**调度算法分析**

备注：每张图都有两份，前一张是后一张的截图，因为担心组件和图片错位。

【FCFS】

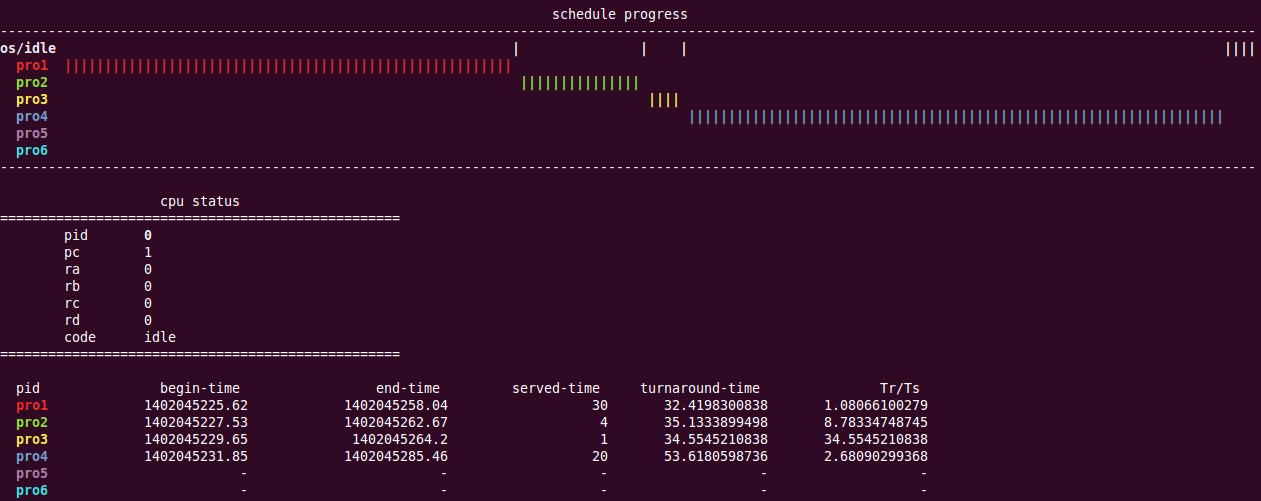
算法原理：当每个进程就绪后，它就加入就绪队列。当前正在运行的进程停止执行时，选择在就绪队列中存在时间最长的进程运行。

优劣势：

1. 执行长进程比执行短进程更好

2. 相对于I/O密集型的进程，FCFS更有利于处理器密集型的进程

解释：一般I/O密集型进程所需要的处理器时间较短（所以I/O密集型进程应该优先服务，这样才能更好地利用外设）。从CPU角度，I/O密集型进程相当于短进程，处理器密集型进程相当于短进程。



【RR】

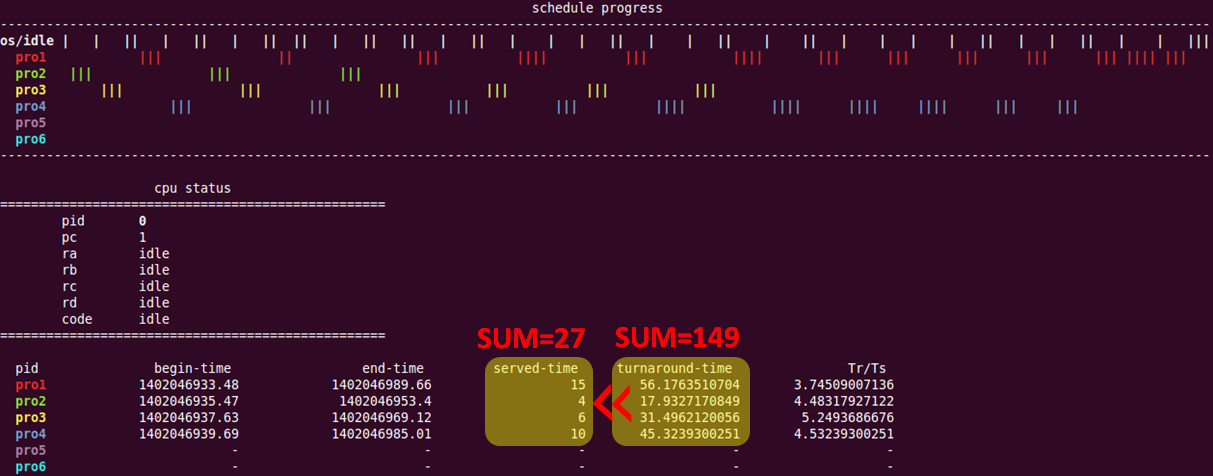
基本原理：系统将所有就绪进程按FCFS的原则，排成一个队列，依次调度，把CPU分配给队首进程，并令其执行一个时间片/CPU时间，通常为几个毫秒～几百毫秒。时间片用完后，该进程将被抢占并插入就绪队列末尾

