操作系统八股

内存

虚拟内存和物理内存

1. 操作系统提供一种机制，其并不像单片机一样直接进行烧录到物理内存
2. 操作系统会提供一种机制，将不同进程的虚拟地址和不同内存的物理地址映射起来
3. 操作系统引入了虚拟内存，进程持有的虚拟地址会通过 CPU 芯片的内存管理单元的映射关系，来转换位物理地址，再通过物理地址访问内存
4. 虚拟内存技术可以使多个进程共享同一个运行库，并通过分割不同进程的内存空间来提高系统的安全性。

管理内存的方法改进

内存分段

虚拟地址通过段表进行映射，段表中包括段选择子和段内偏移量，其中段选择子中最重要的就是段号，来作为段表的索引。

优点：解决了物理内存地址的对应问题。

缺点： 1.内存碎片问题。 2.有内存分段问题导致交换效率低。

内存分页

分页是把整个虚拟和物理内存切成一段段固定的尺寸大小，在这样一个连续的且固定的内存空间里面，我们把他们叫做页，通过页表和页内偏移量来访问。

虚拟地址和物理地址通过页表进行映射。

优点：1.解决了内存碎片的问题，采用了分页，所有的内存都以页为单位进行释放，就不会产生无法给进程使用的小内存了。2.解决了内存交换效率低的问题，如果内存空间不够，操作系统可以把正在运行的内存页面写入磁盘然后释放，需要的时候再加载到内存中。(Swap 机制)

缺点： 空间缺陷：操作系统可以运行非常多的进程。每一个进程都需要页表，那么最后这个页表就会过于庞大

多极页表

通过给页表分级，形成多级页表。

优点： 1.因为操作系统的物理内存往往是不会全部被使用完，所以多级页表会比单纯的内存分页效果更好。

缺点： 多次转换浪费时间

TLB(Translation Lookaside Buffer)页表缓存

CPU 里面有一个专门存放最常访问的页表项 cache。

1. 段页式内存管理

内存分段和内存分页可以组合起来使用，我们把它叫做段页式内存管理。 地质结构由段号，段内页号，页内偏移三部分构成。 方式：先将程序划分为多个有逻辑意义的段，再把每个段划分位多个页，也就是分段画出来的连续空间，再划分固定大小的页。

进程与线程

进程

对于我们运行一个储存在硬盘中的程序，这个运行中的程序就叫做进程

状态：

1. 运行时刻：时刻占用 cpu
2. 就绪状态：可运行，由于其他进程处于运行而暂停
3. 阻塞状态：等待某一事件的发生，就开始运行

上下文切换：

1. cpu：CPU 寄存器和程序计数是 CPU 在运行任何任务之前，所必须依赖的环境，这个就叫做 CPU 的上下文
2. 进程：进程是由内核管理和调度的，所以进程的切换只能发生在内核态；通常，会把交换的信息保存在进程的 PCB 中，当要运行另外一个进程的时候，我们需要从这个进程的 PCB 中取出上下文然后恢复到 cpu 中，使得这个进程可以继续执行

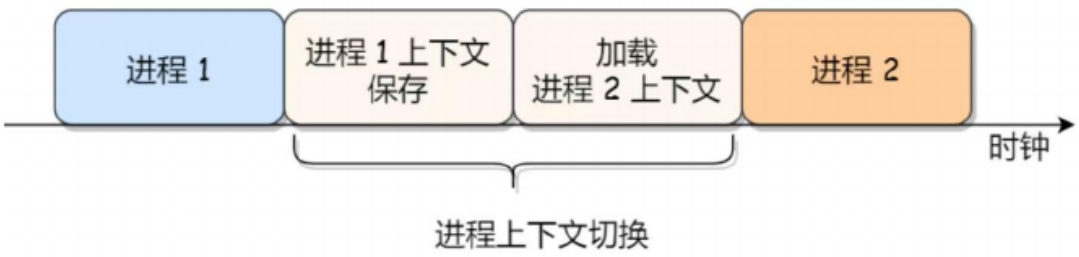

Figure 1: 内存中的进程调度

线程

线程是进程当中一条执行流程，线程之间可以并发运行且共享相同的地址空间 linux 中进程和线程都是用 task-struct 结构体来表示的 在 linux 系统中，根据任务的优先级和响应要求，按需要调度

