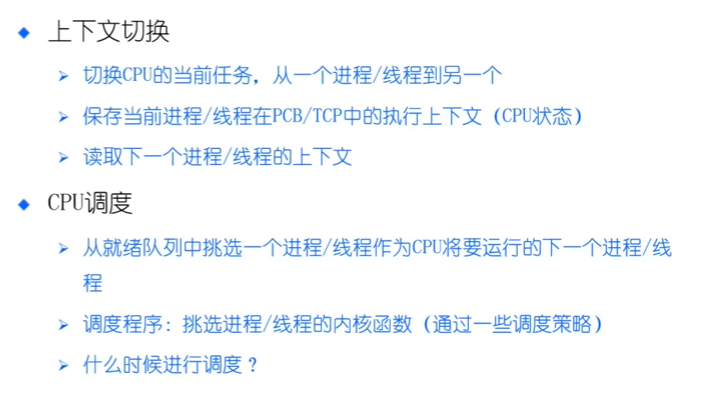
调度：



进程切换到运行态便能够占用cpu执行

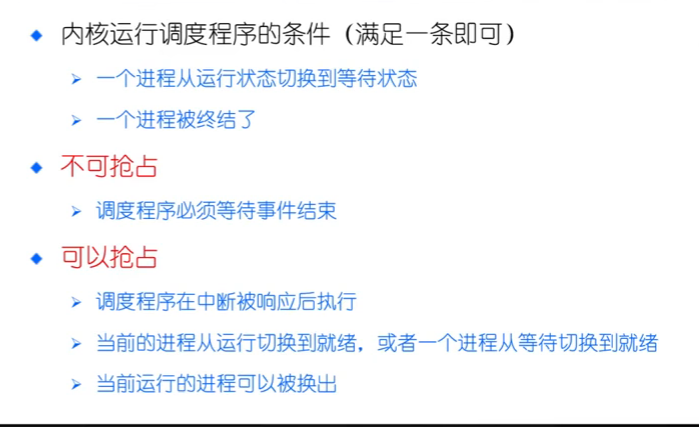
那么进程什么时候切换，怎么切换由cpu调度决定schedule



从一个状态到另一个状态时会触发调度，尤其和运行状态相关

如就绪到运行，运行到等待，运行到退出

一般是调度应用程序，一般是以用户态进程存在



非抢占调度策略，即一个进程启动之后，别的进程不可打断，导致其他进程会等前一个正在运行的进程执行结束。

抢占式：操作系统决定在某个时刻，打断当前用户进程的执行，再去选择别的进程进行执行。

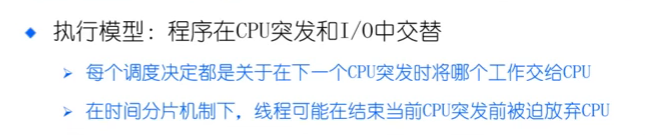
这两个策略针对的是用户态的进程。

内核也有抢占和不可抢占：

如果一个进程执行系统调用，但该调用不会导致进程处于等待状态，还是running，那这个系统调用一定还会返回发起该调用的进程继续执行。也就不会出现抢占。

调度原则：

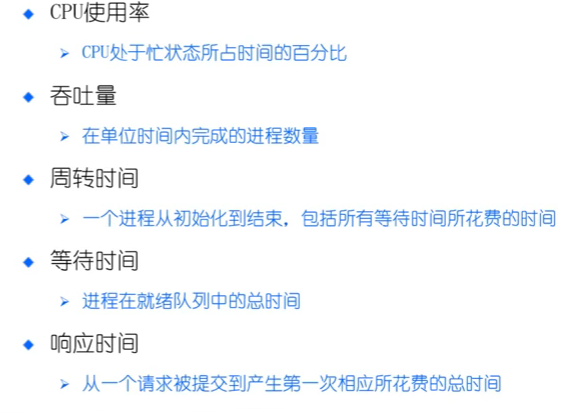




程序可能在执行时会占用cpu和I/O接口（硬盘），交替，

我们希望能够某个进程在等待I/O时可以切换到其他就绪进程执行其CPU操作。