优点：在时间片长度设置合理的情况下，更有利于短进程的调度。

对于轮转法，最主要的设计问题是时间片的长度。由于处理时钟中断、执行调度和分派函数都需要处理器开销。若时间片太短，频繁的切换进程会造成处理器开销大；若时间片太长，轮换法就退化成FCFS

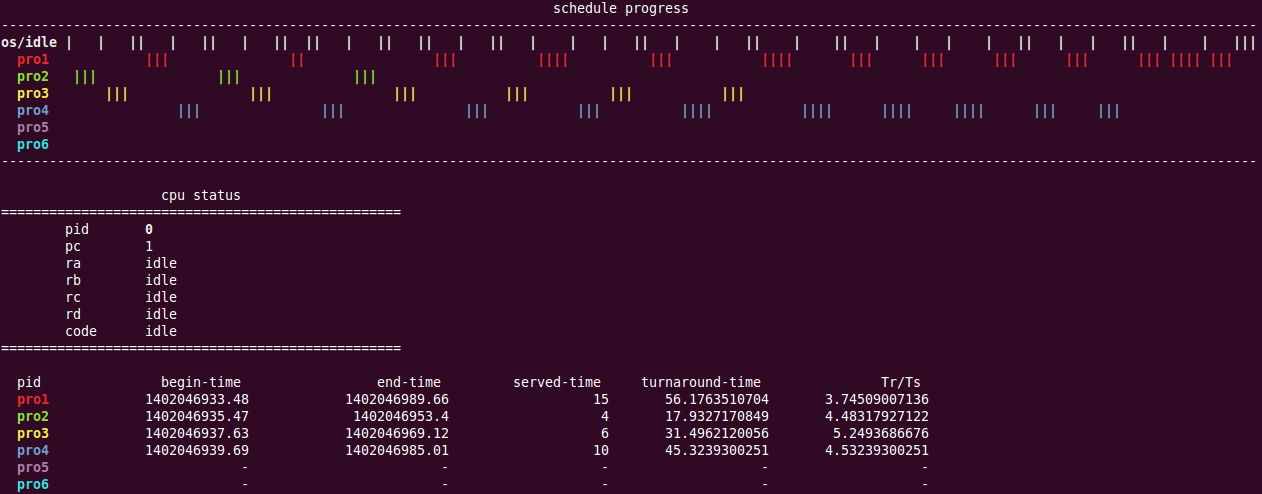
1. 时间片太短（q=1）

|  |  |  |
| --- | --- | --- |
|  | 到达时间 | 服务时间 |
| pro1 | 0 | 15 |
| pro2 | 2 | 4 |
| pro3 | 4 | 6 |
| pro4 | 6 | 10 |



