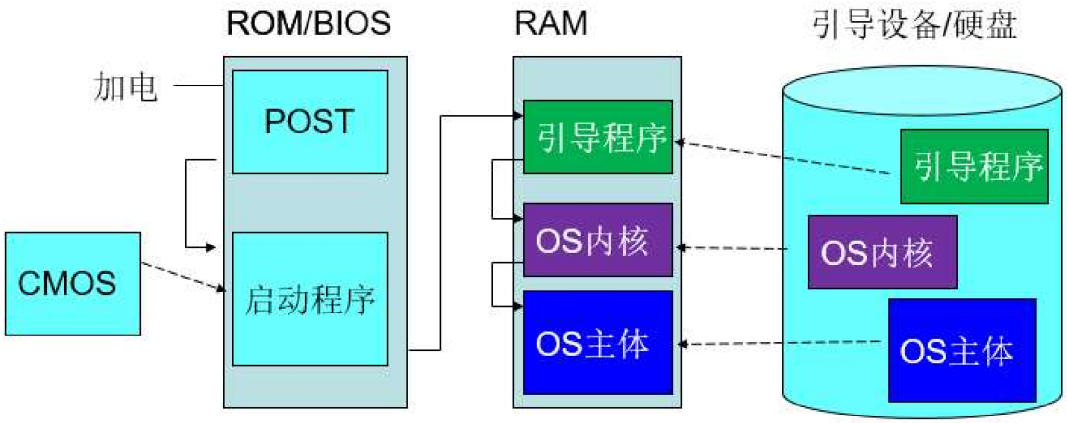
启动过程

加电/复位，bios启动，启动引导程序，内核初始化

- 从加电到用户工作环境准备好的过程

1. **初始引导**

   把**OS内核**装入内存并使之开始工作接管计算机系统



   MBR：主启动记录：引导程序
2. **核心初始化**：OS内核初始化系统的核心数据
3. **系统初始化**：为用户使用系统作准备，使系统处于**待命状态**

WINDOWS/LINUX启动过程

MBR工作原理

操作系统的用户界面

OS提供给用户控制计算机的机制，又称**用户接口**

- 操作界面
- 系统调用

操作界面

- 图形用户接口
- 键盘命令：在控制台环境下接收键盘输入的命令
  - 普通命令

    dos，linux典型命令(`ls,man,cd,tar...`)
  - 批处理：普通命令的集合，批执行，由cmd解释执行（`bat/PowerShell`）
  - **shell**：是操作系统与用户交互的界面；通过控制台执行用户命令；本身不执行命令，组织和管理命令。

shell 脚本：通过类似程序的方式执行具有一定逻辑顺序的**命令序列**完成较复杂的功能和人机交互。

$shell \subseteq$ 脚本

系统调用

~~**新需求：**~~

- 用户模式下的进程可以将处理器切换到核态并执行OS的特定片段(特殊服务)
- OS代码不能直接返回用户程序执行，只能由OS自己返回（OS代码逻辑上为原子操作）

~~**解决方案：增加系统调用指令**~~

运行于核态，每个系统调用具有唯一的编号ID，调用过程会产生自愿中断。

通过**访管指令**和**访管中断**实现

> 系统调用是利用访管指令定义的指令，是用户在**程序一级**请求**操作系统服务**的一种手段。
> 补充：它不是一条简单的硬指令，而是**带有一定功能号的访管指令**。它的功能并非直接由硬件直接提供，而是→操作系统中的一段程序完成的，即由**软件方法实现**的。有显示调用和隐式调用之分。

Linux增加系统调用的过程：

1. 编写系统调用实现部分
2. 声明系统调用函数和编号
3. 编译LINUX内核
4. 编译和安装模块
5. 启动新LINUX内核
6. 编写应用程序测试新的系统调用

1-3章测试整理

7　关于操作系统启动过程说法错误的是：
A. 启动程序属于BIOS的一部分。
B. 安装操作系统的时候会修改甚至重写MBR。
√C. 引导程序采用文件的方式存在于硬盘。
D. GRUB是一个典型的引导程序。

> C. 引导程序确实在辅存中，但并不是以文件的形式，文件系统是在系统初始化之后才建立的。引导程序的主要功能是把操作系统的核心部分放到主存中，并让系统呈可接受命令的状态。

