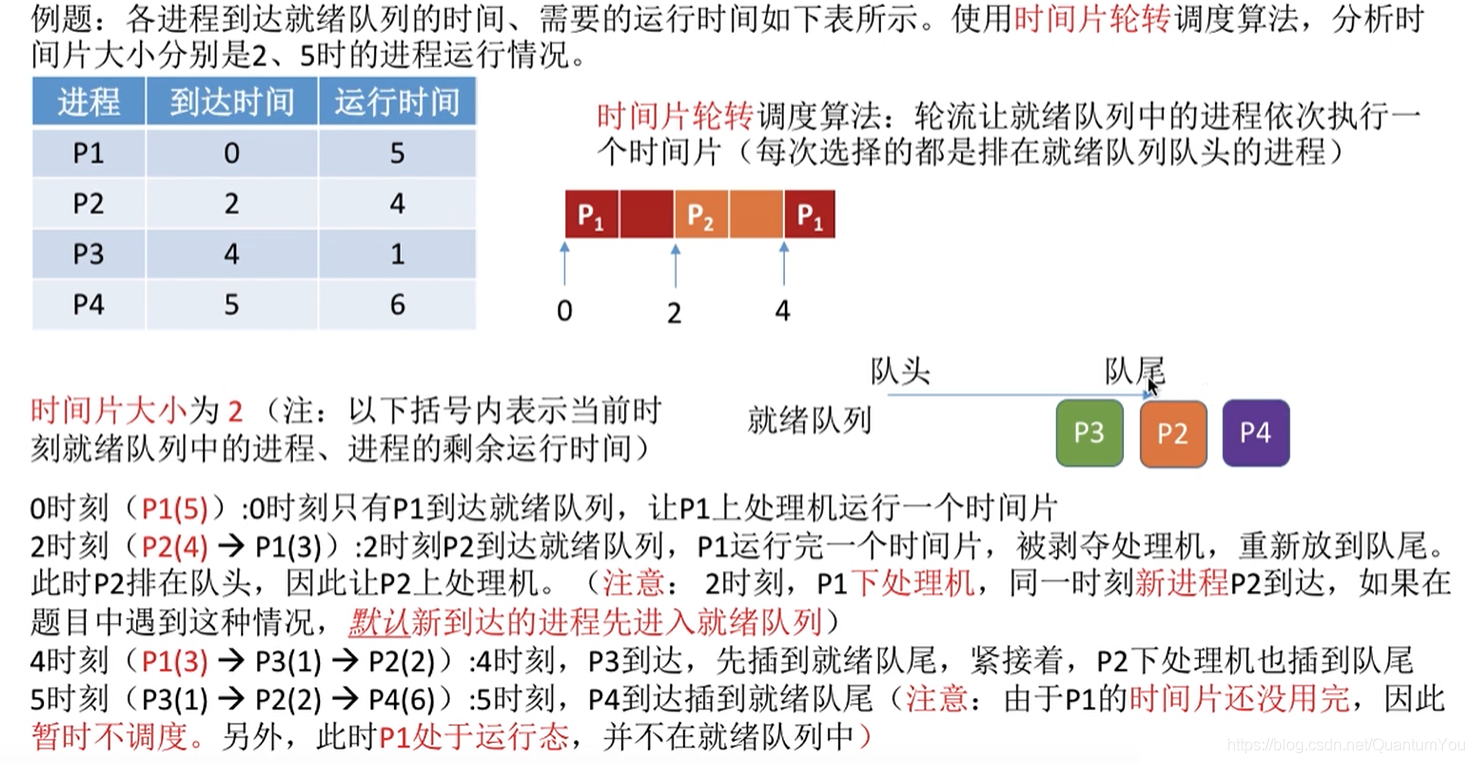
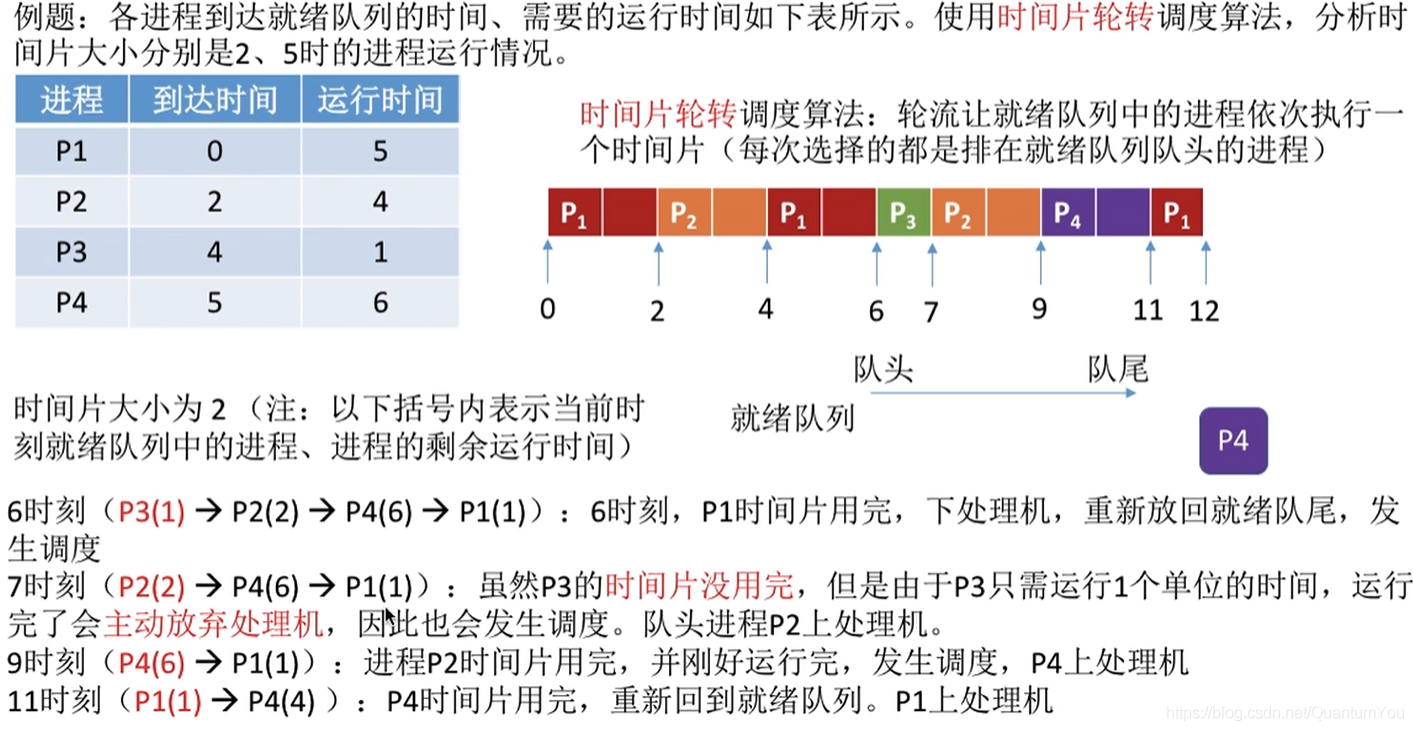
