操作系统

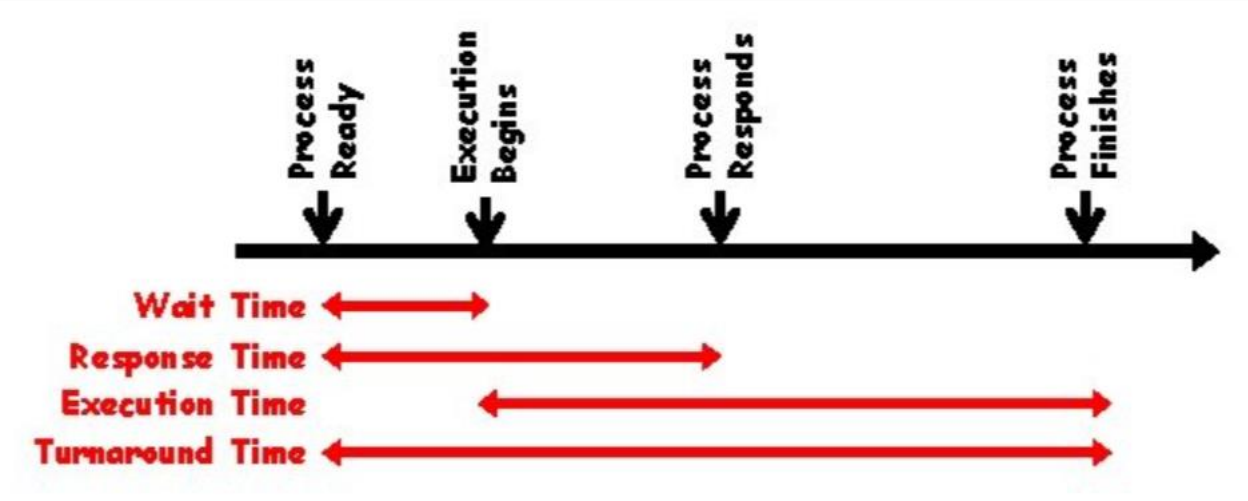
Operating system

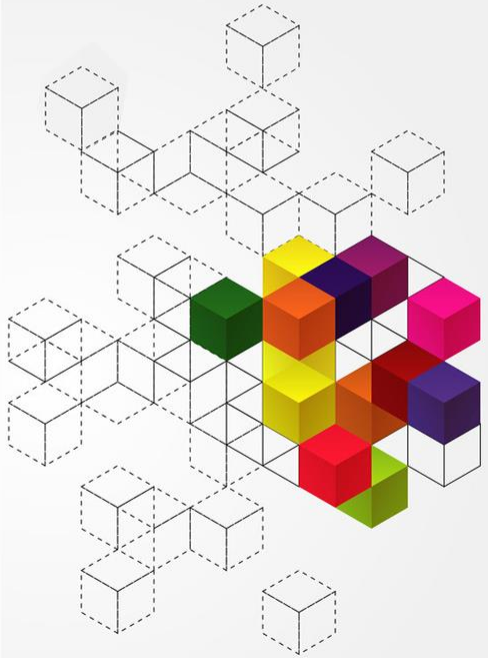
胡燕

大连理工大学

零、CPU调度算法1回顾

- 等待时间、周转时间、响应时间

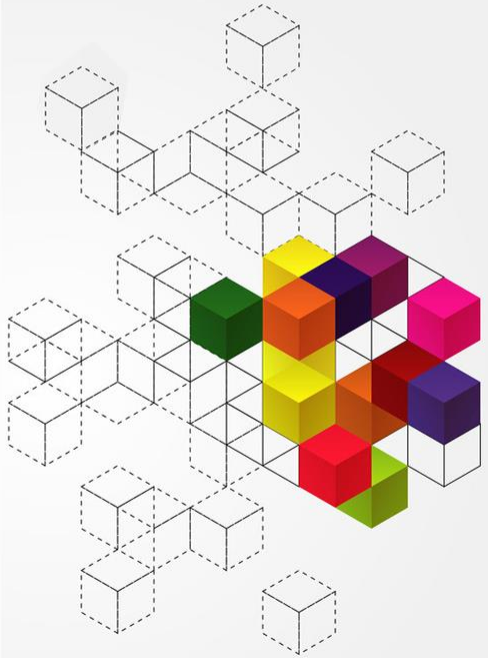




零、CPU调度算法1回顾

- **问题：我们认为可以将FCFS调度算法归入优先级调度算法这一类，为什么?**

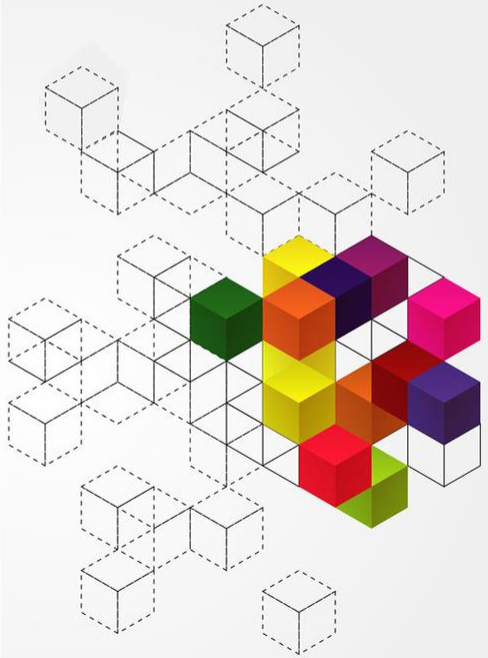
把进程的等待时间作为优先数，优先数越大，优先级越高



零、CPU调度算法1回顾

- **问题：我们认为可以将SJF调度算法归入优先级调度算法这一类，为什么?**

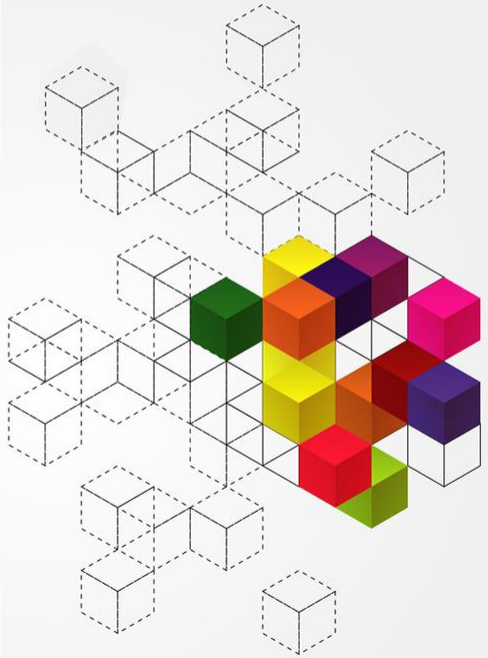
优先数是进程下一个CPU周期长度



零、CPU调度算法1回顾

- **问题：优先级算法存在的问题?**