16　第一个采用分时技术实现的实用且广泛使用的操作系统是 (**UNIX**) 。

22　Linux系统中，系统调用功能是利用 (80) 号中断实现的。（16进制）

> INT 21H为DOS的系统中断

28　Linux操作系统是一个典型的微内核结构的操作系统。
A. 对
√B. 错

> unix一开始就是一体化的，linux由unix发展过来的，出于性能等考虑，也没有必须改的原因，就没大改设计结构。

4.进程管理

程序在并发环境下执行的问题：运行过程不确定，结果不可再现，程序运行被干扰。

进程定义

进程是程序在某个数据集合上的一次运行活动。

- 数据集合：软/硬件环境，多个进程共存/共享的环境

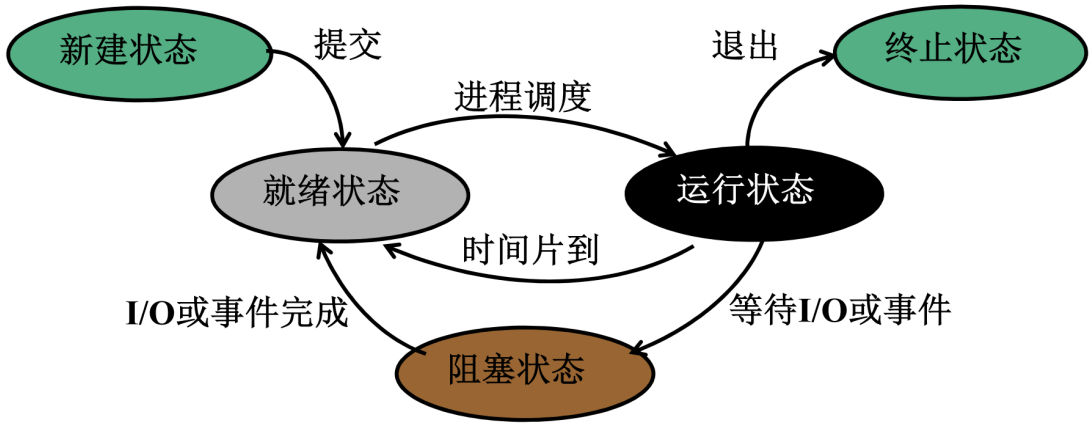
特征

- **动态性**：进程是程序的一次执行过程，动态产生/消亡
- **并发性**：进程可以同其它进程一起向前推进
- **异步性**：进程按照各自的速度向前推进
- **独立性**：进程是系统分配资源和调度CPU的单位

类型：系统进程/用户进程，偏CPU进程/偏IO进程

进程的状态❗

- 运行状态(Running)：进程已经占有CPU，在CPU上运行
- 就绪状态(Ready)：具备运行条件但由于无CPU，暂时不能运行
- 阻塞状态(Block)：因为等待某项服务完成或信号来到而不能运行的状态



Linux进程的状态

- 可运行态
  - 就绪：TASK_RUNNING，在就绪队列中等待
  - 运行：正在运行
- 睡眠态/阻塞态/等待态
  - 深度睡眠：不能被其它进程通过信号和时钟中断唤醒
  - 浅度睡眠

- 僵死态：进程终止执行，释放大部分资源，执行do_exit()后到达
- 挂起态：进程被挂起

进程控制块PCB

- 描述进程的状态、资源和相关进程的关系的一种**数据结构**
- PCB是进程的**标志**
- 创建进程时创建PCB，进程撤销后PCB同时撤销

❗进程 = 程序（代码+数据） + PCB

PCB的基本结构

- 标识符ID
- `status`:状态
- `next`:指向下一个PCB块的指针
- `start_addr`:程序地址
- `priority`:优先级
- CPU现场保留区(堆栈)
- 进程通信机制
- 家族关系
- 资源清单

进程控制

在进程生存全期间，对其全部行为的控制。

- **创建进程**
  - 创建空白PCB
  - 赋予进程标识符ID
  - 为进程分配空间
  - 初始化PCB
  - 插入进程队列：新进程插入就绪队列
- **进程撤销**：收回进程占有的资源，撤销该进程的PCB
- **进程阻塞**
  - 停止运行
  - 将PCB从运行态改为阻塞态
  - 插入对应的阻塞队列
  - 转调度程序
- **进程唤醒**

原语：由若干指令构成的具有特定功能的函数，具有原子性，其操作不可分割。

WINDOWS进程控制

启动程序

- `system()`
- `WinExec()`
- `ShellExecute()`
- `CreateProcess()`

结束进程

- `ExitProcess()/TerminateProcess()`

Linux进程控制

搞清楚`fork(), getpid(), exec(), wait(), exit(), sleep()`函数

写时复制

线程

线程是进程内的一个执行路径；多个线程共享CPU可实现并发运行

适用场景

1. 程序的多个功能需要并发运行
2. 提高窗口程序的交互性
3. 需要改善程序结构的地方
4. 多核CPU上的应用，需要充分发挥多核性能

WINDOWS:`CreateThread()`

LINUX:`pthread_create()`

进程相互制约关系

进程的**互斥关系**：多个进程由于共享具有独占性的资源，必须协调各进程对资源的存取顺序：确保没有任何两个或以上的进程同时进行资源存取。

临界资源：一次只允许一个进程独占访问的资源，具有"排他性"