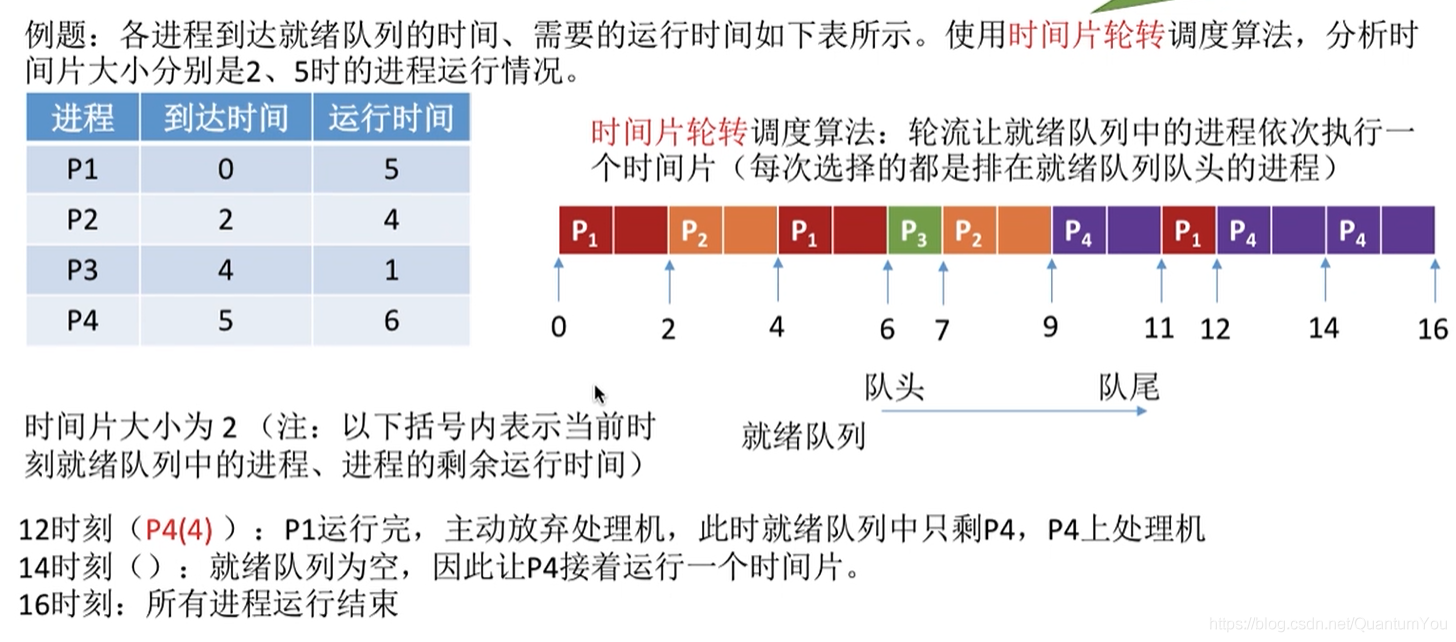
大三技术水平综合成长的一年

