2.1进程与线程（中）

进程的组织

- 进程是一个独立的运行单位，也是操作系统进行资源分配和调度基本单位，由以下三部分构成
- 进程控制块
  - 进程描述信息
    - 进程标识符：标志进程
    - 用户标识符：进程归属的用户，主要为共享和保护服务
  - 进程控制和管理信息
    - 进程当前状态：描述进程状态信息
    - 进程优先级：描述进程枪战处理机优先级
    - 代码运行入口地址
    - 程序的外存地址
    - 进入内存时间
    - 处理机占用时间
    - 信号量使用
  - 资源分配清单
    - 用以说明有关内存地址空间或者虚拟地址空间状况，所打开的文件的列表和所使用的的输入/输出设备信息
    - 代码段指针、数据段指针、堆栈段指针、文件描述符、键盘、鼠标
  - 处理机相关信息
    - 处理机中各寄存器的值
    - 通用寄存器值、地址寄存器值、控制寄存器值，标志寄存器值，状态字
- 程序段：能被进程调度程序调度到CPU执行的程序代码段
- 数据段：进程对应的程序加工处理的原始数据或者程序执行时产生的中间或者最终结果
- 进程的组织方式
  - 链接方式
    - 按照进程状态将PCB分为多个队列
    - 操作系统持有指向各个队列的指针
  - 索引方式
    - 根据进程状态的不同,建立几张索引表
    - 操作系统持有指向各个索引表的指针

进程的通信

- 共享存储
  - 通信进程之间存在一块可以被直接访问的共享空间
  - 低级方式：基于数据结构共享
  - 高级方式：基于存储区共享

操作系统只负责为通信进程提供可共享使用的存储空间和同步互斥工具，数据交换则由用户自己安排读/写指令完成

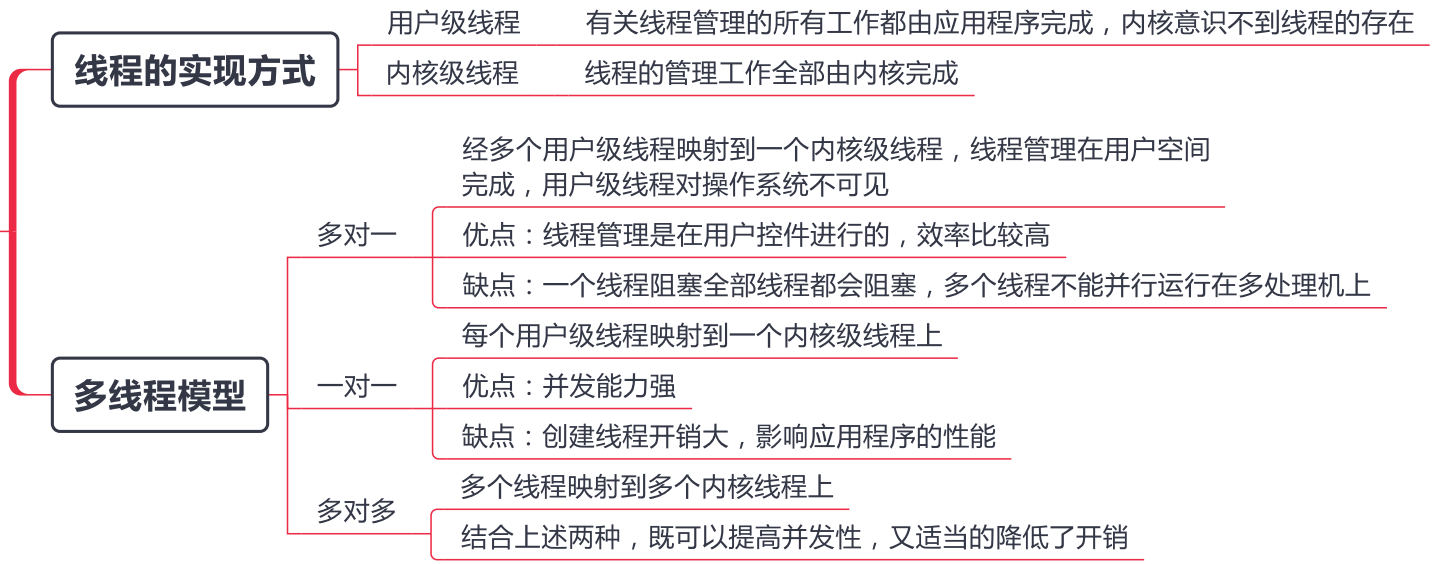
- 消息传递
  - 进程间的数据交换是以格式化的消息为单位的，进程通过系统提供的发送消息和接收消息的两个原语进行数据交换
  - 直接通信方式：发送进程直接发送消息给接收进程，并将它挂在接收进程的消息缓冲队列上，接收进程从消息缓冲队列中取得消息
  - 间接通信方式：发送进程把消息发送给某个中间实体，接收进程从中间实体中获得消息　　例如电子邮件系统
- 管道通信
  - 发送进程以字符流形式将大量数据送入写管道，接收进程从管道中接收数据
    - 当管道写满时，写进程的write()系统调用将被阻塞，等待读进程将数据取走
    - 当读进程将数据全部取走后，管道变空，此时读进程的read()系统调用将被阻塞
  - 功能：互斥、同步、确定对方存在
  - 半双工通信，不可以同时读和写
  - 限制管道的大小
  - 管道变空的时候阻塞读进程