低优先级进程可能被饿死（Starvation）

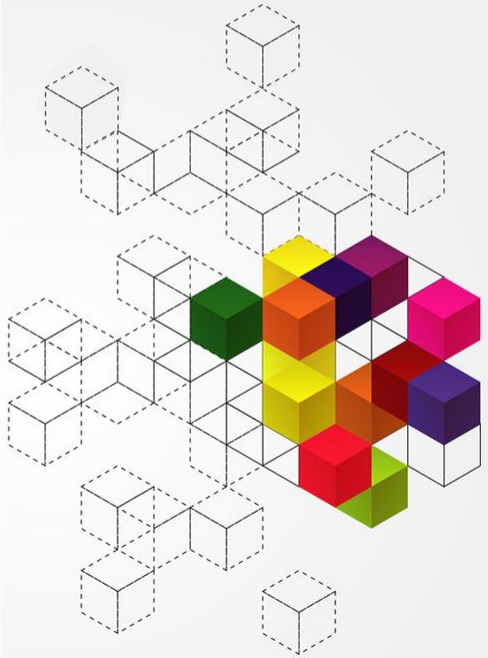


零、CPU调度算法1回顾

- **问题：优先级算法存在的问题?**

低优先级进程可能被饿死（Starvation）

解决问题的方法：aging

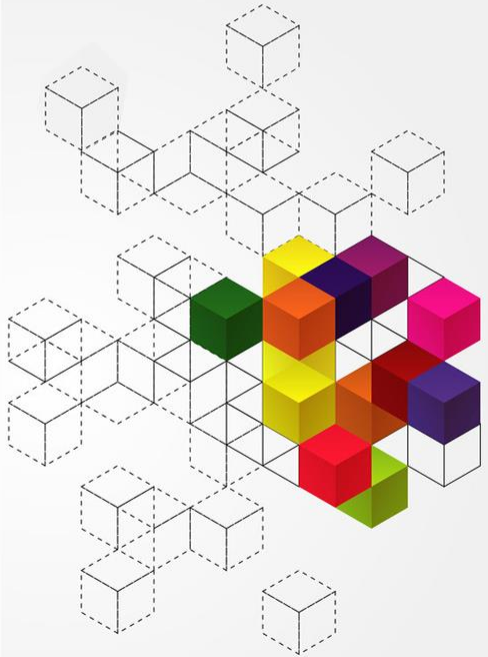


零、CPU调度算法1回顾

- **HRRN（高相应比优先）算法**

$$\text{Response Ratio} = (W + B)/B$$

W: Waiting time
B: Burst Time

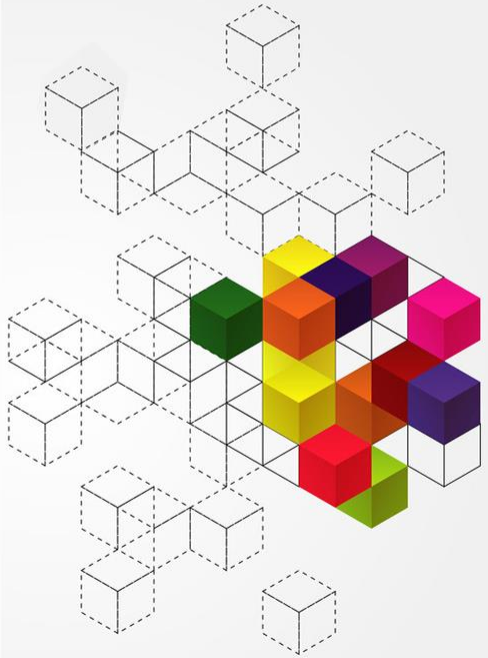


零、CPU调度算法1回顾

- **HRRN（高相应比优先）算法**

Thread	Arrival TIme	CPU Burst Length
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