@[toc]

# 时间片轮转算法

* RR 算法 （Round - Robin）
* **算法思想**：公平地、轮流地为各个进程服务，让每个进程在一定时间间隔内都可以得到响应
* **算法规则** ： 按照各进程到达就绪队列的顺序，轮流让各个进程执行一个时间片（如100ms）.若进程未在一个时间片内执行完则剥夺处理机，将进程重新放到就绪队列队尾重新排队。
* **用于作业 / 进程调度**： 用于进程调度（只有作业放入内存建立了相应的进程后才能被分配处理机时间片）
* **是否可抢占**：若进程未能在时间片内运行完，将被强行剥夺处理机使用权，因此时间片轮转调度算法属于抢占式的算法。由时钟装置发出时钟中断来通知CPU时钟片已到
* **优缺点** ： 优点：公平；响应快，适用于分时操作系统  
  缺点：由于高频率的进程切换，因此有一定开销；不区分任务的紧急程度。
* **是否导致饥饿**（某进程/作业长时间得不到服务） 不会

## 重要例子

时间片大小为2 的情况  
  
  
  
注意： 上述的动态变化，排在就绪队列的队头与尾



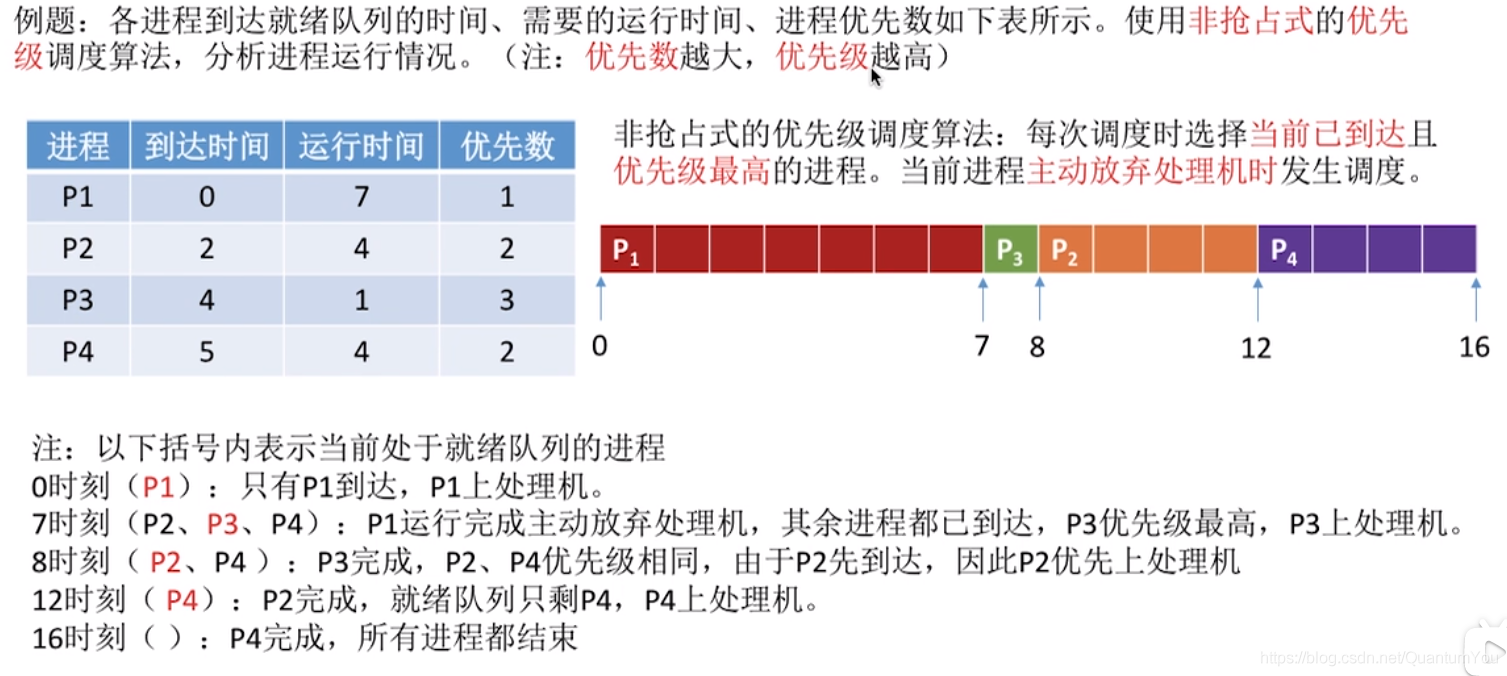
**注意**：

* 如果时间片太大，使得每个进程都可以在一个时间片内就完成，则时间片轮转调度算法退化为先来先服务调度算法，并且会增大进程响应时间。因此时间片不能太大。
* 另一方面，进程调度、切换是有时间代价的（保存、恢复运行环境），因此如果时间片太小，会导致进程切换过于频繁，系统会花大量的时间来处理进程切换，从而导致实际用于进程执行的时间比例减少。可见时间片也不能太小。
* 一般来说，设计时间片时要让切换进程的开销占比不超过1%