临界区：进程中访问**临界资源**的**程序段**，并发程序不能同时进入

硬件方法实现：

- 中断屏蔽方法（关中断 临界区 开中断）
- 测试并设置指令
- 交换指令

详细解释

软件方法：

- 锁
- 信号量

进程的**同步关系**：合作进程中某些操作之间需要满足某种先后关系或某个操作能否进行需要某个前提条件满足，否则只能等待。

$互斥关系 \subseteq 同步关系$

同步机制❗❗❗

- 锁机制

  初始化锁状态S=1（可用）

  上锁`Lock(S)`，开锁`unLock(S)`，均为原语

  临界区访问的四个原则

  - 忙则等待
  - 空闲让进
  - 有限等待：进程进入临界区的请求应在**有限时间**内得到满足
  - 让权等待：等待的进程放弃CPU，让其它进程有机会使用CPU，**锁机制不满足这条**

- 信号量与P-V操作

  信号灯定义为二元矢量(S, q)

  - S(信号量)：初值非负，**S的初值很重要**
  - q：队列（进程PCB集合），初值为空集

  P操作：`P(S, q)` 简记为 `P(S)`

  1. S值减一
  2. 判断S<0，若S≥0，进程继续；若S<0，进程阻塞并加入到q队列，转调度函数，调度其它进程

  所以P操作会使进程在调用处**阻塞**

  V操作：`V(S, q)`

  1. S值加一
  2. 判断S<=0，若S>0，进程继续；若S≤0，进程继续，同时从q中**唤醒**另一个进程

同步机制实质：运行条件不满足时，能让当前进程暂停，运行条件满足时能让进程立即继续

即在**关键操作之前执行P操作，在关键操作之后执行V操作**

重要模型

1. 司机售票员模型
2. 生产者消费者模型
3. 读者写者模型

WINDOWS进程同步机制：

- 临界区(在进程内使用，保证仅一个线程可以申请到该对象)

  `CRETICAL_SECTION`

- 互斥量(可以跨进程使用，速度慢)

  `CreateMutex/OpenMutex/ReleaseMutex/DleteMutex`

- 信号量（多个线程/进程访问临界区）

  `CreateSemaphore/ReleaseSemaphore/OpenSemaphore`

- 事件
- 等待操作

  `WaitForSingleObject/WaitForMultipleObjects`

5.资源分配与调度

哲学家问题模型

解决方案：限制最多4位用餐，或者要拿就一次拿两双，不满足就释放。

死锁

定义

在两个或多个进程中，每个进程都持有某种资源，但又继续申请其它进程已持有的某种资源。此时每个进程都拥有其运行的一部分资源，但是又都不够，从而每个进程都不能向前推进，陷于阻塞状态。

死锁起因

- 系统资源有限
- 并发进程的推进顺序不当

必要条件

- 互斥条件
- 不剥夺条件
- 部分分配条件：进程所需资源逐步分配，需要时申请和分配
- 环路条件：环中每个进程已占有的资源被前一进程所申请，而自己所需新资源又被环中后一进程所占有。

解决策略

- 预防死锁
- 避免死锁
- 检查和恢复死锁：撤销或挂起一些占有资源少的进程，以回收一些资源

预先静态分配法：破坏部分分配条件

有序资源分配法：破坏环路条件

- 系统中的每个资源被分配有一个唯一序号
- 进程每次申请资源只能申请序号更大的资源

6.进程调度

调度定义

在一个队列中，按某种策略选择一个最合适个体。

进程调度的目标

衡量指标

周转时间：进程提交给计算机到完成所花费的时间t

$t = t_c - t_s$，即进程完成时间 - 进程提交时间

平均周转时间：

$$t = (t_1 + t_1 + \ldots + t_n)/n$$

平均周转时间越短，系统吞吐量也就越大，资源利用率也越高。

带权周转时间w：$w = t/t_r$，t_r为进程运行时间，表示进程在系统中的相对停留时间。

平均带权周转时间

进程调度算法❗

1. **先来先服务**：效率不高，没有考虑运行时间的长短，**不利于短作业**
2. **短作业优先调度算法**：选取时间最短的作业投入运行，忽视作业等待时间，易出现资源“**饥饿**”现象
3. **响应比高者优先调度算法**：

   响应比 = 响应时间（等待时间 + 运行时间）/ 运行时间 = 带权周转时间

   响应比高者优先投入运行

有利于短作业和等候已久的作业，兼顾长作业

4. **优先数调度算法**：把CPU分配给**进程优先数**最高的进程

   进程优先数 = 静态优先数(创建时根据进程属性确定，不变) + 动态优先数（运行期间特定事件发生引起改变）

5. **循环轮转调度算法**：循环队列（逻辑上），以时间片为单位轮流使用CPU，使用后的进程加到队列尾。

调度方式

对于**正在运行**的进程的处理方式：非剥夺方式(不可抢占)/剥夺方式(可抢占)