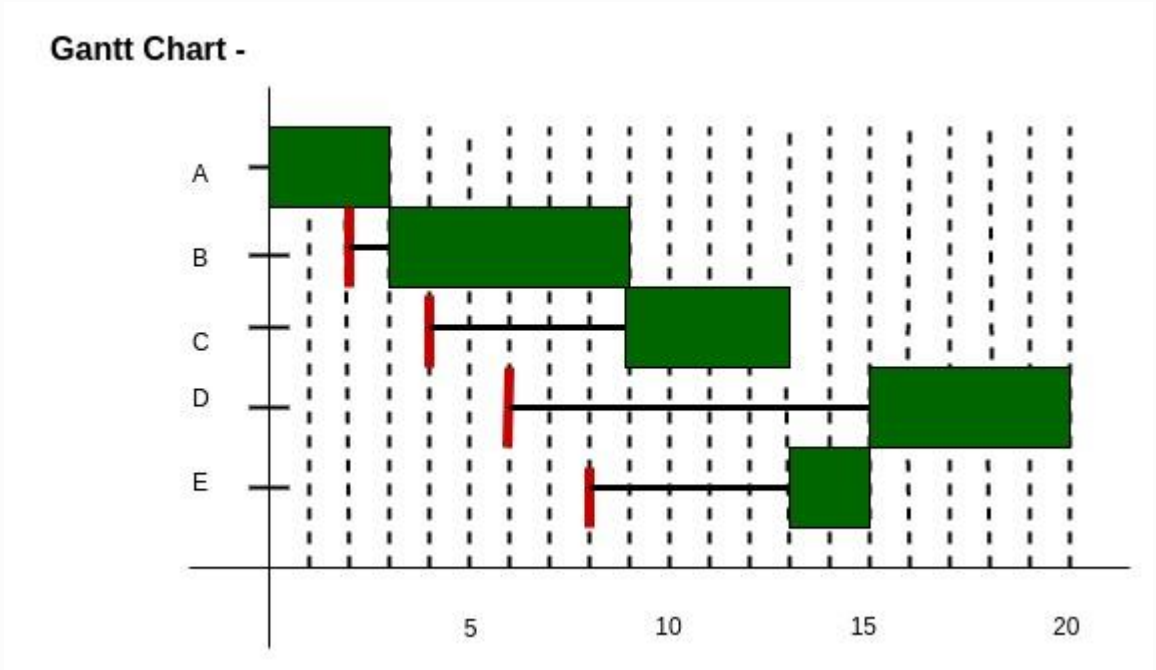
用HRRN算法进行调度，其Gantt图应为?

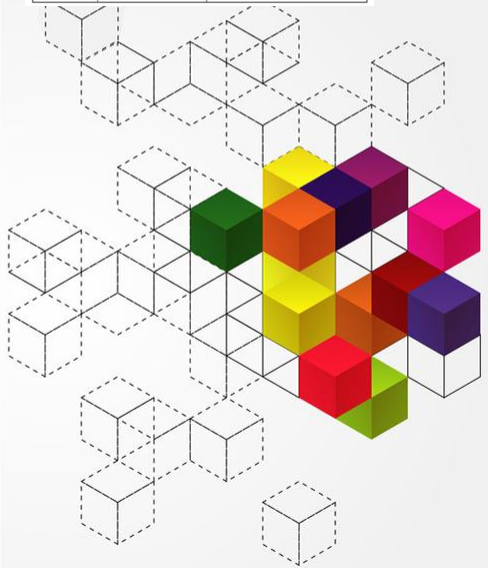


零、CPU调度算法1回顾

- **HRRN（高相应比优先）算法**

Thread	Arrival TIme	CPU Burst Length
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2



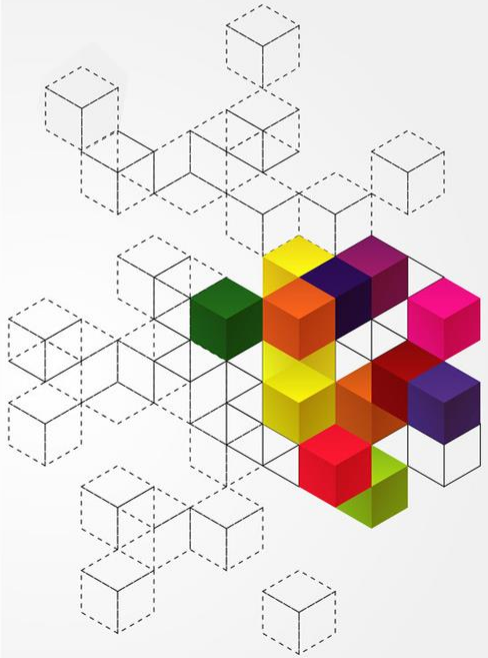


内容纲要

5.4 调度算法Part2

一、RR调度算法原理

二、RR调度算法性能分析



一、RR调度算法原理

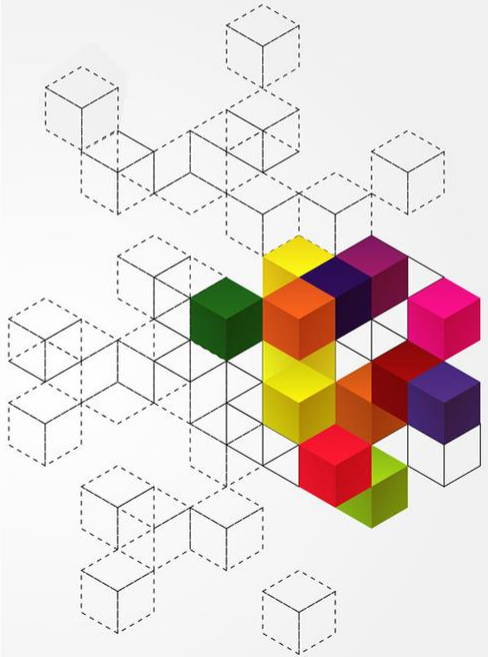
- **轮转调度（Round Robin）**
  - 基本思想：进程轮流使用CPU

