进程控制

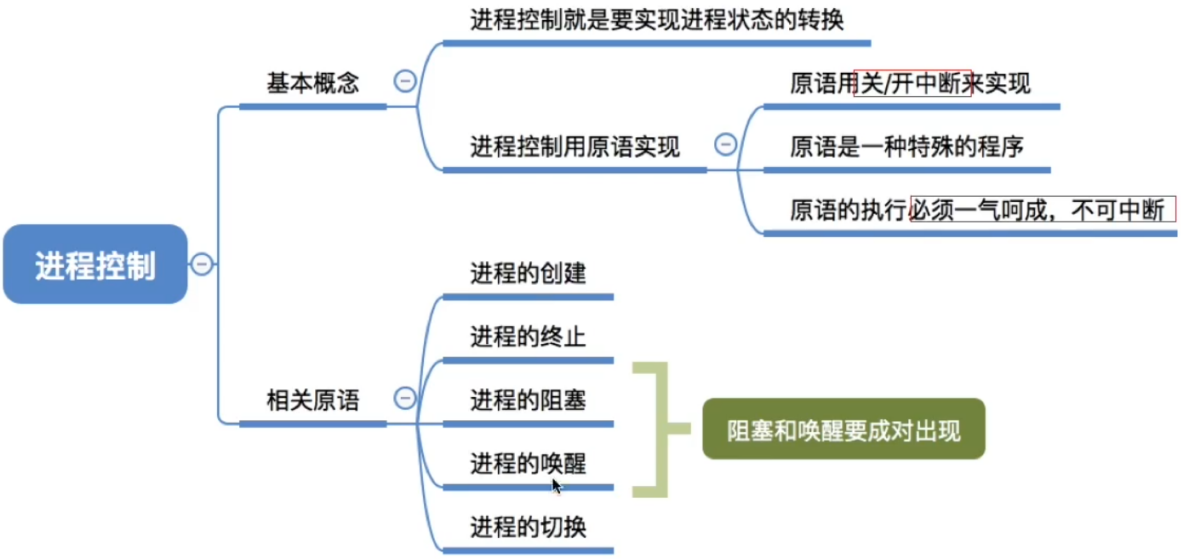
无论哪个进程控制原语，要做的无非三类事情：

1. 更新PCB中的信息

   修改进程状态（state）保存/恢复运行环境

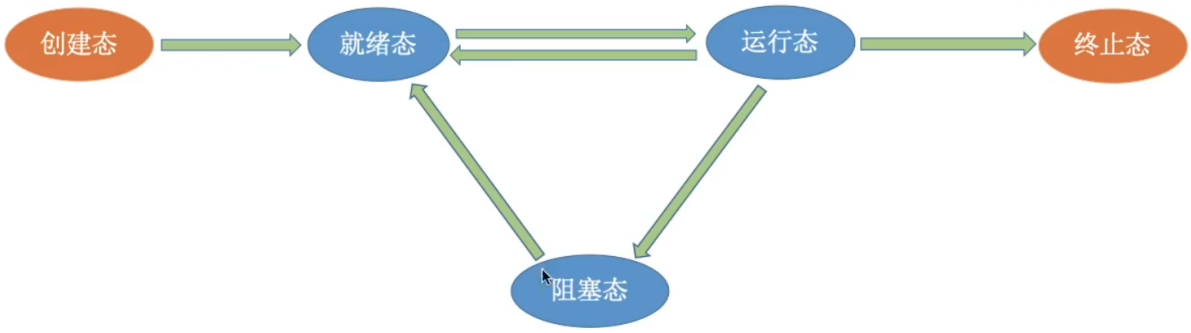
2. 将PCB插入合适的队列

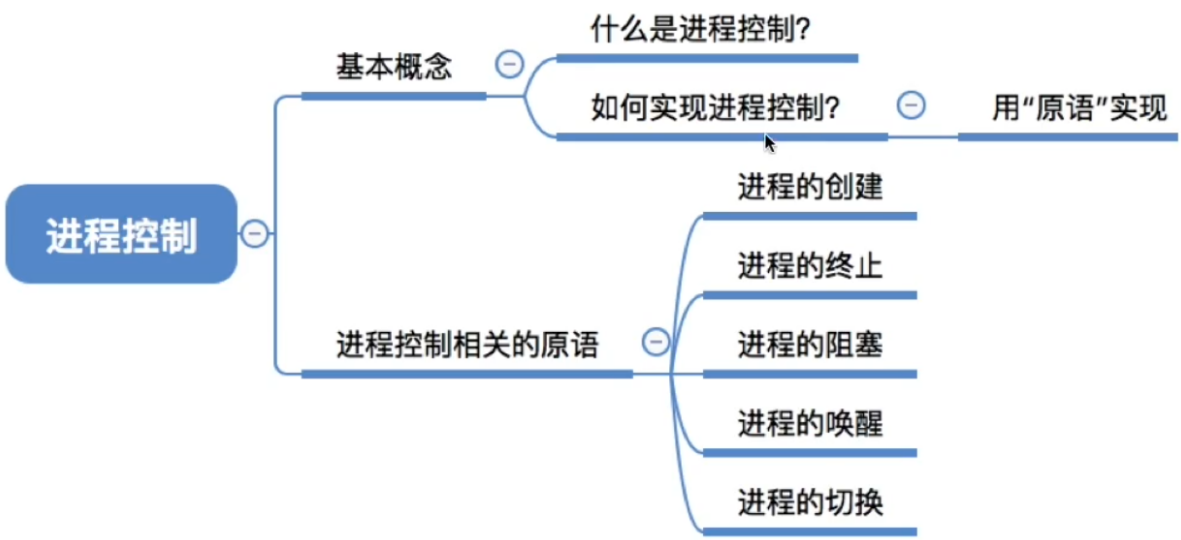
3. 分配/回收资源



进程控制的主要功能是对系统中的所有进程实施有效的管理，它具有创建新进程、撤销已有进程、实现进程状态转换等功能。