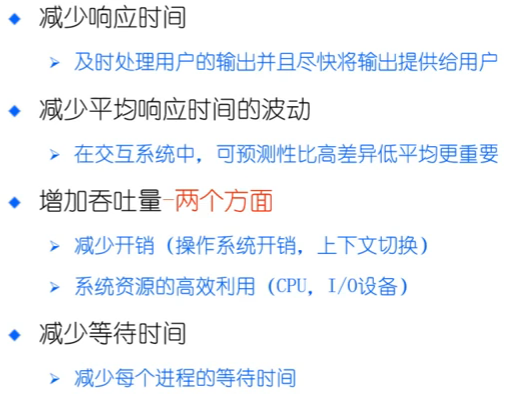
协调进程CPU繁忙和I/O繁忙状态。

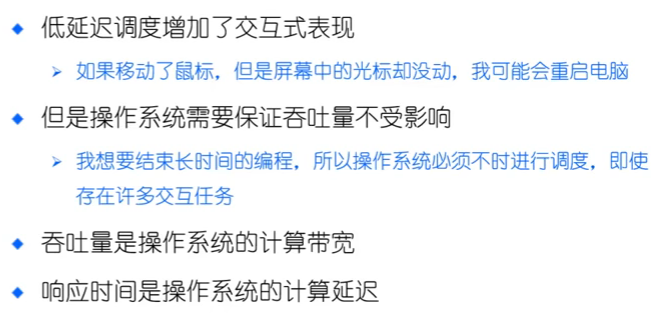


评价指标：上述很多

CPU利用率高，当前系统效率很好

我们希望调度算法：

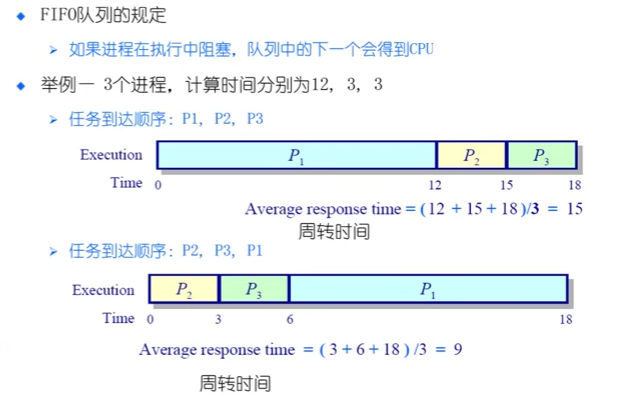




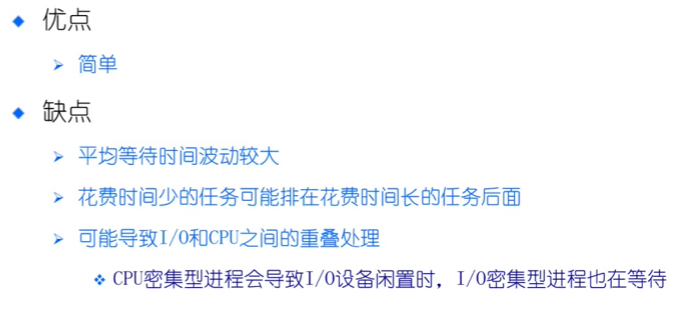
调度算法：



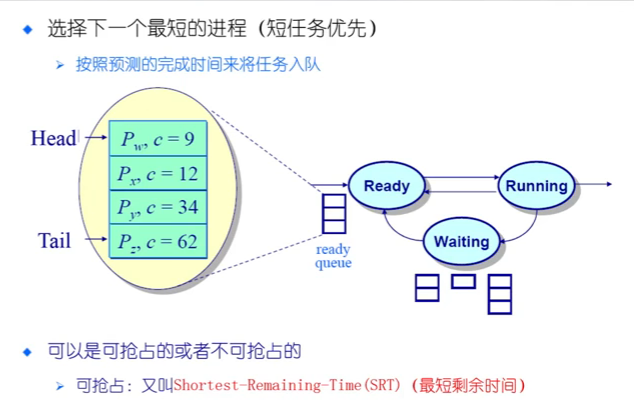
**FCFS先来先服务：**



这里的周转中P2和P3从就绪态便可以计算。



**SRT短进程优先：**

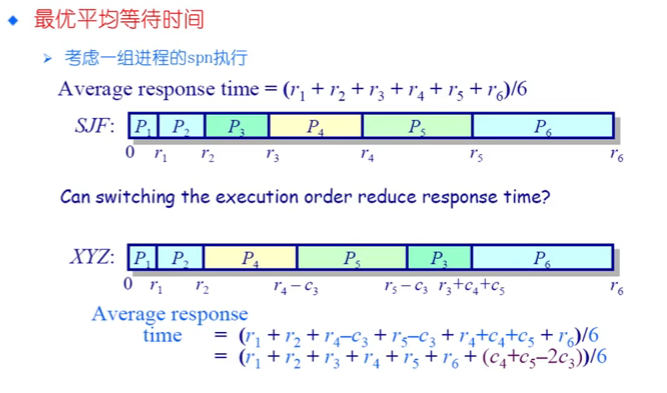


如果在当前进程运行时来了一个执行时间更短的进程处于就绪态，此时有两种：

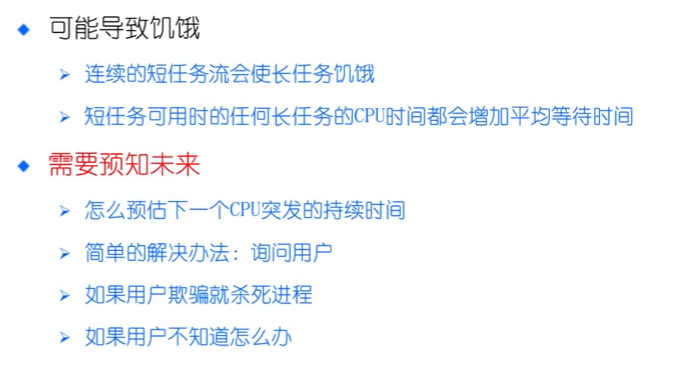
当前运行的进程依旧运行，不理会新来的进程，即非抢占（SPN/SJF），也就是进程正在时不会被优先级更高的打断。

抢占（SRT），也就是如果新来的进程的时间比正在执行的更短，优先级更高，那么正在执行的进程被打断，从运行态变成就绪态，挂会到就绪队列中，执行新来的优先级更高的进程。