RR算法基本做法：

① 系统将所有就绪进程按到达时间的先后次序排成一个队列

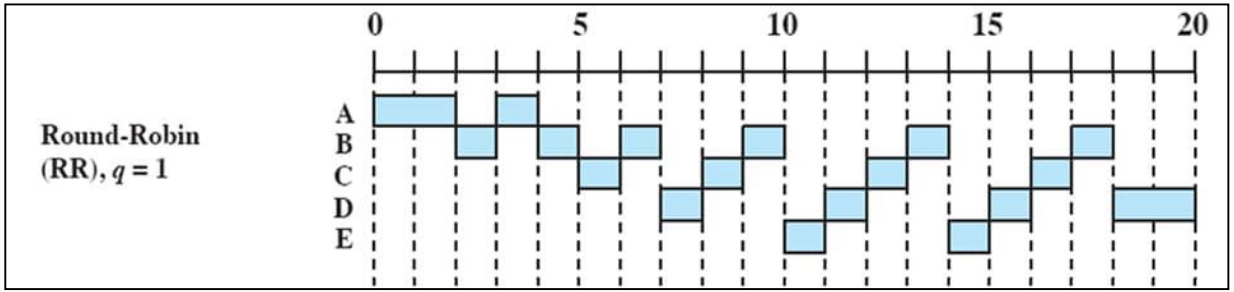
② 进程调度程序总是选择就绪队列中第一个进程执行，即先来先服务的原则，但仅能运行一个时间片

