操作系统的功能

- 负责为应用程序分配和调度软硬件资源
- 进程管理
- 内存管理
- 设备管理
- 文件管理

操作系统逻辑结构

- 操作系统 = 微内核+核外服务器

CPU的态

- CPU态（Mode）：CPU工作状态，对资源，指令的描述
- 核态，全部资源
- 用户态：能访问部分资源，用户程序
- 管态：核态和用户态之间
- 主存和辅存：主存能够直接和CPU交换信息，辅存不能，半导体存储器常做主存
- 实际存储体系：寄存器，高速缓存，主存，辅存。

操作系统启动过程

- 实模式->保护模式
- BIOS 固件（firmware）功能：启动配置，提供基本I/O服务，加电自检和自举
- POST->CMOS设置（硬盘启动）->读取MBR->控制权交给MBR->MBR读取分区表->PBR控制后面的引导过程。
- 初始引导：把OS核心装入内存，并使之开始工作接管计算机系统
    - 最常用的引导程序:grub
- Windows启动过程
    - POST：加电后ＢＩＯＳ启动主机自检程序。
    - 初始引导：BIOS从MBR读入引导程序，引导程序启动DOS7.0,调入操作系统核心。Windows接管系统
    - 核心初始化，系统初始化
- Linux启动过程
    - POST
    - MBR
    - KERNEL映像自解压并执行
    - 内核初始化
    - 内核启动
    - 内核完成引导后，加载init程序，进程号1，init进程通过/etc/inittable脚本进行初始化

系统调用

- 指操作系统给的服务，运行于核态，每个系统调用具有唯一标号。

进程管理

- 进程的特征:动态性，并发性，异步性，独立性（系统分配资源和调度CPU的单位）
- 进程的状态：
    - 运行状态：在运行

    - 就绪状态：具备运行条件但没有CPU运行
    - 阻塞状态：等待指令开始运行
- 进程控制块（PCB）
    - 描述进程状态的数据结构
    - PCB是进程的标志，生命周期随着进程变化
    - 进程 = 程序+PCB
- 原语
    - 由若干指令构成的具有特定功能的函数
    - 具有原子性，不可分割
    - 进程控制原语：创建原语，撤销原语，阻塞原语，唤醒原语
- Linux进程控制
    - pid_t fork()
        - 新进程是当前进程的子进程
        - 子进程与父进程有着相同的代码空间，堆栈，等，还可以并发运行
        - Linux所有进程都是由fork创建（除了init），
        - exec：功能，装入一个指定的可执行程序运行

临界区

- 忙则等待：当临界区忙时，其他进程必须在临界区外等待
- 空闲让进：当无进程处于临界区，任何有权进程可进入临界区
- 有限等待：进程进入临界区的请求必须在有限时间内得到满足
- 让权等待：等待进程放弃CPU

锁机制

- 基本原理：
    - 设置一个“标志”表明是否可用

P—V操作

- S指的是信号量，Q指的是进程队列
- P操作，P(S,Q)
    - S减一，S就是差
    - 若差大于0，该进程继续
    - 若差小鱼0，则该进程阻塞并加入到进程队列里面，转调度函数
    - 进程调用P操作可能会使进程阻塞
- V操作，
    - S值加1，为和
    - 若和大于0，则该进程继续
    - 若和小于等于0，则该进程继续同时从q中唤醒一个进程
    - V操作可能会唤醒被阻塞的进程

linux进程同步机制

- wait函数，
    - 阻塞自己，直到子进程结束，然后手机子进程信息并销毁子进程
- exit函数，
    - 释放资源并报告父进程
    - 利用status传递进程结束的状态
    - 变为僵尸状态，保留部分PCB信息供wait收集

        - 是否正常结束，占用CPU时间，缺页中断次数
    - 调用schedule函数
- 父子进程共享普通变量，但互不影响
- 父子进程共享文件资源，且处理的是同一个文件资源

匿名管道

- 管道是进程间的一种通信机制，一个进程（A)可以通过管道把数据传输给另一个进程，另一个进程从管道读取数据
- 匿名管道是单向的
- 管道需要重定向才能读取

linux信号通信

- 可以通过修改信号处理函数更改程序结束后的

死锁

- 定义:两个或者多个进程无限期等待永远不会发生的一种系统状态
- 死锁的起因：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：进程互斥使用资源，资源具有独占性（破坏难度高）
    - 不剥夺条件：进程在访问完资源前不能被其他进程强行剥夺
    - 部分分配条件：进程边运行边申请条件，临时需要临时分配
    - 环路条件:多个进程构成环路，环中每个进程已占用的资源被前一进程申请，自己申请的资源被后一进程占用
- 银行家算法
- 检测和恢复死锁：实现难度大，恢复方法靠人工，撤销一些进程
- 预先静态分配法：进程运行前一次性把资源全部分配
    - 执行可能延迟
    - 应用开销增大
    - 资源利用率低
- 有序资源分配法
    - 每个资源有唯一一个序号
    - 每次申请资源只能申请序号更大的资源
- Windows和Linux没有采用死锁解决方案（鸵鸟策略）