cpu 如何选择线程？ 完全公平调度：一般情况下，在优先级相同的情况下，那么都是选择时间最少的依 次调度，如果有权重值则需要加权调度

线程的选择

线程的优点：
1. 一个进程中可以有多个线程
2. 多个线程之间可以并发执行
3. 各个线程之间可以共享地址空间和文件等资源

线程的缺点： 当进程中一个线程崩溃的时候，会导致其所属进程的所有线程崩溃

线程与进程的比较：
1. 进程是资源分配的单位，线程是 cpu 调度的单位
2. 进程拥有一个完整的资源平台，线程只是独享必不可少的状态
3. 线程同样也是有就绪，阻塞，执行三种基本状态，同样具有状态之间的转换关系
4. 线程能减少并发执行的时间和空间开销

++ 线程创建时间比进程快 ++ 线程终止要比进程快 ++ 线程的切换要比进程快 ++ 同一个进程各线程之间共享内存和文件资源

线程分类

1.用户线程：在用户空间实现的进程，不是由内核管理的线程

优点：
1. 他有私有的线程控制块 TCB，可以用于不支持线程技术的操作系统
2. 速度比较快。用户线程的切换也是由线程库函数来完成的。无需用户态和内核态的切换。速度快

缺点：
1. 因为操作系统不参与导读。如果一个线程发起系统调用堵塞，所有的用户线程都不能执行了
2. 一个线程正在运行，除非他能主动交出 cpu 使用权。否则他所在的进程无法运行
3. 由于时间分配给进程，与其他进程相比在多线程执行的时候每个线程得到的时间较慢

2.内核线程：在内核中实现的线程，是由内核（操作系统）管理的线程

优点：
1. 在一个进程中，如果某个内核线程发生系统调用堵塞，并不影响其他内核线程的运行
2. 分配给线程，多线程的进程获得更多的 cpu 运行时间

缺点：
1. 由内核来维护进程和线程的上下文信息。如 PCB 和 TCB
2. 线程的创建。终止和切换都是通过系统调用的方式来进行的，对于系统来说系统开销比较大

3.轻量级线程 在内核中来支持用户线程

一对一： 优点：并行，当一个线程被堵塞不会影响其他的线程 缺点：创建线程的开销大，每一个用户线程就会产生一个内核线程

多对一： 优点：可以多开 缺点：一个线程出问题就全部出问题

多对多：优点：充分结合上面两个优点。完美合理的调度

调度

触发时机： 状态改变到时候就会触发

调度分类： 非抢占式调度：一直运行到结束才会调用另外一个 抢占式调度：一个还没运行完就回去调度另外一个

原则：CPU 利用率：保证 cpu 始终繁忙，提高 cpu 利用率 系统吞吐量：保证长任务段任务均匀分配。在单位时间内完成的任务数量最多 周转时间：进程运行和阻塞时间综合，一个进程的周转时间越小越好 等待时间：处于就绪队列时间越短越好。 响应时间：用户提交依次请求到相应的时间越短越好

调度算法：算法用来决定优先运行那个进程 先来先服务调度算法（deque） 最短作业优先调度算法（优先选择时间最短的进行） 高响应比优先调度算法（通过优先权进行计算，优先权=（等待时间+要求服务时 间）/要求服务时间 时间片轮转调度算法（每一个进程都有一个单位时间，时间一过就交给下一个进程进行）时间一般在 20ms-50ms 最高优先级调度算法（通过优先级进行调度）：
1. 静态优先级（运行的时候已经确定了优先级）
2. 动态优先级（主要看时间）
3. 也有抢占式和非抢占式之分

多级反馈队列调度算法