Linux进程调度

基于优先级，支持普通进程和实时进程，实时进程优先级显著高，普通进程公平使用CPU

`task_struct`的相关字段

- `policy`

  普通进程`SCHED_OTHER(0)`，实时进程：`SCHED_FIFO(0), SCHED_RR(2)`

- `priority`

  进程的静态优先级

- `rt_priority`

  实时进程特有的优先级：`rt_priority+1000`

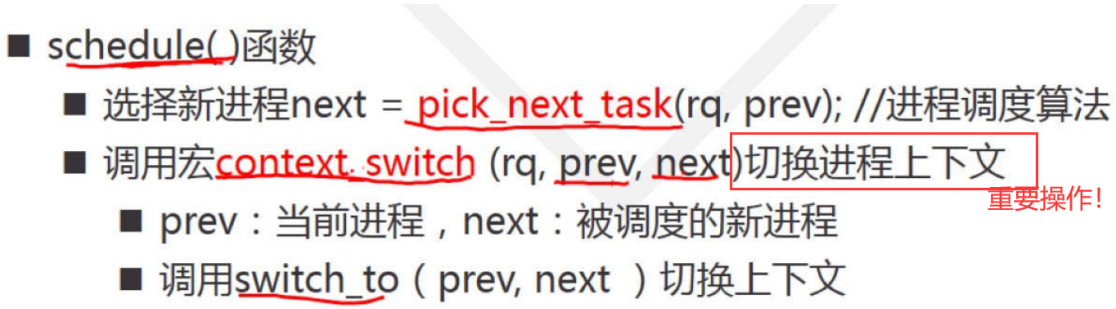
- `counter`

  进程能连续运行的时间片数量，被看作**动态优先级**，初值等于`priority`，时钟中断tick，`counter--`。

  新建子进程`counter`继承父进程的**一半**，防止无限制创建子进程而长期占用CPU

进程切换

内核挂起当前CPU上的进程并恢复之前挂起的某个的某个进程。**进程上下文**包含进程执行的所有信息。



4-6章测试整理

2　关于进程状态说法错误的是 。
√A. 单CPU的系统中处于运行态的进程可以有多个。
B. 进程在整个生存期间会根据不同条件转换状态。
C. 阻塞态的进程即便给它CPU它也无法运行。
D. 处于就绪态的进程都在等待CPU。

> A. 运行态它就是占用CPU的进程，像申请了I/O设备的就是阻塞态了。只能有一个

5　关于进程控制说法错误的是 。
A. 进程生存期间都受操作系统控制。
B. 进程控制采用原语实现。
C. 进程被唤醒的条件和被阻塞的原因一致。
√D. 进程被撤销时操作系统收回其占用资源，但是不释放相应的PCB。

> D. 根据进程撤销原语，撤销的时候是要释放相应的PCB的。
> C选项可以理解成：解铃还须系铃人。

23　在Winodws7中，进程是CPU的调度单位。
A. 对
√B. 错

> 线程。

25　阻塞的进程获得相应服务或信号后会立即开始运行。
A. 对
√B. 错

> 先转为就绪态。

7.主存管理

内存管理功能

实际存储体系

- 三级存储体系
- Cache(快，小，贵) + 内存(适中) + 辅存(慢，大，廉)

存储器功能需求

- 容量足够大
- 速度足够快
- 信息永久保存
- 多道程序并行：共享和保护

存储管理功能

1. **地址映射**：

   定义：把**程序**中的地址（虚拟地址/虚地址/逻辑地址）变换成**内存**的**真实地址**（实地址/物理地址）的过程

   地址重定位，地址重映射

   方式：

   - **固定地址映射**

     编程或编译时就确定逻辑地址和物理地址的映射关系

     特点：程序加载时必须放在指定区域；易产生地址冲突，运行失败；不适应多道环境

   - **静态地址映射**

     程序**装入时**由操作系统完成逻辑地址到物理地址的映射

     保证在运行之前所有地址都绑定到主存

     基址寄存器(重定位寄存器BAR)

物理地址(MA) = 装入基址(BA) + 逻辑地址(VA)

上述两种方式的特点：程序运行之前确定映射关系;程序装入后不能移动(移动也必须要放回原位置);程序占用**连续**的内存空间。

- **动态地址映射**

  在程序执行过程中把逻辑地址转换为物理地址

  MA = BA + VA

  特点：程序占用的内存空间可动态变化；程序不要求占用连续的内存空间（需记录每个基址BA）；便于多个进程共享代码（共享代码单独一段存放）

  缺点：需要硬件支持（MMU，内存管理单元），软件复杂

2. **虚拟存储**

   解决问题：程序过大或过多时，内存不够，不能运行；多个程序并发时地址冲突，不能运行。

   基本原理：

   - 借助**辅存**在逻辑上扩充内存
   - 过程：
     - 迁入：要运行的而部分装入内存，在辅存存放的部分临时**按需调入**内存
     - 迁出：把当前不运行的部分暂时存放在辅存，尽量腾出足够的内存供进程正常运行