内存管理功能

- 实际存储器体系
    - CPU
    - Cache（快，小，贵，在CPU内）+内存（适中）+辅存（慢，大，廉）
    - 内存：RAM，就是CPU的容量
- 存储管理的功能
    - 地址映射：地址重定位
        - 固定地址映射：编程或者编译时确定逻辑地址和物理地址的映射关系。容易产生冲突
        - 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映射的过程。
        - 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
    - 虚拟内存
        - 虚拟内存是面向用户的虚拟封闭储存空间：在32位系统容量4G，是线性地址空间。
        - 使得的程序能在较小的内存中运行

        - 使得多个程序能在较小的内存中运行
        - 多个程序并发运行时地址不冲突
        - 内存利用效率高：无碎片，共享方便
    - 内存分配功能
    - 存储保护功能
        - 防止访问越界
        - 防止访问越权
        - 方法：界止寄存器
- 物理内存管理
    - 分区存储管理
        - 单一区存储管理
        - 固定分区，固定的划分为若干大小不等的分区
        - 动态分区
    - 分区的分配
        - 从空闲区表的第一个区开始，寻找符合要求的空闲区
        - 分割空闲区一般从底部分割，保证顶部的寻址地址不变，便于更新空闲区表
        - 空闲区表如何排序：
            - 放置策略，
            - 按空闲区位置（地址）递增排序，按空闲区
            - 最佳适应法，空闲区表按大小递增排序，以求尽可能先使用较小的空闲区，保留大的
            - 最快适应法：能够最快的找到合适的空闲区，
    - 覆盖——overlay
        - 目的，在较小的内存空间中运行较大的程序
        - 内存分区：常驻区，覆盖区（可以被多段程序利用）
        - 效率低，从外存装入内存耗时
    - 交换技术——Swapping
        - 内存不够时把进程写到磁盘
        - 当进程要运行时重新协会内存
    - 内存碎片
        - 过小的空闲区无法处理
        - 动态分区的缺点：内存的反复分配和分割，产生内存碎片
        - 解决方法：
            - 规定门限：若剩下的部分小于门限，则不作分割
            - 内存拼接技术：
            - 解除程序占用连续内存才能运行的限制
- 虚拟内存管理
    - 大的程序能在小的内存中运行
    - 多个程序在较小的内存中运行
    - 多个程序并发时地址不冲突
    - 提高利用率，无碎片内存
- 页式内存管理
    - 把进程空间（虚拟）和内存空间划成等大小的小片
        - 进程小片：页
        - 内存小片：页框
    - 页表

        - 虚拟（VA）地址可以分解为页号P和页内便宜W
        - 页号 = VA/页的大小，偏移 = VA%页的大小
        - 页面映射表：记录页与页框之间的对应关系。也叫页表
    - 页式地址映射
        - 功能：虚拟地址->物理地址
        - 过程：分离P和W，查页表，计算物理地址MA
- 快表（Cache）
    - 慢表：页表放在内存中
    - 快表：页表放在Cache中
    - 块表的特点：
        - 容量小，访问快，成本高
        - 块表是慢表中部分内容的复制
        - 地址映射优先访问块表：若在块表中找到所需数据，则称为“命中”，没有命中，则访问慢表并更新快表
- 页面的共享
    - 一个程序分为代码段和数据段，代码段可以共享
    - 在不同进程的页表中天上相同的页框号，多个进程能访问相同的内存空间，从而实现页面共享
    - 共享页面在内存中只有一份真实储存，节省内存
- 缺页中断
    - 页表扩充——带中断位的页表
    - 页表扩充——带访问位和修改位的页表
    - 概念：当所要访问的目的页不在内存中时，则系统产生异常中断——缺页中断
    - 缺页中断处理程序：中断处理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中，并更新页表中该页对应的页框号以及修改中断位I为0
- 页面淘汰策略
    - 缺页中断时装入新的页面时要淘汰页
    - 页面抖动：页面在内存和辅存间频繁交换的现象，是IO操作，会降低系统效率
    - OPT算法（最佳算法）：淘汰不再需要或最远的将来才会用到的页面
    - FIFO算法（先进先出淘汰算法）
    - LRU算法（淘汰最长时间未被使用的页面）
    - LFU算法（最不经常使用算法）
- 缺页因素
    - 淘汰算法
    - 页框越少，越容易缺页
    - 页面大小：页面太大：浪费内存。页面太小：页表长度增加，浪费内存，同时换页频繁，浪费效率
- 进程分段
    - 进程按逻辑意义分为多个段，每段有段名，长度不定，进程有多段组成
    - 段式内存管理系统的内存分配
        - 以段为单位装入，每段分配连续的内存，段和段之间不要求相邻
        - 段式虚拟内存VA包含段号S和段内偏移W
    - 段表（SMT）：记录段，内存，地址的映射关系
    - 段的优点：页面不方便共享，段方便共享
    - 段页式存储管理
        - 在段中划分页面——段号-页号-页内偏移