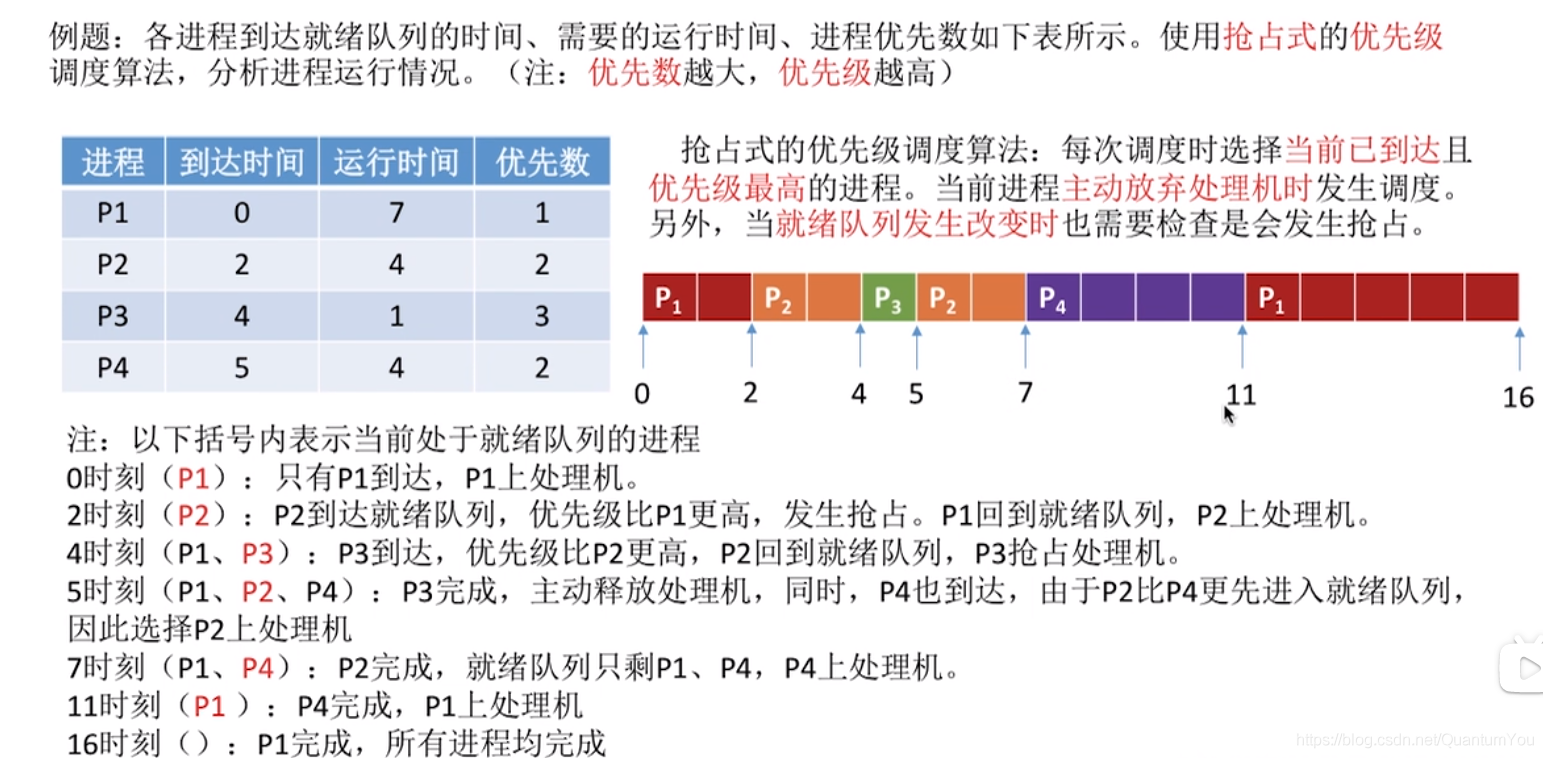