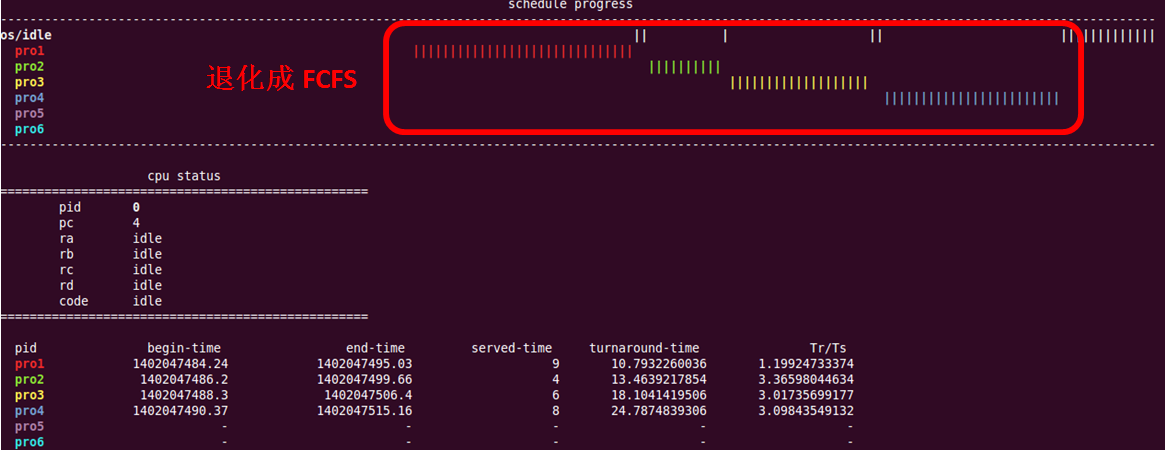
**SUM=149**

**SUM=27**

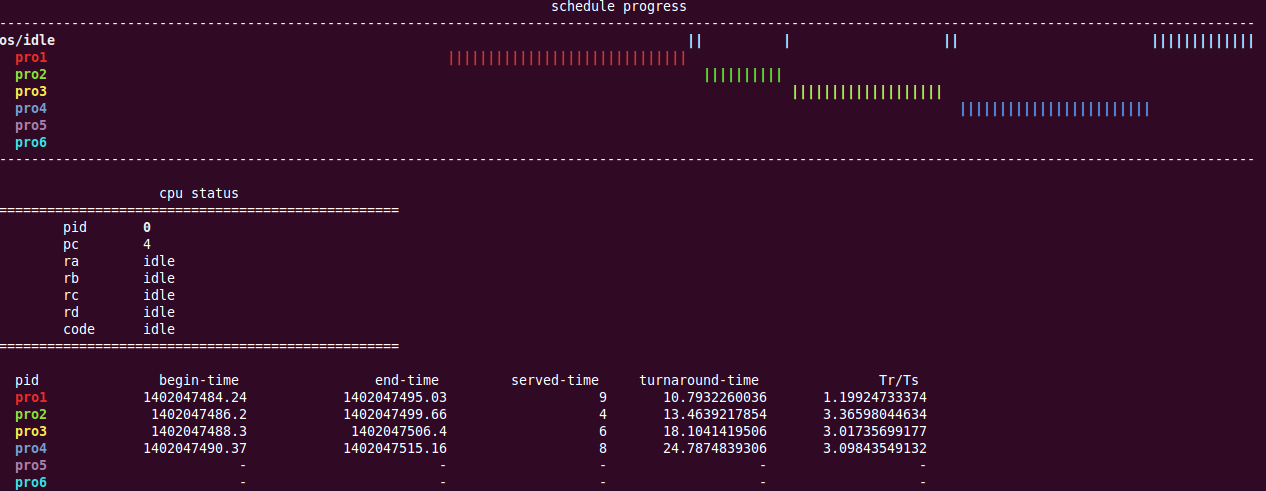


1. 时间片太长（q=10）

|  |  |  |  |
| --- | --- | --- | --- |
|  | 到达时间 | 服务时间 | 备注 |
| pro1 | 0 | 15 |  |
| pro2 | 2 | 4 | 长进程 |
| pro3 | 4 | 6 |  |
| pro4 | 6 | 10 |  |