简化理解：进程控制就是要实现进程状态转换





如何实现进程控制？

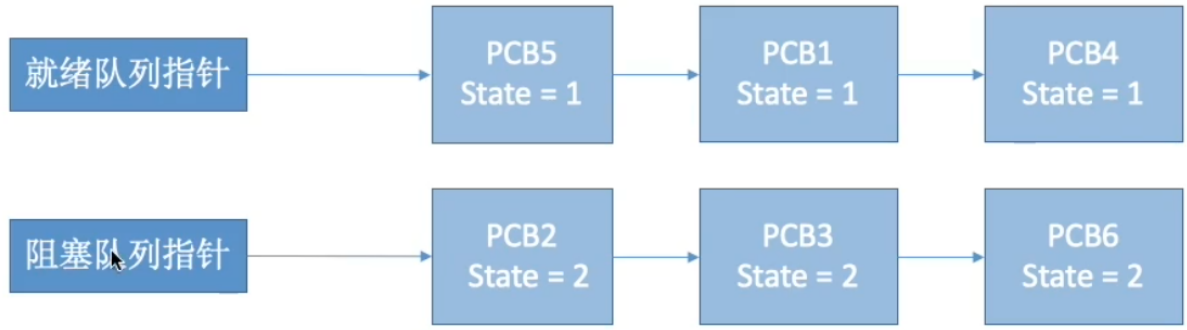
用”原语“实现

计算机系统的层次结构

原语是一种特殊的程序，它的执行具有原子性。也就是说，这段程序的运行必须一气呵成，不可中断

思考：为何进程控制（状态转换）的过程要”一气呵成“？

假设PCB中的变量state表示进程当前所处状态，1表示就绪态，2表示阻塞态



假设此时进程2等待的事件发生，则操作系统中，负责进程控制的内核程序至少需要做这样两件事：