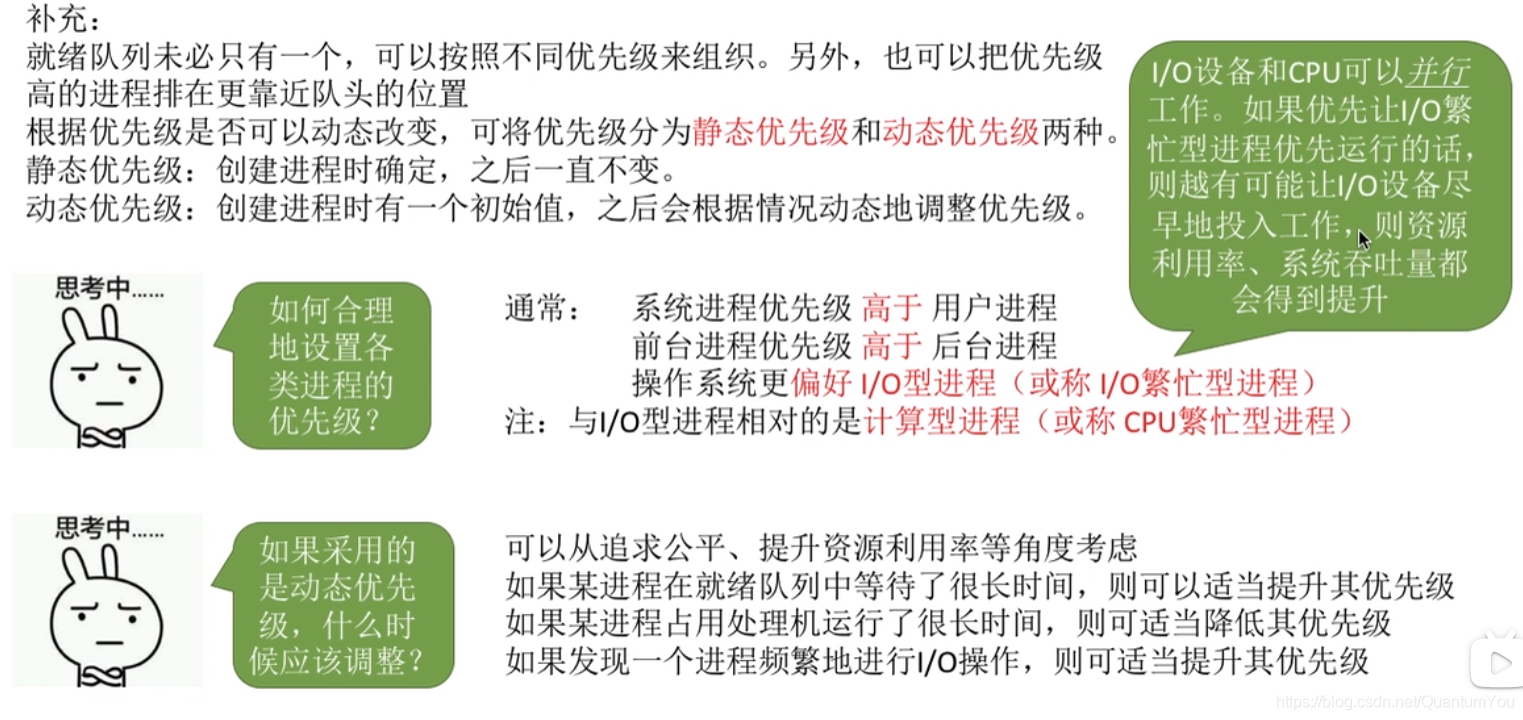
# 优先级调度算法