   线性地址空间，和物理地址分离（地址无冲突）

3. **内存分配**

   为程序运行分配足够的内存空间

   - 放置策略
   - 调入策略
   - 淘汰策略

4. **存储保护功能**

   - 保证在内存中的多道程序只能在给定的存储区域内活动并互不干扰：**防止访问越界/越权**

   保护方法：

   - 界址寄存器

     设置下限寄存器和上限寄存器

     基址寄存器和限长寄存器

     适用于连续物理分区

   - 存储键保护：适用于不连续的物理分区以及共享中的权限

物理内存管理

分区存储管理

把用户区内存划分为若干**大小不等**的分区，供不同程序使用。适用于单用户单任务

- **固定分区**

  需要维护分区表：分区位置、大小、使用标志。

  浪费内存，大程序可能无法运行

- **动态分区**

  程序装入时创建分区，使分区大小与程序大小相等

  存在**内存碎片**的问题

分区放置策略

空闲区表：描述内存空闲区的位置和大小的数据结构

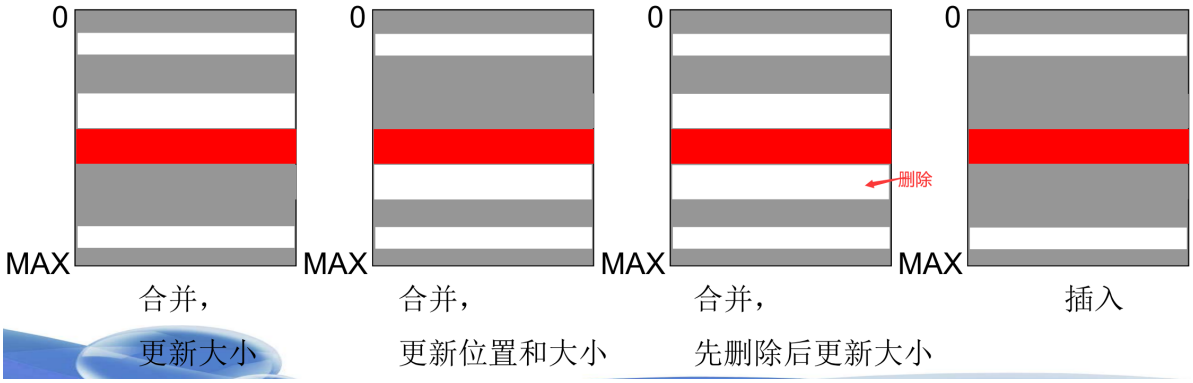
- **首次适应法**：空闲区表按**首址递增**排序，尽可能先利用低地址空间
- **最佳适应法**：空闲区表按大小递增排序，尽量选中满足要求的最小空闲区
- **最坏适应法**：按大小递减排序，尽量使用最大的空闲区，所以只作一次查找就行

分区分配

从用户选中的满足其要求的分区中(底部)分割出所需大小给用户

剩余部分作为空闲区登记在空闲区表中

分区回收



解决内存碎片的方法

- 规定门限值：剩余部分小于门限值，则不再分割，全部分给用户
- 内存拼接技术：技术复杂!

覆盖

程序分成若干代码段或数据段

常用的段装入常驻区（核心段）

不常用的段装入覆盖区：

- 正运行的段处于覆盖区
- 暂时不运行的段放在外存(磁盘)
- 即将运行的段装入覆盖区

缺点：编程复杂，需要程序员划分模块并确定覆盖关系；耗时长，从外存装入内存很耗时

Swapping技术

对换单位过大（整个进程）

换入换出需要地址重定位

虚拟内存管理

程序运行的局部原理

页式虚拟存储管理

把进程空间（虚拟）和内存空间都划分为等大小的小片

进程以**页**为单位装入内存，页在内存中占用的页框不必相邻

内存以**页框**为单位分配使用

页式地址：VA分成页号P和页内偏移W

P = VA/页大小，W = VA%页大小\

或已知页大小为2^n，则$P = VA >> n$；$W = VA\&(2^n - 1)$, 即低n位

页面映射表：记录页与页框之间的对应关系，也叫页表

页面地址映射：虚拟地址->物理地址

1. 从VA中分离出P和W
2. 查页表：以P为索引查页框号P'
3. 物理地址 MA = P' × 页大小 + W

快表机制

快表：页表放在Cache中；慢表：页表放在内存中

特点：

- 容量小，访问快，成本高
- 快表是慢表的部分内容的复制
- 地址映射时优先访问快表
    - 在快表中找到，则为“命中”
    - 没有则需要访问慢表，并更新快表
- 由页表长度寄存器L，分离出的P与L比较，如果P≥L，则地址越界