1. 将PCB2的state设为1

   完成了第一步后收到中断信号，那么PCB2的state=1，但是它却被放在阻塞队列里

   如果不能“一气呵成”，就有可能导致操作系统中的某些关键数据结构信息不统一的情况，这会影响操作系统进行别的管理工作

2. 将PCB2从阻塞队列放到就绪队列

如何实现原语的“原子性”？

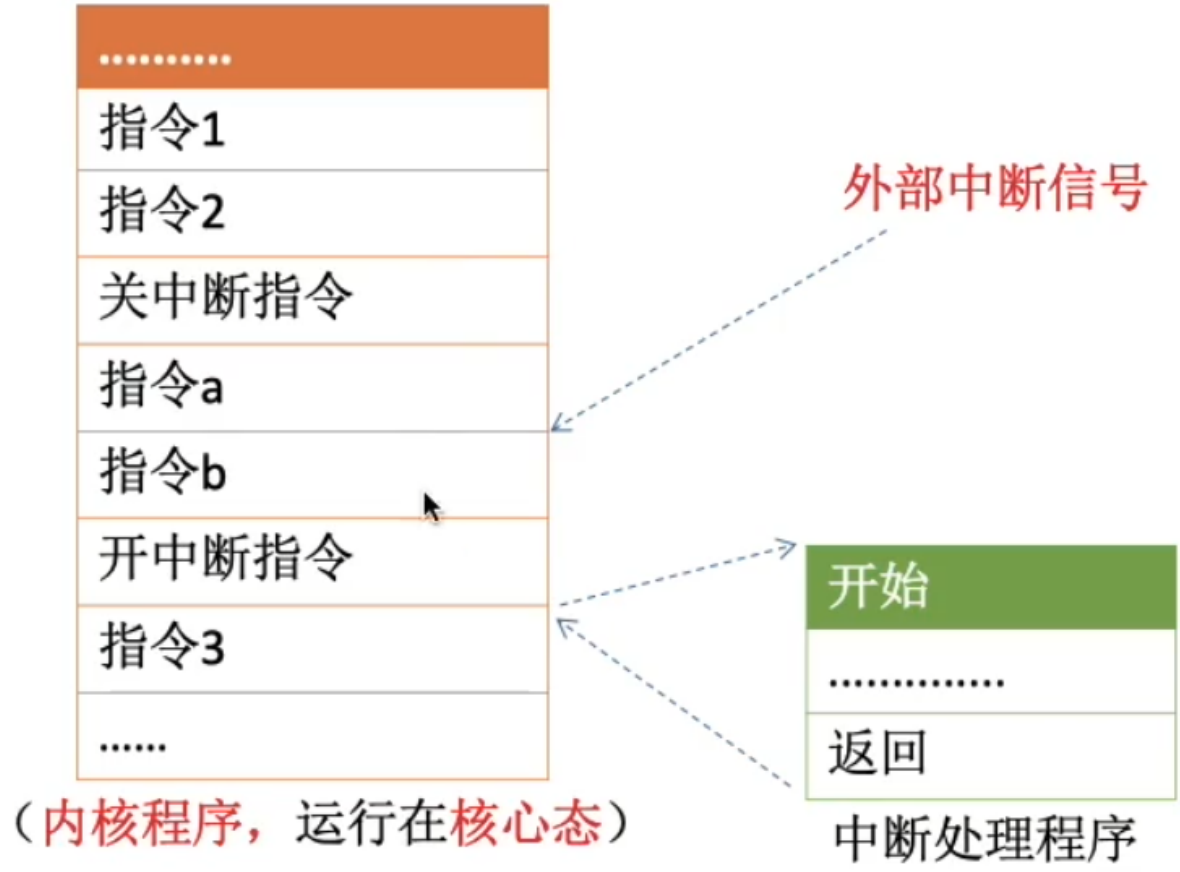
原语的执行具有原子性，即执行过程只能一气呵成，期间不允许被中断。

可以用”关中断指令“和”开中断指令“这两个特权指令实现原子性