③ 在使用完一个时间片后，即使进程并未完成其运行，它也必须释放出（被剥夺）处理机给下一个就绪的进程，而被剥夺的进程返回到就绪队列的末尾重新排队，等候再次运行

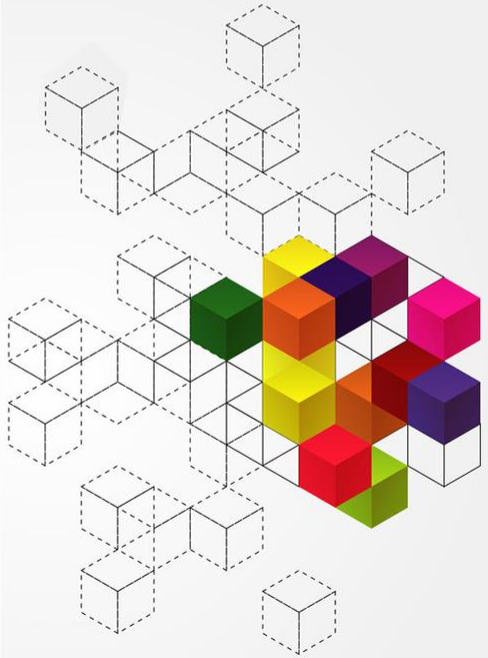


一、RR调度算法原理

- **轮转调度调度算法示意**



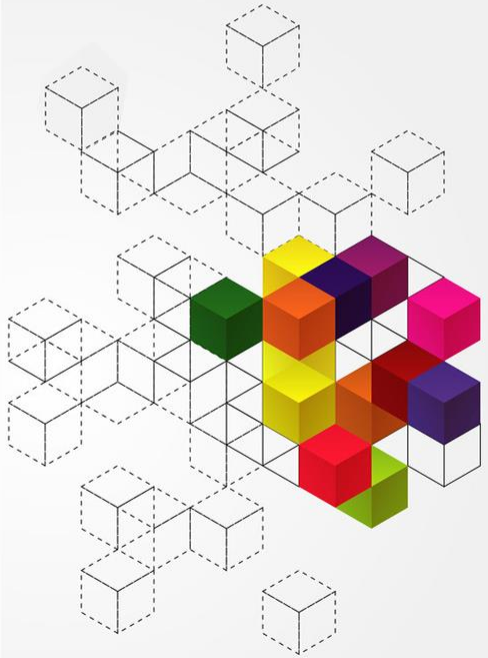
关键参数：时间片（示例中为1）



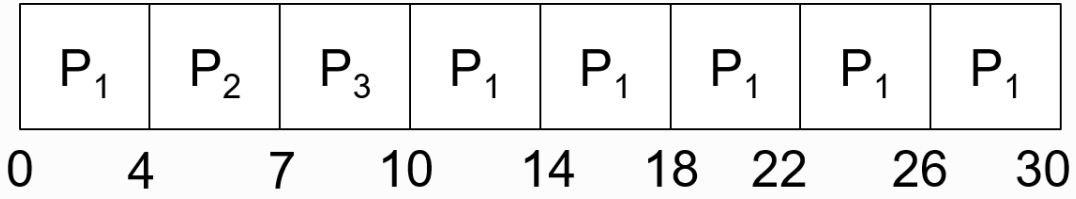
一、RR调度算法原理

示例（Time Quantum=4）

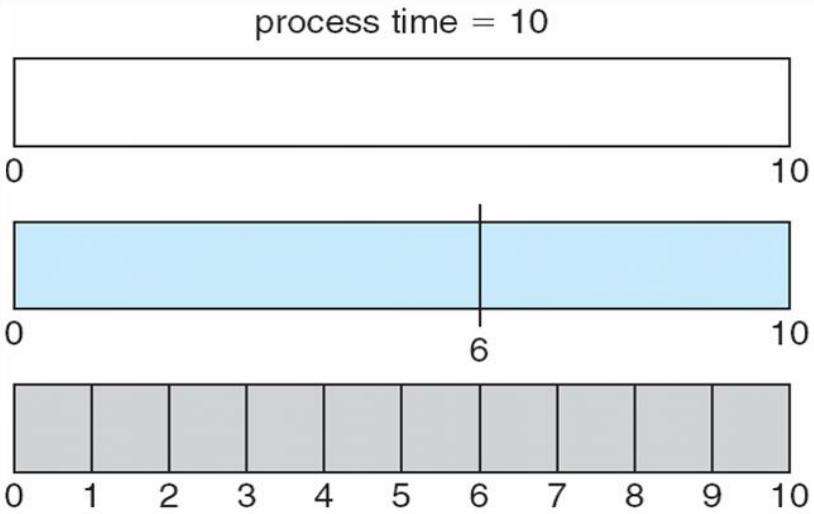
Process	Burst Time
P1	24
P2	3
P3	3



轮转调度作用于示例的甘特图

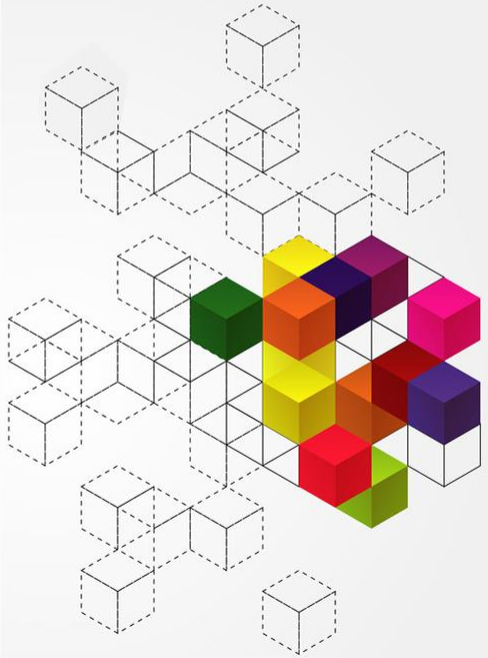


二、RR调度算法分析



quantum	context switches
12	0
6	1
1	9

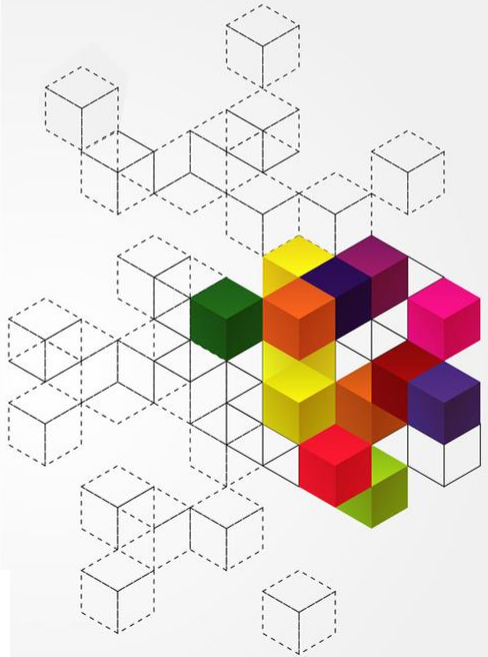
时间片越短，上下文切换次数越大

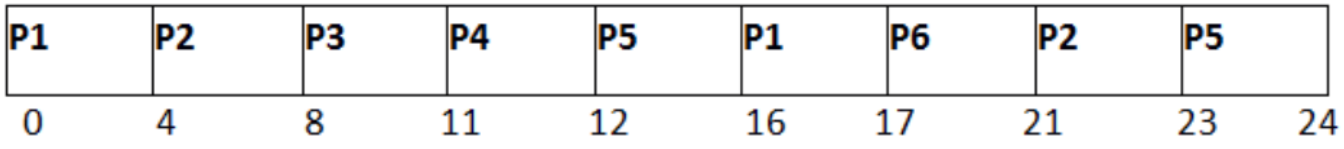


二、RR调度算法分析

6个进程，时间片=4，轮转调度的Gantt图是?

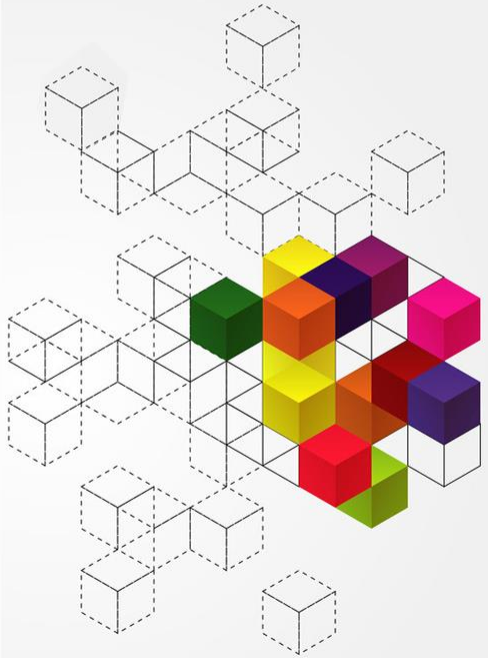
Process ID	Arrival Time	Burst Time
1	0	5
2	1	6
3	2	3
4	3	1
5	4	5
6	6	4