**退化成FCFS**



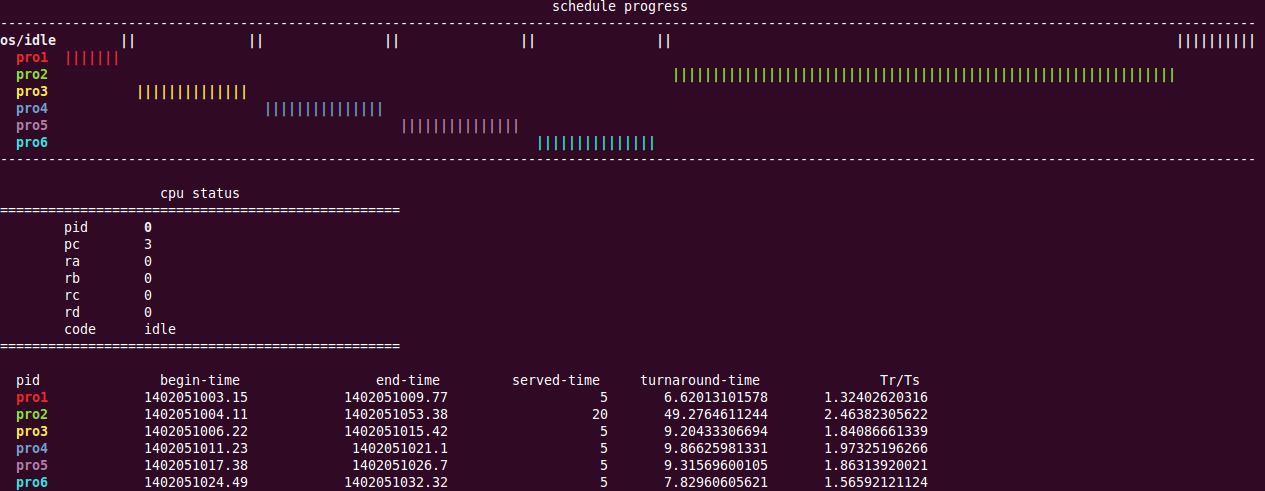
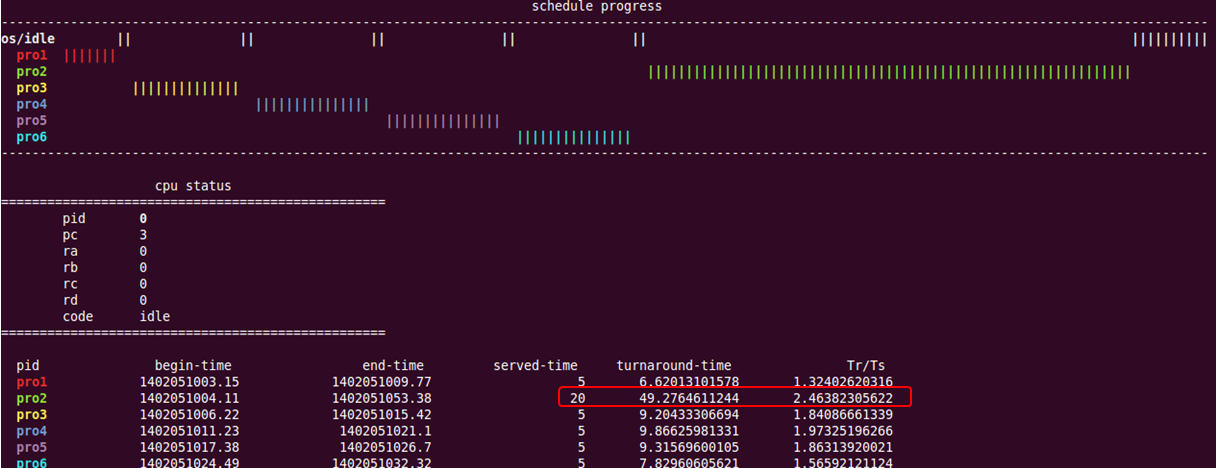
【SPN】

基本思想：从就绪队列中选出一估计运行时间最短的进程，将处理器分配给它。

优缺点：

1. 降低作业的平均等待时间，提高系统吞吐量
2. 对长作业不利；未考虑作业的紧迫程度
3. 对进程估计执行时间难以预测。

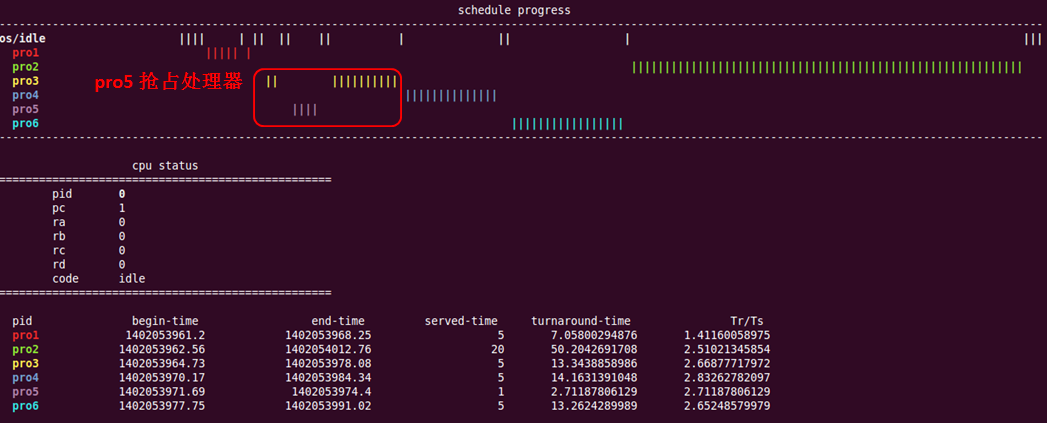
|  |  |  |  |
| --- | --- | --- | --- |
|  | 到达时间 | 服务时间 | 备注 |
| pro1 | 0 | 5 |  |
| pro2 | 1 | 20 | 长进程Tr/Ts大 |
| pro3 | 3 | 5 |  |
| pro4 | 8 | 5 |  |
| pro5 | 14 | 5 |  |
| pro6 | 21 | 5 |  |