正常情况：CPU每执行完一条指令都会例行检查是否有中断信号需要处理，如果有，则暂停运行当前这段程序，转而执行相应的中断处理程序。

CPU执行了关中断指令之后，就不再例行检查中断信号，直到执行开中断指令之后才会恢复检查。

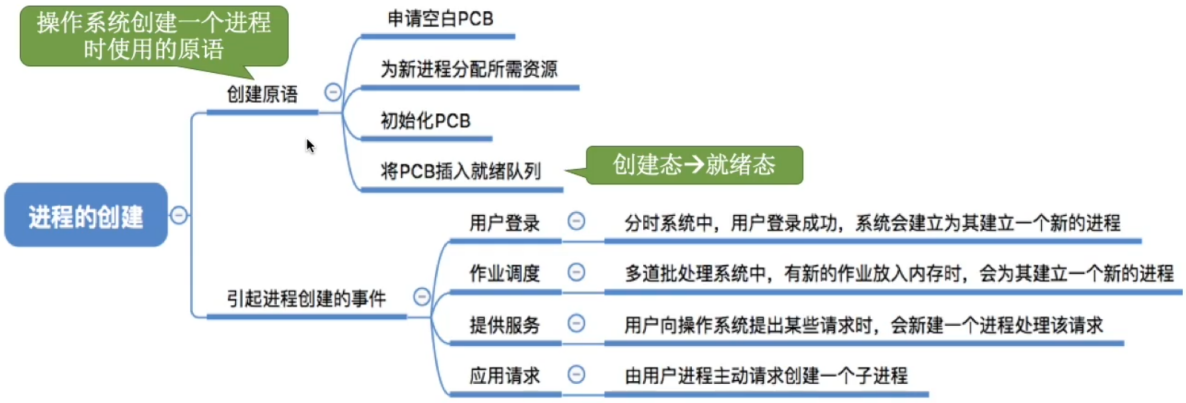
这样，关中断、开中断之间的这些指令序列就是不可被中断的，这就实现了”原子性“



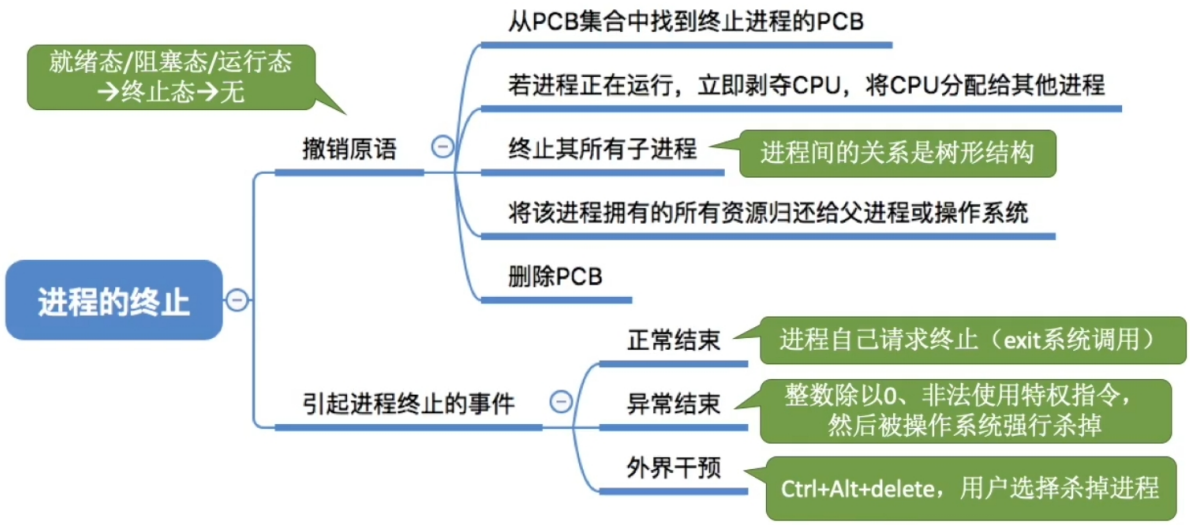
思考：如果这两个特权指令允许用户程序使用的话，会发生什么情况?