本讲小结

- RR调度算法原理
- RR调度算法评价

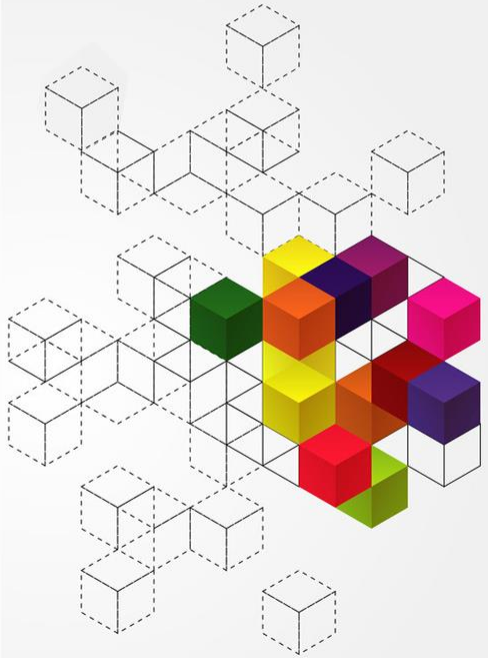


内容纲要

5.5 调度算法Part3

一、多级队列调度

二、多级反馈队列调度

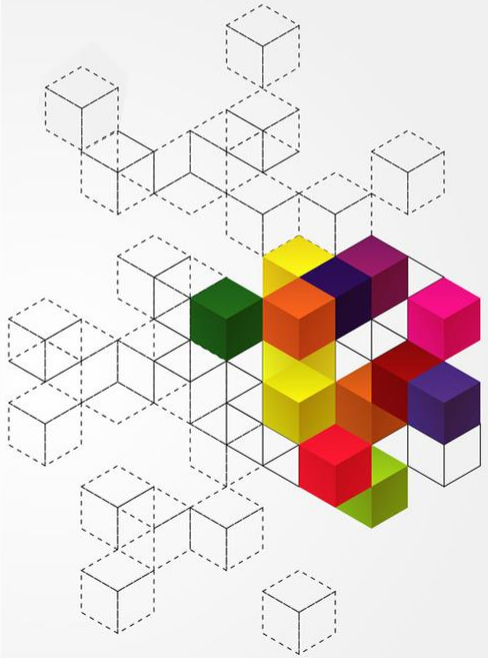


一、多级队列调度

- Why Multilevel Queue？

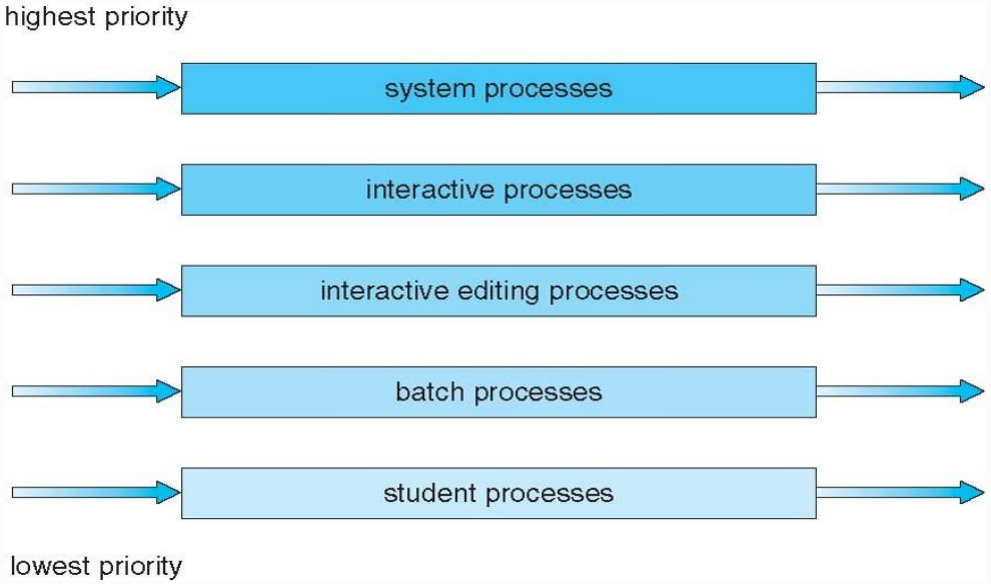
用单个队列管理所有就绪进程不是个好主意
-不同类型的进程放在同一个队列，调度算法检索效率低

