计算机操作系统

- **初识操作系统**
  - 操作系统概念和功能
    - 包装硬件接口
    - 美化接口供上层用户与应用程序调用
      - GUI调用
      - 用户交互
        - 交互式指令-cmd小黑窗
        - 批量式交互-脚本交互
      - 程序交互
        - 通过程序语言编写程序
    - 统筹管理硬件（是硬件功能的扩展）
    - 分配软件资源和硬件资源
      - 硬件管理
      - 文件管理
      - 内存分配
      - 进程管理?
  - 操作系统特征
    - 并发性
      - 并发与并行的区别
    - 共享性
      - 并发与共享性不可分割
    - 虚拟性
      - 时分复用技术
        - 微分时刻
      - 空分复用技术
        - 微分CPU
    - 异步性

- 资源共享占用时的阻塞，程序的运行是走走停停的

- 操作系统的发展与分类
  - 手动打孔输入
  - 单道处理系统
  - 单道批处理系统（操作系统诞生）
  - 多道批处理系统
- 操作系统运行机制
  - 两种指令
    - 特权指令
      - 寄存器内，psw==1
    - 非特权指令
      - 用寄存器内二进制数的某位是0还是1区分
  - 两种处理状态
    - 核心态
      - 转变到用户态时主动让出权限
    - 用户态
      - 变为核心态时，发出“中断信号”，核心态抢回权限，硬件自发
        - 中断
        - 异常
  - 两种程序
    - 内核程序
      - 组成操作系统内核
    - 应用程序
- 异常与中断
  - （内中断）异常
    - 陷入（trap指令）
      - 主动交出cpu控制权
    - 故障fault
      - 程序应用的故障，例如0/0错误

- 终止abort
  - 系统级别的异常
- （外中断）中断
  - 时钟信号
  - I/O信号中断

- 系统调用
  - 系统调用是系统对程序和程序员提供应用的接口
  - 库函数与系统调用的区别（c语言库函数）
    - 部分库函数需要系统调用
    - 库函数是高级语言对系统调用的进一步封装
    - 部分库函数不需要系统调用
  - 系统调用可以限制某些资源不会被同时访问，进而引发问题
  - 系统调用过程
    - 向核心态传参
    - 接受中断trap指令
    - 内核程序处理系统调用请求
    - 返回用户态
- 操作系统结构
  - 大内核/宏内核
    - 微内核
    - 进程管理
    - 存储管理
    - 内存分配
  - 微内核
    - 时钟管理
    - 中断处理
    - 原语
  - 层次结构
    - 优点：易于扩展、调试和维护

- 缺点：调用时不灵活，难以定义层次边界（互相调用）
- 模块化
  - 优点：支持动态加载，模块间通信效率高
  - 缺点：难以定义接口，调试性差
- 外核
- 虚拟机
  - VMM直接与硬件接触
  - VMM与宿主操作系统接触

分支主题 2

分支主题 3

第二章

- 进程
  - 概念
    - 进程是操作系统分配资源和硬件的基本单位
    - 程序是静态的二进制指令，进程实体是程序运行过程中某一时刻的状态
  - 组成
    - PCB（OS中管理进程的一种数据结构）
      - 进程描述信息PID
      - 控制和管理信息
      - 资源分配信息（内存大小等
      - 处理机相关信息
    - 程序段
      - 指令序列（程序代码）
    - 数据段
      - 运行过程中产生的数据（局部变量等）
  - 特征
    - 动态性
      - 有生命周期，动态产生、变化、消亡
    - 并发性

- 各个进程可以并发执行
- 独立性
  - 独立运行、独立获得资源、独立接受调度
- 异步性
- 结构性
  - PCB结构实体

- 进程生命周期
  - 状态
    - 创建
    - 就绪
    - 运行
    - 阻塞
    - 终止
  - 状态转换
    - 创建好后等待cpu处理
    - CUP处理时遇到时间片结束/等待分配资源-->阻塞
    - 等到处理时间/需要的资源空闲-->就绪运行
    - 到CUP上处理运行
    - 程序故障终止/程序指令结束-->消亡
- 进程组织方式
  - 链接方式
  - 索引方式
- 进程控制
  - 状态切换原语
    - 进程创建原语
    - 进程终止原语
    - 进程唤醒原语
    - 进程阻塞原语
    - 进程切换原语

- 就绪态-->运行态
- 原语基本事件
  - 1. 更新PCB状态信息（status字段）
  - 2. 将PCB索引放入适合的队列
  - 3. 分配/回收资源（更改运行环境）
- 引起进程状态变化事件

进程通信

- 共享存储
  - 设置共享内存区域，互斥访问共享空间
  - 两种方式
    - 共享内存通信
      - 操作系统分配共享内存
      - 进程自行决定如何操作内存
    - 共享数据结构通信
      - 只能由单一数据结构存储、读取进而共享数据
- 消息传递
  - 传递结构化的消息（消息头信息/消息体）
  - 系统提供“发送/接受”原语
  - 两种方式
    - 直接传递
    - 简介通信（邮箱）
- 管道通信（一种方式）
  - 先进先出FIFO（循环队列）
  - 进程读取或写入时，阻塞其他进程；空时读进程堵塞，满时写进程堵塞
  - 读写都可以多进程，但是要互斥
  - 一个管道只可进行半双工通信
  - 设置一个特殊的共享文件--内存缓冲区