* **算法思想**：随着计算机的发展，特别是实时操作系统的岀现，越来越多的应用场景需要根据任务的紧急程度来决定处理顺序
* **算法规则** ： 每个作业/进程有各自的优先级，调度时选择优先级最高的作业/进程
* **用于作业 / 进程调度**： 既可用于作业调度，也可用于进程调度。甚至，还会用于在之后会学习的I/O调度中
* **是否可抢占**：抢占式、非抢占式都有。做题时的区别在于：非抢占式只需在粧程主动放弃处理机时进行调度即可，而抢占式还需在就绪队列变化时，检查是否会发生抢占。
* **优缺点** ：优点：用优先级区分紧急程度、重要程度，适用于实时操作系统。可灵活地调整对各种作业/进程的偏好程度。缺点：若源源不断地有高优先级进程到来，则可能导致饥饿
* **是否导致饥饿**（某进程/作业长时间得不到服务） 会

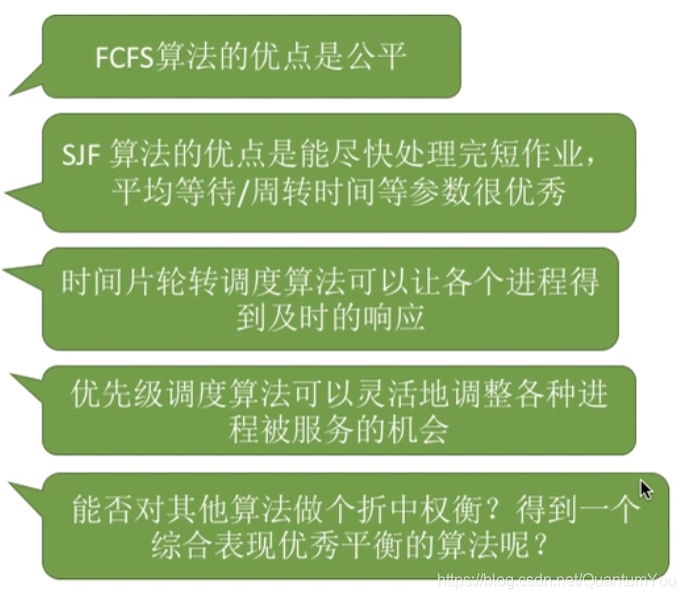
## 重要例子

非抢占式调度算法：  