页面共享

在不同进程的页表中填上相同的页框号(如代码段)，从而都能访问相同的内存空间

- 只有一份真实存储，节省空间

页表建立

- 操作系统**为每一个进程都建立一个页表**
- 页表长度和首址存放在进程控制块PCB中
- 运行进程的页表驻留在内存，页表长度和首址分别由页表长度寄存器和页表首址寄存器指示

页表扩充

- 中断位I，1表示该页不在内存，会引起缺页中断；0表示在内存
- 辅存地址：该页在辅存中的地址
- 访问位
- 修改位

缺页中断

- 在地址映射过程中，当所要访问的目的页不在内存时，则系统产生异常中断即缺页中断
- 产生缺页中断后，程序把所缺的页从页表指出的辅存地址调入内存的某个页框，并更新页表中该页对应的页框号以及修改中断位I为0

比较详细的缺页中断响应过程：

1. 根据访存指令得到虚拟地址及页号P，根据P查询页表
2. 该页不在内存中，发生缺页中断
3. 判断是否有空闲页框，如果没有转4，如果有转5
4. 根据淘汰策略选一页淘汰，如果该页发生修改，则需要重写（即写入外存）
5. 从外存中调入所需页，调整页表，重启中断命令

缺页率 f = 缺页次数 / 访问页面的总次数

淘汰策略

- **最佳算法(OPT)**：淘汰不再需要或最远将来才需要的页面

  理论上最佳（即缺页率最低），但实践中无法实现。

- **先进先出淘汰算法(FIFO)**：淘汰在内存中停留时间最长的页面

  顺序访问时才比较理想，存在异常现象：对于特定的序列，分配的页框越多，缺页率越高

- **最久未使用淘汰算法(LRU)**：淘汰最长时间未被使用的页面

  实现：移位寄存器R，近似算法：页表访问位

- **最不经常使用算法(LFU)**：选择到**当前为止**被访问次数最少的页面

  每页设置访问计数器，页面被访问时，该页面的计数器加1

  发生缺页中断时，选择计数值最小的页面淘汰，并将所有计数器清0

段式和段页式虚拟存储

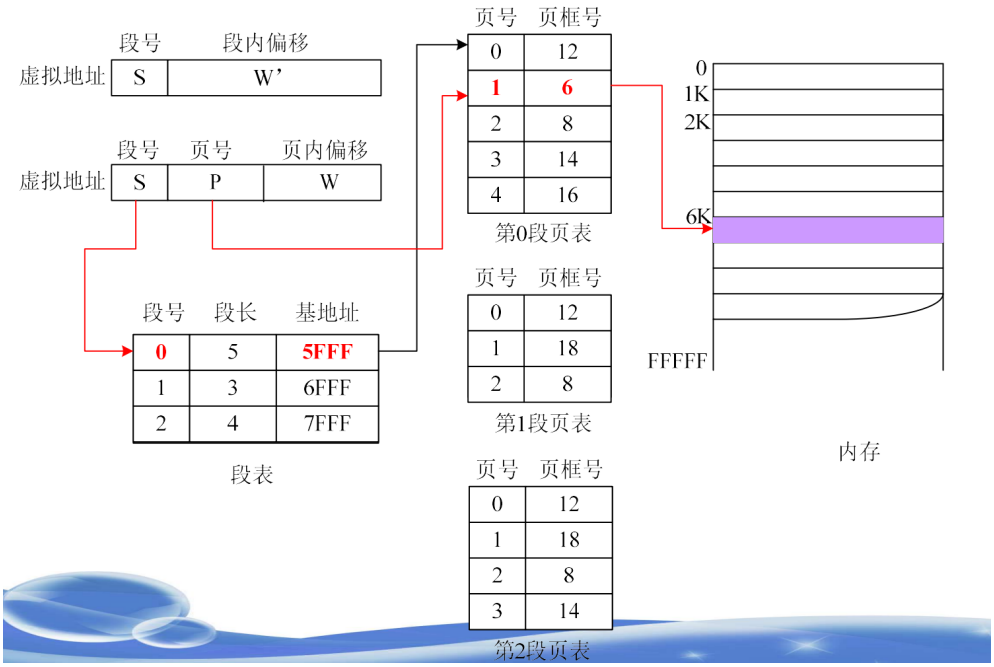
段式地址映射机制：

1. 从VA中分离出段号S和段内偏移W
2. 检索S，查询该段的基地址B和长度L
3. 物理地址MA = B + W

存在偏移量溢出，访问越界检查：0≤W<L

段页式系统地址构成：段号S + 页号P + 页内偏移W

- 系统为每个进程建一个段表
- 每个段建立一个**页表**
- **段表**给出每段的**页表基地址**及**页表长度**（段长）
- 页表给出每页对应的页框



8.设备管理