短进程优先的算法的周转时间、等待时间是最短的，最小的。

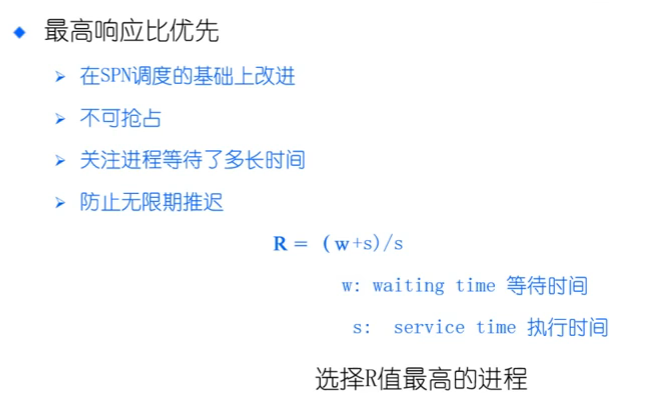


其缺点为：



我们打开一个进程很难知道它什么时候结束，不知道其执行时间。

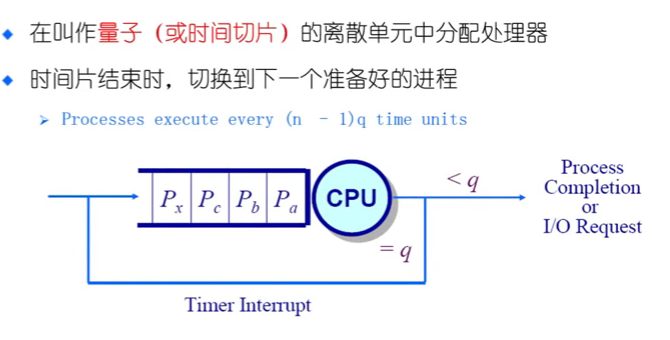
**HRRN最高响应比优先：**



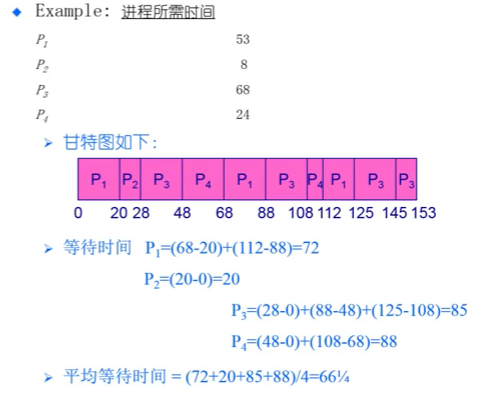
Service time和SPN一样需要去预估。

**Round robin轮循调度算法：**

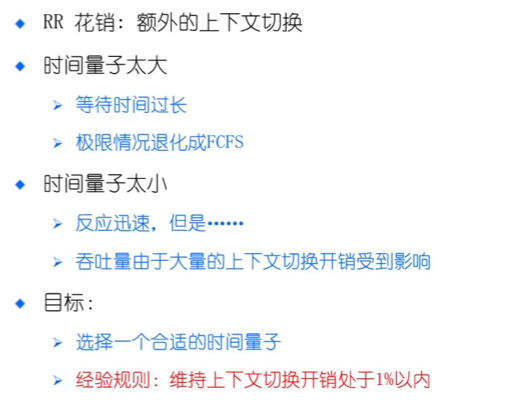
让各个进程轮流占用CPU去执行。



例子如下：



RR轮循的缺点：

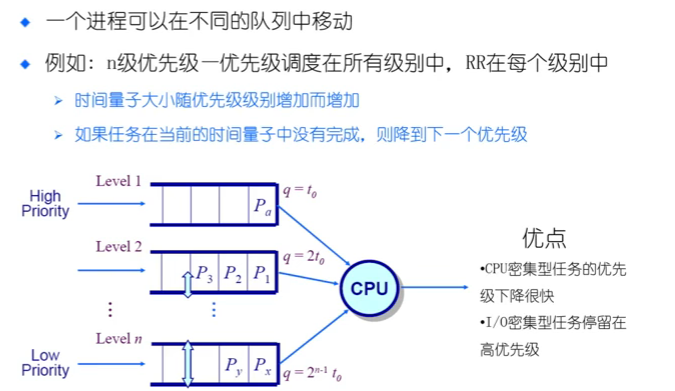