抢占式调度算法（到达的线程优先级比当前优先级高）

## 算法总结

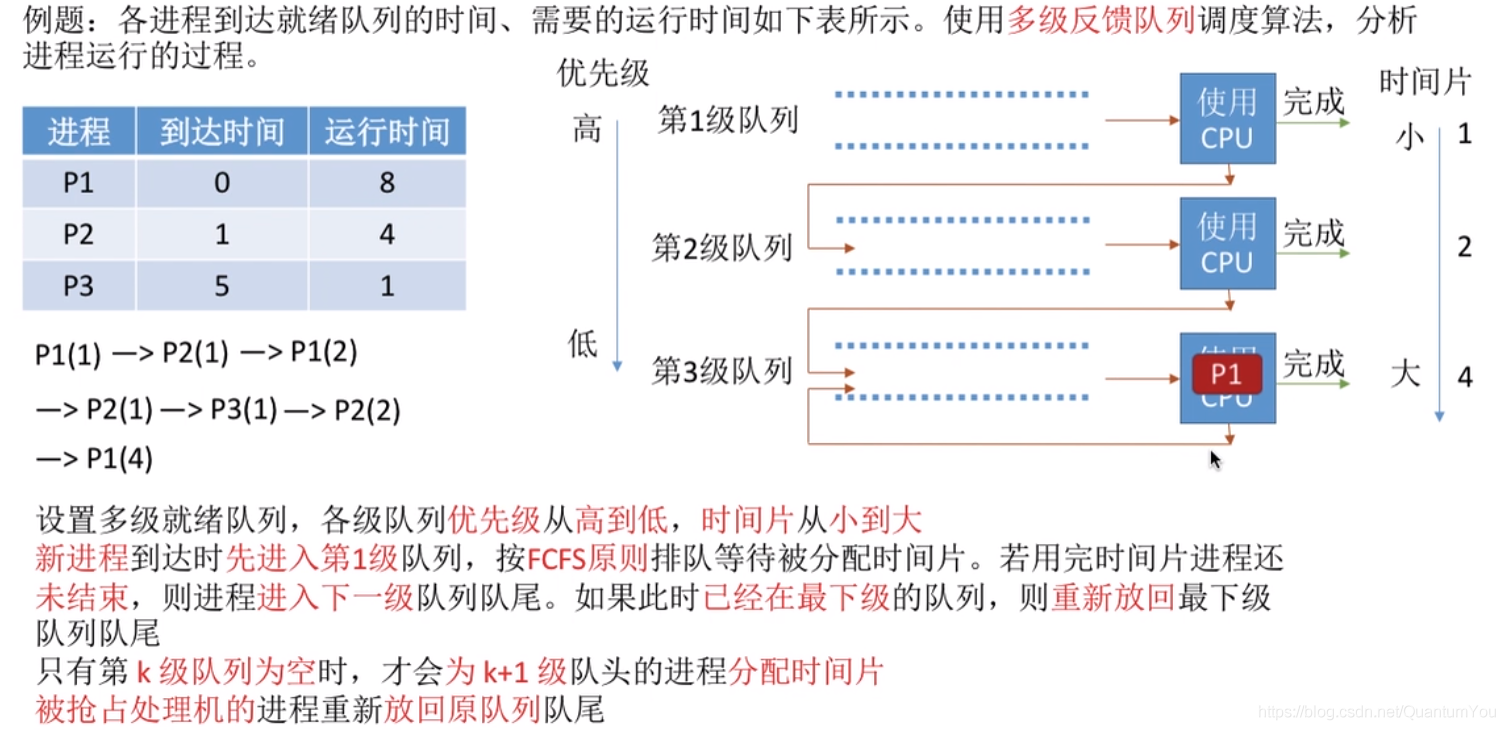


# 多级反馈队列调度算法

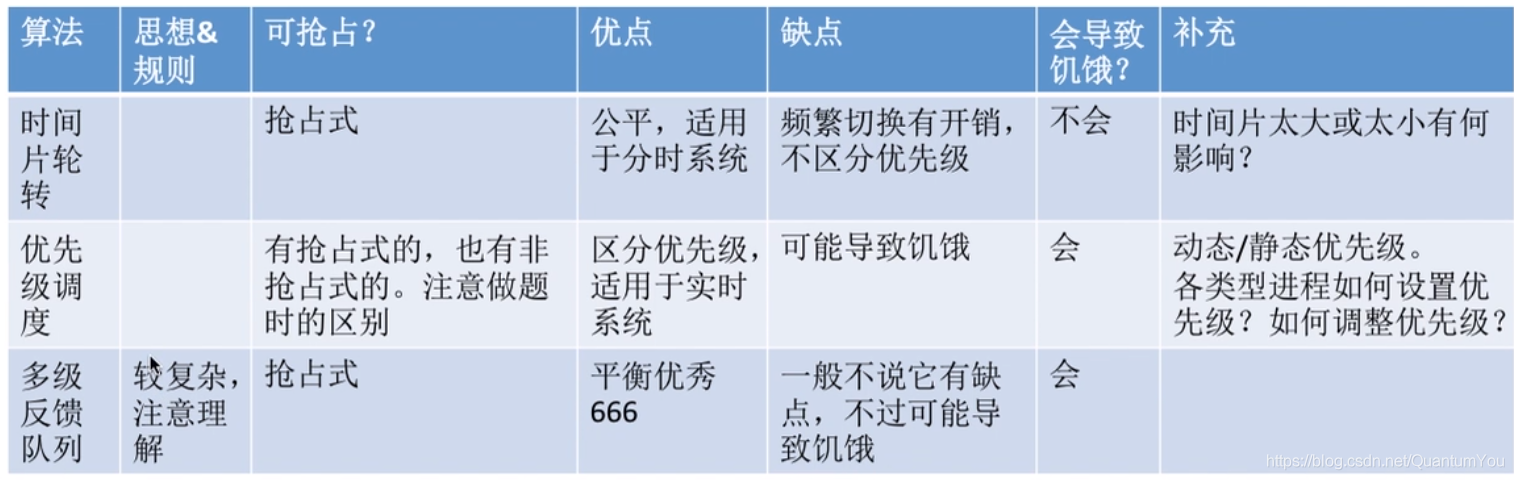
* **算法思想**：对其他调度算法的折中权衡
* **算法规则** ：1、设置多级就绪队列，各级队列优先级从高到低，时间片从小到大 2、新进程到达时先进入第1级队列，按FCFS原则排队等待被分配时间片，若用完时间片进程还未结束，则进程进入下一级队列队尾。如果此时已经是在最下级的队列，则重新放回该队列队尾3、只有第k级队列为空时，才会为k+1级队头的进程分配时间片
* **用于作业 / 进程调度**： 用于进程调度
* **是否可抢占**：抢占式的算法。在k级队列的进程运行过程中，若更上级的队列（1~k-1级）中进入了一个新进程，则由于新进程处于优先级更高的队列中，因此新进程会抢占处理机，原来运行的进程放回k级队列队尾。
* **优缺点** ：对各类型进程相对公平（FCFS的优点）；每个新到达的进程都可以很快就得到响应（R的优点）；短进程只用较少的时间就可完成（SPF的优点）；不必实现估计进程的运行时间（避免用户作假）；可灵活地调整对各类进程的偏好程度，比如CpU密集型进程、J/O密集型进程（拓展：可以将因/O而阻塞的进程重新放回原队列，这样J/o型进程就可以保持较高优先级）
* **是否导致饥饿**（某进程/作业长时间得不到服务） 会

## 重要例子

多级反馈队列调度算法运行动态顺序



## 小结

  
注意：比起早期的牝处理操作系统来说，由于计算机造价大幅降低，因此之后出现的交互式操作系统（包括分时操作系统、实时操作系统等）更注重系统的响应时间、公平性、平衡性等指标。而这几种算法恰好也能较好地满足交互式系统的需求。因此这三种算法适合用于交互式系统。（比如UNⅨ使用的就是多级反馈队列调度算法）