管道中的数据被读取后会马上被丢弃

线程概念和多线程模型

- 线程基本概念
  - 减小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
  - 引入线程后，进程只作为系统资源的分配单元，线程作为处理机的分配单元
- 线程与进程的比较
  - 调度
    - 传统中进程是资源和独立调度的基本单位
    - 引入线程后，进程是独立调度的基本单位，线程是资源的基本单位
    - 不同进程的线程切换会引起进程切换
  - 拥有资源　　进程是资源分配的基本单位
  - 并发性　　引入线程后，进程可以并发执行，多个线程之间也可以并发执行，提高了系统的吞吐量
  - 系统开销　　同一进程的线程切换要比进程切换开销小的多
  - 地址空间和其他资源　　进程的地址空间之间相互独立，统一进程的各线程之间共享进程的资源，某进程的线程对其他进程不可见
  - 通信方面
    - 进程间通信需要进程同步和互斥手段的辅助，保证数据的一致性
    - 线程间可以直接读/写进程程序段来进行通信
- 线程属性
  - 不拥有系统资源，拥有唯一标识符和线程控制块
  - 不同的线程可以执行相同的程序，同一个服务程序被不同用户调用时，操作系统将其创建为不同线程
  - 统一进程的线程共享该进程拥有的全部资源
  - 线程是处理机的独立调度单位
  - 线程也有生命周期，阻塞，就绪，运行等状态
  - 多CPU计算机中,各个线程可占用不同的CPU
  - 每个线程都有一个线程ID、线程控制块（TCB）
  - 切换同进程内的线程,系统开销很小
  - 切换进程,系统开销较大
  - 由于共享内存地址空间,同一进程中的线程间通信甚至无需系统干预

2.1进程与线程（下）


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
线程的管理工作全部由内核完成
多线程模型
多对一
经多个用户级线程映射到一个内核级线程，线程管理在用户空间完成，用户级线程对操作系统不可见
优点：线程管理是在用户控件进行的，效率比较高
缺点：一个线程阻塞全部线程都会阻塞，多个线程不能并行运行在多处理机上
一对一
每个用户级线程映射到一个内核级线程上
优点：并发能力强
缺点：创建线程开销大，影响应用程序的性能
多对多
多个线程映射到多个内核线程上
结合上述两种，既可以提高并发性，又适当的降低了开销