Solution: 将进程分门别类，用不同的队列进行管理

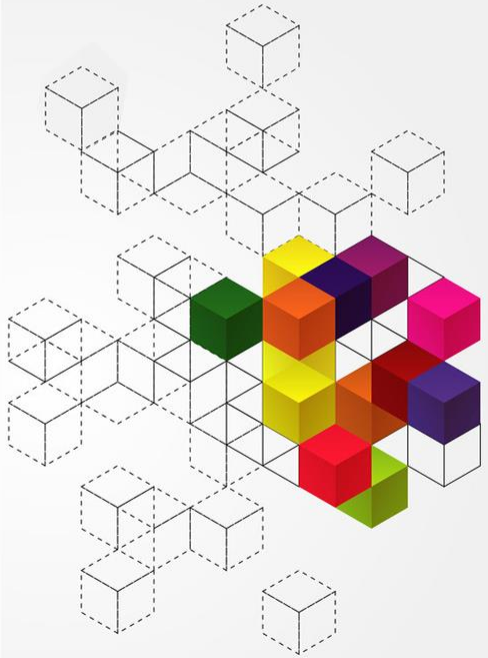


一、多级队列调度

- 多级队列样例



不同类型进程放入不同的队列；
每个队列中的进程赋予固定的优先级

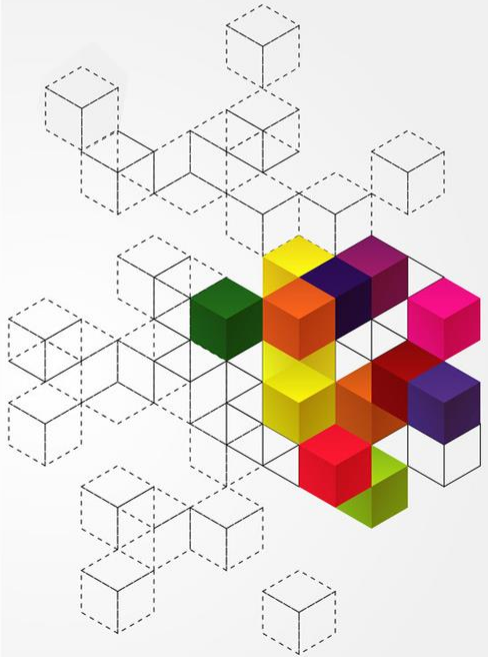


一、多级队列调度

- 多级队列调度实现思路

设计考虑1：不同队列可由不同的调度算法管理

例如: 将进程分为前台交互就绪队列和后台批处理就绪队列；前台进程调度使用RR算法，后台进程调度采用FCFS算法



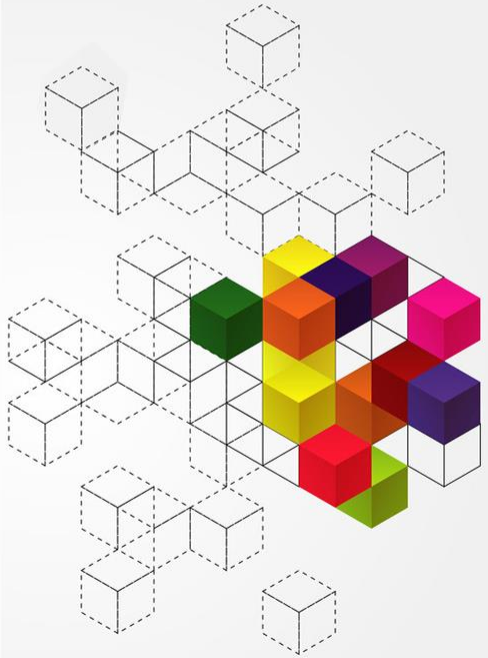
一、多级队列调度

- 多级队列调度实现思路

设计考虑1：不同队列可由不同的调度算法管理

设计考虑2：为不同队列进程分配不同大小的时间配额

例如: 前台进程队列使用80%的CPU时间，后台进程队列享用20%的CPU时间

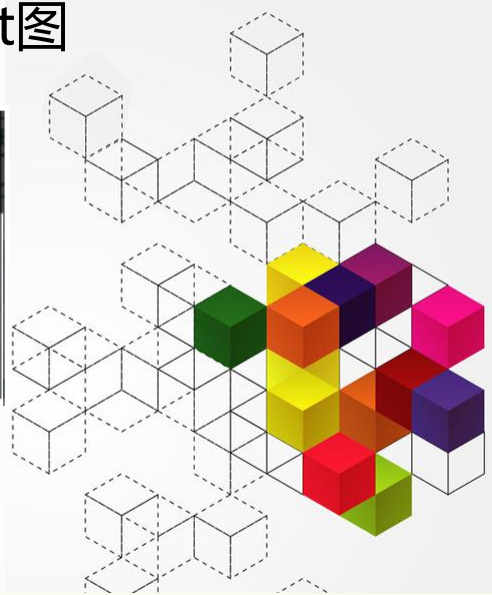


主观题 6分

设置

队列1的优先级高于队列2，队列1采用RR算法，时间片=2
队列2采用FCFS算法。请给出多级队列调度的Gantt图

Process	Arrival Time	CPU Burst Time	Queue Number
P1	0	4	1
P2	0	3	1
P3	0	8	2
P4	10	5	1