线程

- 线程属性

- 同一个线程下的进程共享同一内存空间，可以互相通信
- 线程几乎不拥有系统资源
- 线程拥有自己的线程控制块（TCB）和数据结构
- 一个进程可以拥有多个线程
- 线程也有就绪、运行、阻塞态
- 线程是处理机调度的基本单位
- 线程可以占用多个CPU
- 同一进程内的线程切换不会引起线程切换
- 同一进程内的线程切换，资源调度消耗小

- 引入线程优势
  - 资源分配、调度
    - 线程作为资源调度的基本单位
    - 进程分配管理资源
  - 并发性
    - 线程间也能并发，提高了系统的并发度
  - 系统开销
    - 线程切换系统开销小，线程级并发切换时不需要切换运行环境
- 线程的实现
  - 实现方式
    - 用户级线程
      - 编码级别的线程--线程库
      - 特点
        - 线程的管理由应用程序负责
        - 线程间切换不需要转换状态，在用户态下转换
        - *操作系统分配和调度的基本单位还是线程
        - 进程间线程一旦有一个阻塞，其余线程也被阻塞，系统并发度不高
    - 内核级线程
      - 操作系统内核支持的线程
      - 特点

- 线程管理和切换由操作系统负责
- 线程间切换需要进行从用户态到核心态的转变
- 各线程分别被操作系统管理，并发度高
- 操作系统会为每个线程分配TCB，对内核级线程进行控制

- 理解
  - 用户级线程是“代码实体”
  - 内核级线程是“运行实体”
  - 只有获得了运行机会的内核级线程才是处理机分配的单位

- 多模型（用户级映射到内核级）
  - 一对一
    - 一个用户级线程映射到一个内核级线程
    - 优缺点
      - 当一个线程被阻塞后另一个线程不受影响，系统并发度高
      - 浪费资源，开销太大（切换线程时）
  - 多对一
    - 多个用户级线程对应一个内核级线程
    - 优缺点
      - 线程切换只需要在用户级下进行
      - 一个线程阻塞其余线程同样被阻塞，并发度低
  - 多对多
    - N个用户级线程对应M个内核级线程（N>=M）
    - 优缺点
      - 其中一个内核级线程阻塞，另外的内核级线程还可以继续运行，解决了多对一模型并发度低的问题
      - 有时候级线程的切换只需要在用户态下进行（由程序负责），解决了切换到内核态开销大的问题

- 线程的状态
  - 运行
  - 就绪
  - 阻塞

- 线程的组织与控制
  - 线程表
    - 多个线程组成线程表
  - 线程控制块
    - 线程标识符（TID）
    - 程序计数器PC（线程目前进行到哪里）
    - 其他寄存器（中间运行的结果）
    - 堆栈指针
      - 堆栈保存函数调用信息、局部变量
    - 线程运行状态（运行/就绪/阻塞）
    - 线程优先级
    - （当线程切换时2，3，4要保存/恢复）
- 调度的概念以及层次
  - 概念：操作系统根据某种特定的算法决定任务的执行顺序
  - 层次
    - 高级调度/作业调度
      - 事件：把程序从外存中加载到内存中
      - 频率：低
      - 一个作业只会发生一次调入，一次调出，当调入时程序会为其创建PCB，调出时销毁PCB
      - 无->创建态->就绪态
    - 中级调度/内存调度
      - 事件：当CPU内存告急时可以挑选一些空闲的任务下处理机，调回外存中保存，变成挂起状态
      - 频率：中等
      - 挂起态变回其他状态
    - 低级调度/进程调度
      - 事件：当线程（进程）处于就绪态时需要调回处理机处理
      - 频率：最高

- 操作系统需要频发的切换在CPU上工作的任务，宏观看起来作业像是在同时执行的一样
- 就绪态->运行态

- 进程调度的时机
  - 需要进程调度
    - 进程主动放弃处理机
      - 进程执行结束
      - 遇到异常
      - I/O阻塞
    - 进程被动放弃处理机
      - 时间切片结束
      - 有更高优先级的进程进入就绪队列
      - 有更紧急的事需要处理（I/O中断）
  - 不能进行调度
    - 进行原语操作时
    - 处理中断操作过程中（中断信号，与硬件相关，不可中断
    - 在操作系统内核程序临界区
      - "操作系统内核程序临界区"与"临界区"的区别
        - 临界区内的临界资源（例如打印机）使用时要上锁，但此时进程调度是允许的（允许其他进程进行）
        - 内核程序临界区内的临界资源在使用时，其他线程不会被指挥调度（不可以同时运作），否则会影响到操作系统内核的其他管理工作
      - 内核程序临界区：一般用来访问某种内核数据结构的，例如进程的就绪队列（PCB就绪队列）

- 进程调度方式
  - 非剥夺调度
    - 只允许进程主动放弃处理机
    - 开销小但无法及时处理紧急任务
  - 剥夺调度
    - 当有更紧急的任务出现时，立即暂停当前任务让给更紧迫的进程

- "广义进程调度"

- 包含了"侠义进程调度"
  - 从就绪队列中选出一个进程
- 进程切换
  - 把选中的进程换上处理机
    - 包含的步骤
      - 1. 对原来的进程进行数据保存
      - 2. 对新的进程进行数据的恢复
    - 进程的切换需要的代价比较大

- 线程状态七模型
  - 创建态
  - 就绪态
    - 就绪状态下被挂起：就绪挂起
  - 挂起
    - 就绪挂起
      - 激活时变为：就绪态
    - 阻塞挂起
      - 当唤醒事件出现但仍处于挂起态时会变为：就绪挂起；未出现唤醒事件时激活会返回：阻塞态
  - 运行态
    - 也可以直接进入：就绪挂起态
  - 阻塞态
    - 阻塞状态下被挂起：阻塞挂起
  - 终止态