主要功能

- **设备分配**：是设备管理的基本任务

  设备管理程序按照一定策略为申请设备的进程分配设备，记录设备的使用情况

- **设备映射**：逻辑设备到物理设备的转换 => 逻辑名到物理名的转换

  从应用软件角度，逻辑设备是一类物理设备的抽象

  从设备管理程序角度，物理设备是逻辑设备的实例

  设备独立性

- **设备驱动**：对物理设备进行控制，实现I/O操作；

  操作系统仅对设备驱动的**接口**提出要求，驱动程序由厂商自己根据要求编写

设备分配

- 独占设备

  申请-使用-释放

- 共享设备

  含隐式申请，进程可能会进入阻塞状态

- 虚拟设备：借助虚拟设备，在共享设备上模拟独占型设备

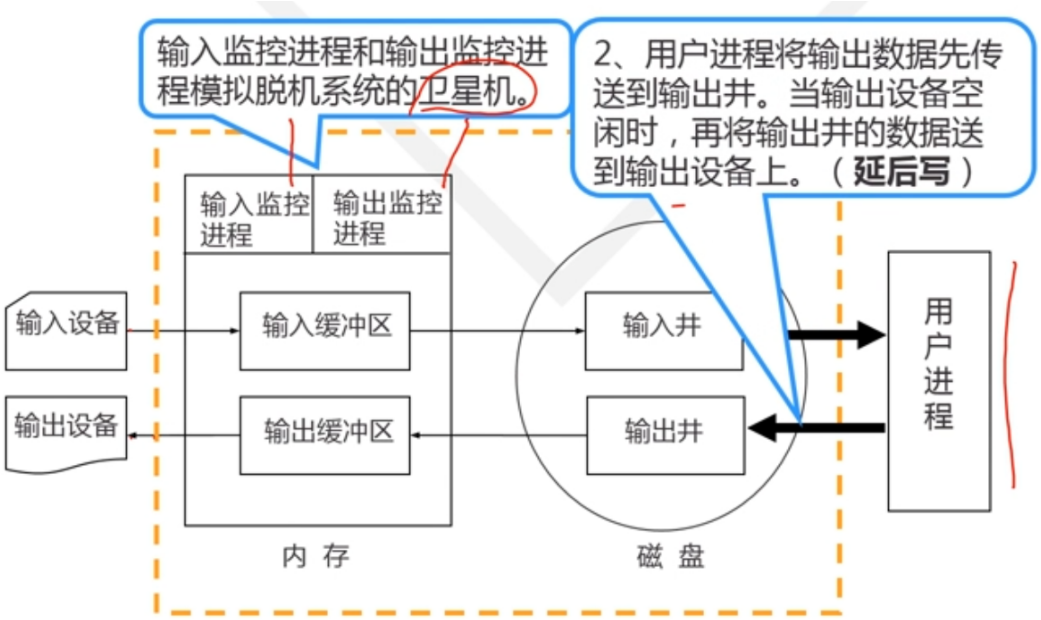
虚拟分配：申请独占设备，实际分配独占设备对应的虚拟设备

虚拟技术：在一类物理设备上模拟另一类物理设备的技术

通常借助**辅存**的一部分区域模拟**独占设备**，将**独占设备**转化为**共享设备**。

SPOOLING系统（假脱机输入输出系统）

- 任务执行前：预先将程序和数据输入到输入井中
- 任务运行时：使用数据时，从输入井直接取出
- 任务运行时：输出数据时，把数据写入输出井
- 任务运行完：仅当外设空闲时，输出全部数据和信息



缓冲技术

1. 连接不同数据传输速度的设备
2. 协调数据记录大小的不一致
3. 正确执行应用程序的语义拷贝

提前读：内核提前从输入设备把数据读入到I/O缓冲区中

延后写：用户处理数据的同时内核输出前一数据

提高效率！减少访问目标设备的次数。

9.文件系统

文件定义：文件是系统中信息存放的一种组织形式。由若干信息项构成

分类

按用途分类：系统文件/库文件/用户文件

按操作权限分类：只读文件，读写文件，不保护文件

按性质分类：普通文件，目录文件，设备文件

文件属性：指定文件的类型、操作特性和存取保护等一组信息。一般存在文件的目录项中

文件的结构

- **逻辑结构**：强调文件信息项的构成方式和用户的存取方式
  - 流式文件
  - 记录式文件

  文件存取方式：
  - 顺序存取
  - 随机存取
- **物理结构**：强调合理利用存储空间，缩短I/O存取时间
  - 连续文件：文件目录记录文件长度和首个物理块号
  - 串联文件：文件目录记录文件名+存储指针（指向第一个存储块）
  - 索引文件：系统建立索引表记录文件的逻辑块和存储块的对应关系，索引表+数据区