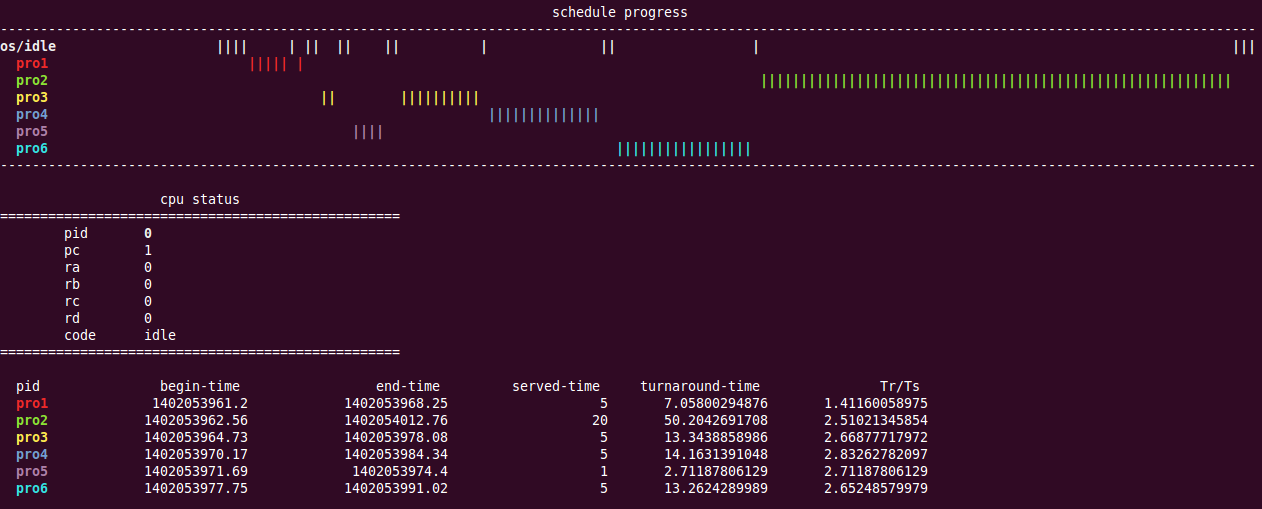
【SRT】

基本原理：SRT是针对SPN增加了抢占机制的版本。

|  |  |  |  |
| --- | --- | --- | --- |
|  | 到达时间 | 服务时间 | 备注 |
| pro1 | 0 | 5 |  |
| pro2 | 1 | 20 |  |
| pro3 | 3 | 5 | 被抢断进程 |
| pro4 | 8 | 5 |  |
| pro5 | 9 | 1 | 抢断进程 |
| pro6 | 15 | 5 |  |

