- 进程（Process）和线程（Thread）的各种状态转换
- 系统调用（System call）
- 进程控制块（Process Control Block）
- 多道程序设计（Multiprogramming）
  - 上下文切换
  - CPU利用率
- 线程控制块（Thread Control Block）

Orphans孤儿进程：父已终止仍运行，被init收养

Zombie僵尸进程：父未调用wait()或waitpid()时子进程exit退出

> 子进程(init除外)在exit()之后，并非马上就消失掉，而是留下一个称为僵尸进程(Zombie)的数据结构，等待父进程处理

不要kill init，会出事

一、多道程序设计

- 允许多程序同时进入内存，共享系统各种硬软件资源，在CPU中交替运行（不等IO）
- 各进程在一段时间内并发运行，CPU和IO并行
- 优点：提高资源利用率和吞吐量
- 缺点：用户响应时间长，没有人机交互性
- 特点：多道，宏观上并行，微观上串行

上下文切换

只发生在内核态

切换CPU到另一个进程时，需要保存当前进程状态到PCB并恢复另一个进程状态

（上下文：某一时刻CPU寄存器和PC内容)

切换处理及时两对操作：

- 当前进程上下文保存到PCB，装入分派程序上下文开始运行
- 移出分派程序上下文，新进程的CPU现场信息装入处理器各个寄存器

硬件减少开销方法：

两组寄存器，一组内核用，一组用户用。上下文切换时只需要改变一个指针，指向当前寄存器组

CPU利用率

二、系统调用

指用户程序调用操作系统提供的子功能，系统调用可视为特殊的公共子程序

把应用程序的请求传给内核，调用相应的内核函数完成所需的处理，将处理结果返回给应用程序

- 类别：设备管理、文件管理、进程控制、内存管理
- 目的：用户程序不能直接执行对系统影响大的操作，必须通过系统调用请求操作系统代为执行，保证系统稳定和安全，防止用户程序随意更改或访问重要系统资源、影响其他进程运行
- 用户态进入内核态，用户堆栈也切换为系统堆栈，但还是属于该进程

三、进程线程状态转换

- 创建时机：用户登录、程序执行、~~(设备分配)~~
- 创建过程：申请空白PCB并填写+分配运行资源（内存、文件、IO、CPU时间）+转入就绪态
- 阻塞原因：等IO、等待非CPU资源（申请内存失败）、执行P（wait）操作（原因不同可有多个阻塞队列）
- 阻塞是进程自身主动的行为，唤醒是针对阻塞的
- 线程状态：执行、就绪、阻塞

四、进程控制块

为了使并发的每个程序都能独立运行，必须配置一个专门的数据结构PCB

- 进程实体（映像）：PCB，数据段，程序段
- 创建、撤销进程的都是PCB，是进程存在的唯一标志
- 组织各进程PCB方法：链接、索引

5、什么是进程控制块（PCB），它有哪些主要内容?

定义：每个进程在进程表中的表项

主要内容：包含了进程状态的重要信息：程序计数器、堆栈指针、内存分配状况、所打开文件的状态、账号和调度信息，以及其他在进程由运行态转换到就绪态或阻塞态时必须保存的信息

作用：保证该进程随后能再次启动，就像从未被中断过一样。

五、线程控制块

- 记录寄存器和栈等现场
- 线程标识符+一组寄存器（PC+状态+通用）+线程状态+优先级+线程切换时专用于保护现场的存储区+堆栈指针（局部变量，返回地址）
- 所有线程共享进程的地址空间和全局变量（甚至可以读写另一个线程的堆栈）
- 同进程创建的线程可共享：代码段、打开的文件、栈指针、~~（全局变量）~~

T：fork

printf函数，但是没有换行的话是不会输出的，而是先将要输出的内容存放再缓冲区中，当碰到换行时，将缓冲区中的内容再一起输出

- fork后子进程会复制父进程的缓冲区

(9条消息) 操作系统---fork函数解析与例题详解_IT_xiaoye的博客-CSDN博客