进程控制相关的原语

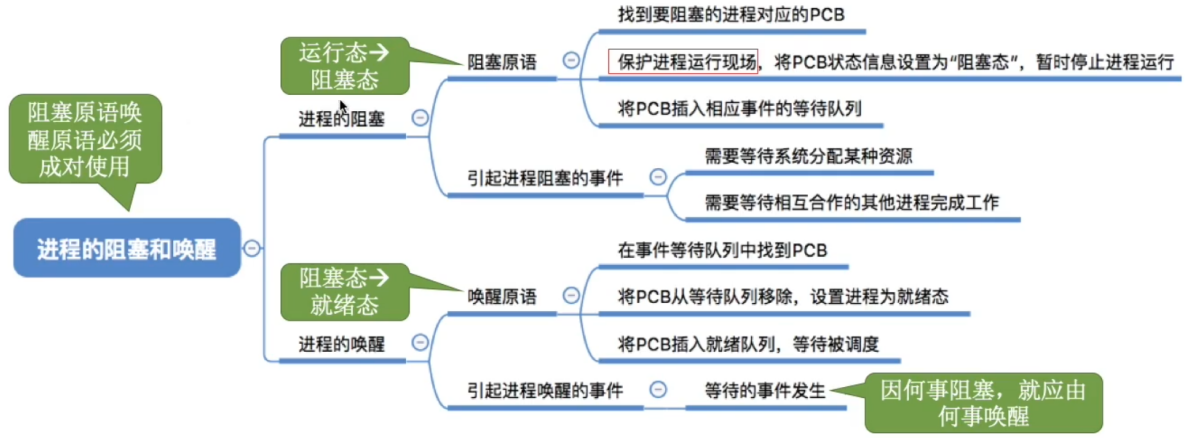
创建原语



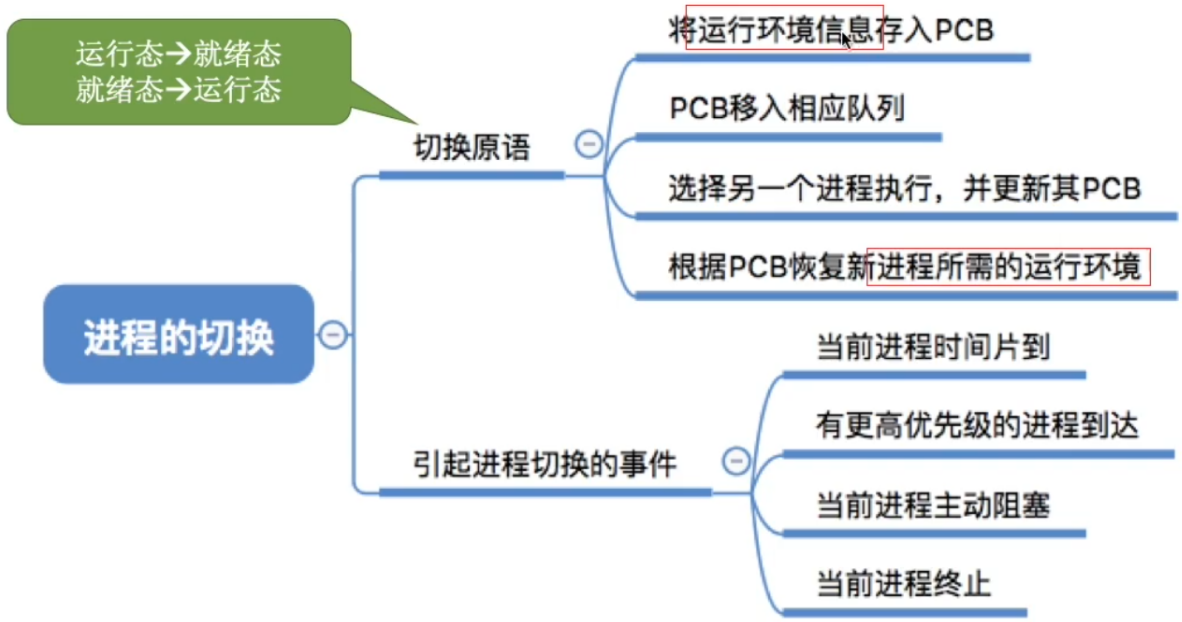
撤销原语



阻塞和唤醒原语



切换原语



CPU中会设置很多寄存器，用来存放程序运行过程中所需的某些数据。

- PSW

  程序状态字寄存器

- PC

  程序计数器，存放下一条指令的地址

- IR

  指令寄存器，存放当前正在执行的指令

- 通用寄存器

  其他一些必要信息

思考：执行完指令后，另一个进程开始上CPU运行。

注意：另一个进程在运行过程中也会使用各个寄存器

解决办法：在进程切换时先在PCB中保存这个进程的运行环境（保存一些必要的寄存器信息）

当原来的进程再次投入运行时，可以通过PCB恢复它的运行环境