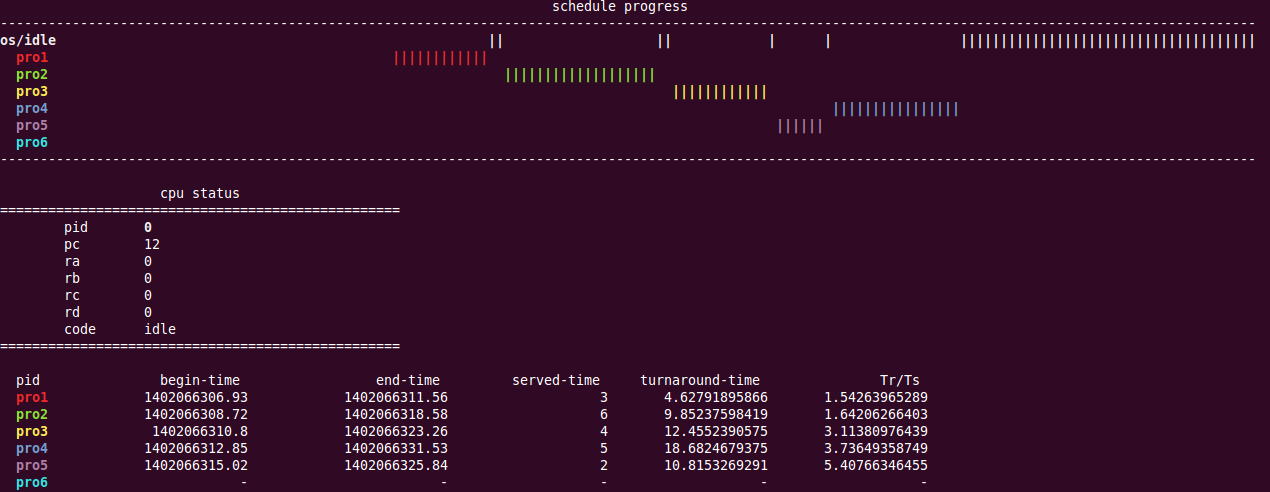


**pro5抢占处理器**



【HRRN】

|  |  |  |
| --- | --- | --- |
|  | 到达时间 | 服务时间 |
| pro1 | 0 | 3 |
| pro2 | 2 | 6 |
| pro3 | 4 | 4 |
| pro4 | 6 | 5 |
| pro5 | 8 | 2 |



基本原理：优先权=（等待时间+要求服务时间）/要求服务时间

（1） 如果进程的等待时间相同，则要求服务的时间愈短，其优先权愈高，因而该算法有利于短作业；  
（2）当要求服务的时间相同时，作业的优先权决定于其等待时间，因而实现了先来先服务；  
（3）对于长作业，当其等待时间足够长时，其优先权便可升到很高，从而也可获得处理机。

 该算法既照顾了短进程，又考虑了进程到达的先后顺序，也不会使进程长期得不到服务。因此，该算法实现了一种较好的折衷。当然，再利用该算法时，每要进行调度之前，都需先进行响应比的计算，这会增加系统的开销。

【FB】

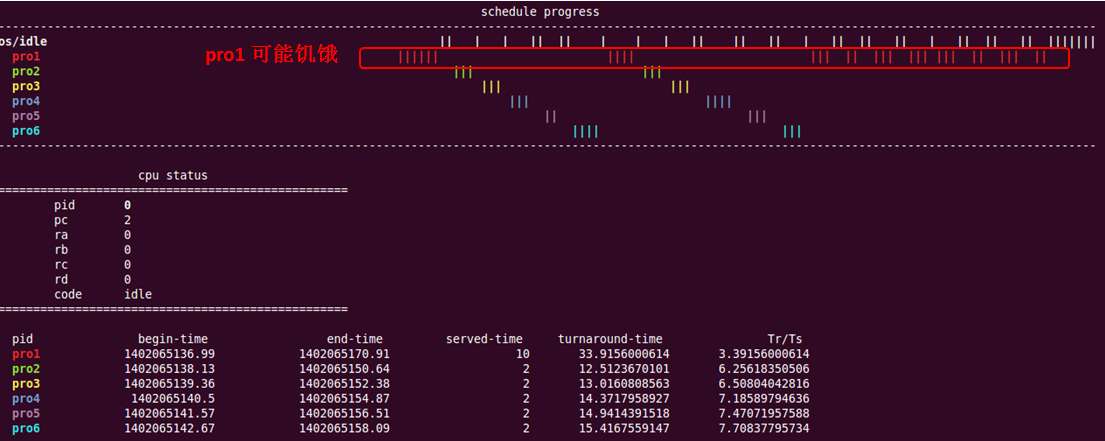
基本原理：调度基于抢占原则（按时间片）并使用动态优先级机制。新加入的进程放在RQ0，运行中的进程在每次被抢占后，被降级到下一个低优先级队列中。类似于时间片为1的轮转法。

缺点：若频繁有新进程进入系统，可能出现长进程饥饿。

解决方法：

1. 通过改变抢占次数解决，即越低优先级进程的允许执行之间越长。
2. 进程在队列中的等待时间超过一定量后，把它提升到一个优先级较高的队列中。

|  |  |  |  |
| --- | --- | --- | --- |
|  | 到达时间 | 服务时间 | 备注 |
| pro1 | 0 | 10 | 可能饥饿 |
| pro2 | 1 | 2 |  |
| pro3 | 2 | 2 |  |
| pro4 | 3 | 2 |  |
| pro5 | 4 | 2 |  |
| pro6 | 5 | 2 |  |



**pro1可能饥饿**