文件存储空间管理

记录磁盘空闲块的方法：

- 空闲文件目录：所有的空闲文件代表存储设备的空闲文件
- 空闲块链
- 位示图

文件目录：记录文件名和存放地址的目录表

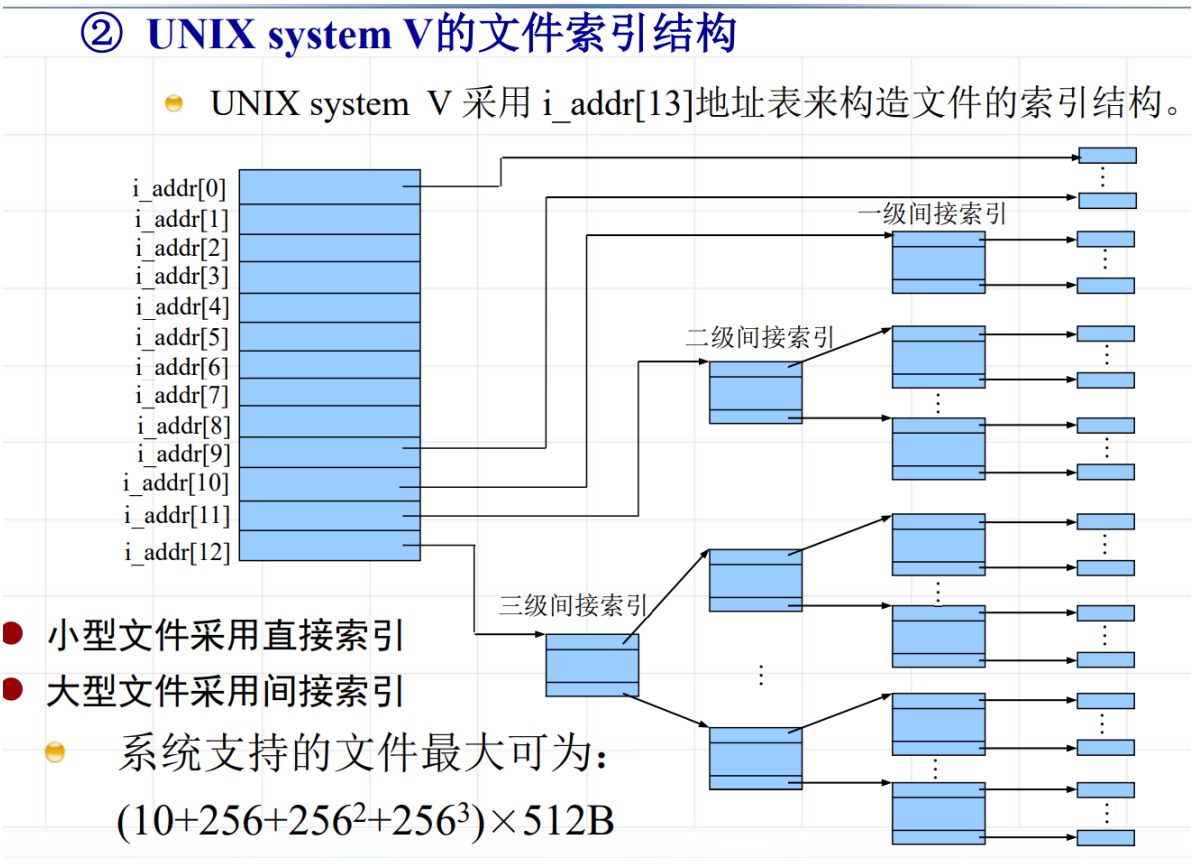
目录文件：文件目录以文件形式存于外存。功能：将文件名转换为外存物理位置的功能

- 单级目录 / 二级目录 / 多级目录(树型结构)

文件全名：包括从根目录开始到文件为止的通路上所有子目录路径和该文件的符合名组成的字符串

绝对路径名 / 相对路径名

UNIX采用多级间接索引(树形结构)：



期末测试及其它试题整理

1　当操作系统处理缺页中断的时候，CPU处在 。
A. 用户态
√B. 核态
C. 不确定的状态
D. 空闲状态

42　多道批处理系统的特点就是把CPU时间分成小片轮流地为多个作业服务。
A. 对
√B. 错

> 分时才多片，多道只是：这个进程不用CPU了就给别的进程用。

44　WINDOWS 10中线程是资源分配和CPU调度的基本单位。

A. 对

√B. 错

> 进程是操作系统资源分配的基本单位，而线程是任务调度和执行的基本单位。好坑哦。