有时候FCFS的平均等待时间甚至比RR好一些，但是RR会让每个进程都有执行。

**MFQ多级反馈队列：**

把就绪进程分成很多个队列，每个队列采用不用的调度算法

高优先级的任务都执行完之后才去执行低优先级的任务



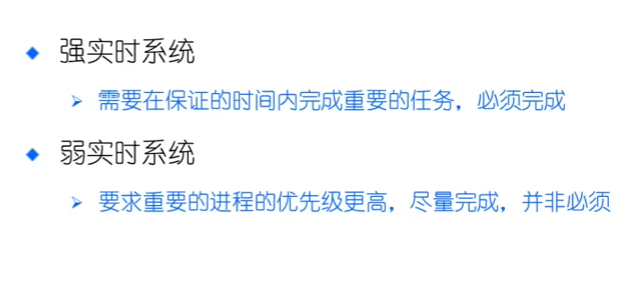
等待时间越长，优先级越高，执行时间越长，优先级越低

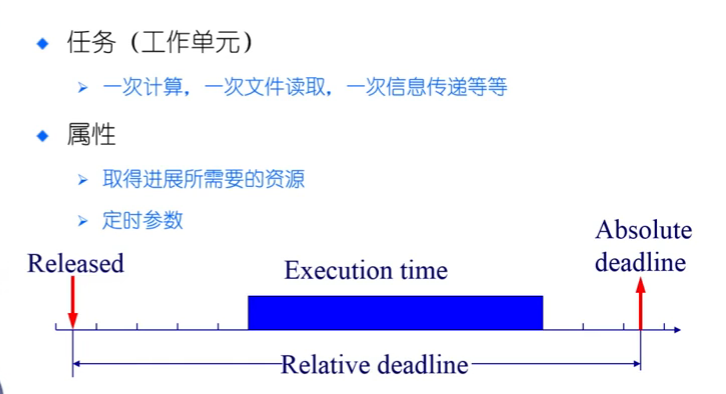
也就是I/O密集型的优先级会不断提高，而CPU密集型会降低

**FSS公平共享调度**

不是重点

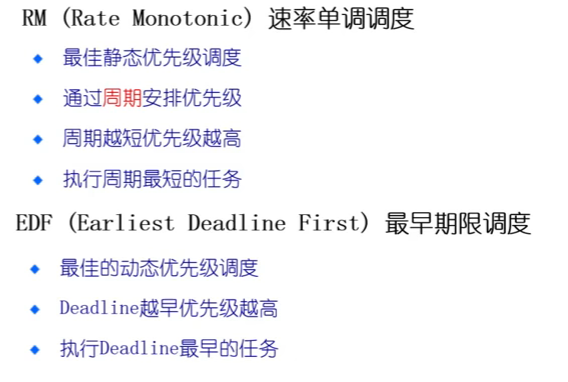
实时调度：





Release time就绪时间

Deadline就是截止时间，时间片的截止时间



多处理器的CPU调度