正常使用主观题需2.0以上版本雨课堂

作答

一、多级队列调度

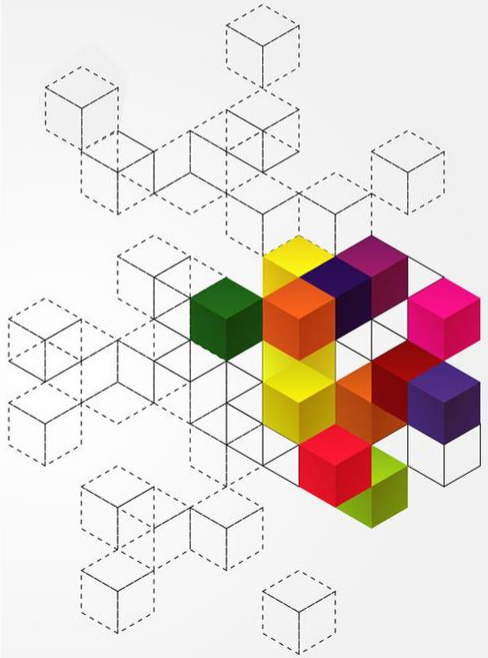
- 多级队列调度的优点与问题

优势

- 不同类型队列自由选择调度算法（调度策略灵活）
- 可以为不同类别的进程队列分配不同的时间配额（CPU时间分配策略灵活）

问题

- 进程固定于特定优先级队列（不便于动态调控）
- 高优先级队列中的进程具有优势，低优先级队列中的进程面临饥饿问题的概率高（Starvation）



二、多级反馈队列调度

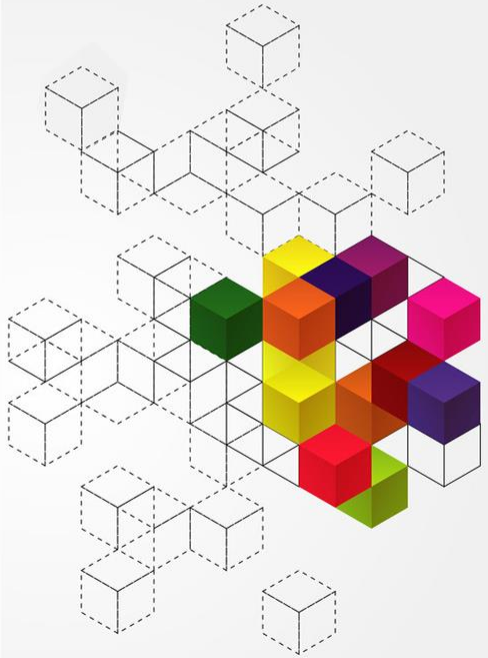
MLQ

问题1	问题2
进程固定于特定队列（不灵活）	低优先级进程饥饿概率高（不公平）

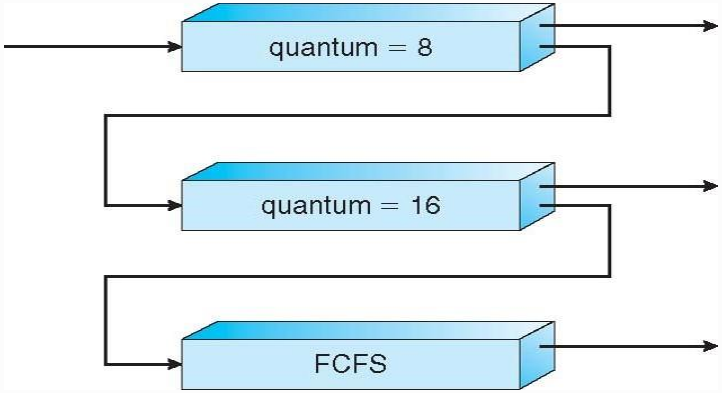
MLFQ

策略1	策略2
令进程可在不同队列间移动（增加灵活性）	Aging（进程老化）：对等待过久的进程，提升其优先级（避免饥饿，保证公平）

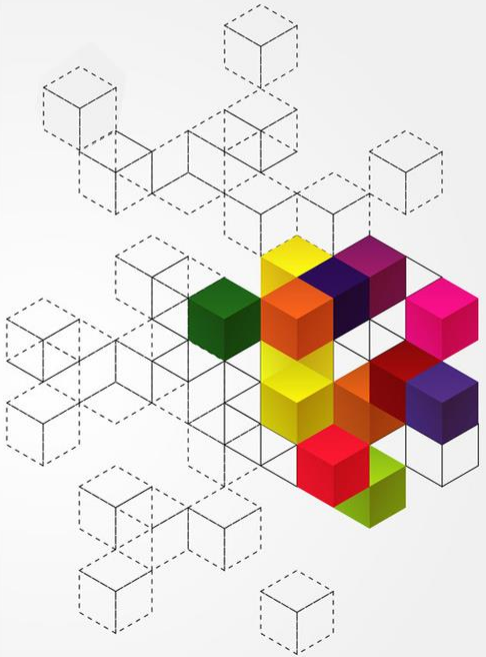
Multi-Level Feedback Queue



二、多级反馈队列调度



- **多级反馈队列调度的示意**
  - 3级队列
  - Q_0 – RR with time quantum 8 milliseconds
  - Q_1 – RR time quantum 16 milliseconds
  - Q_2 – FCFS



填空题 4分

设置

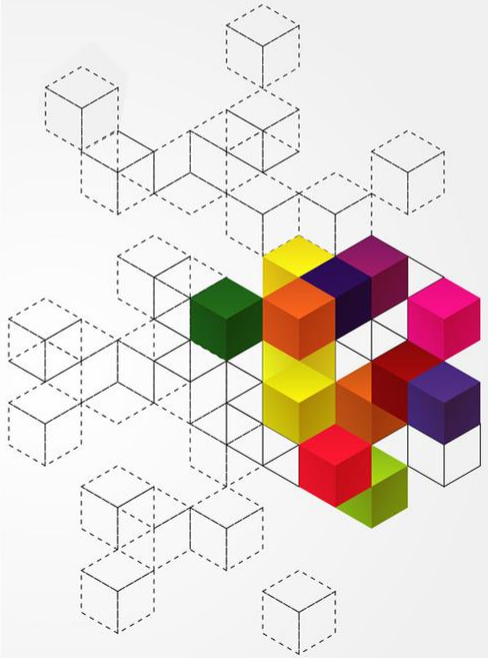
操作系统内有1个执行CPU周期长度为40秒的进程，若系统采用多级反馈队列调度，每级队列均采用RR算法，最高优先级队列的时间片=2秒，相向每一级的时间片增加5秒。请问进程会被中断 [填空1] 次，最终在第 [填空2] 级队列执行结束?

正常使用填空题需3.0以上版本雨课堂

作答

本讲小结

- 多级队列调度
- 多级反馈队列调度



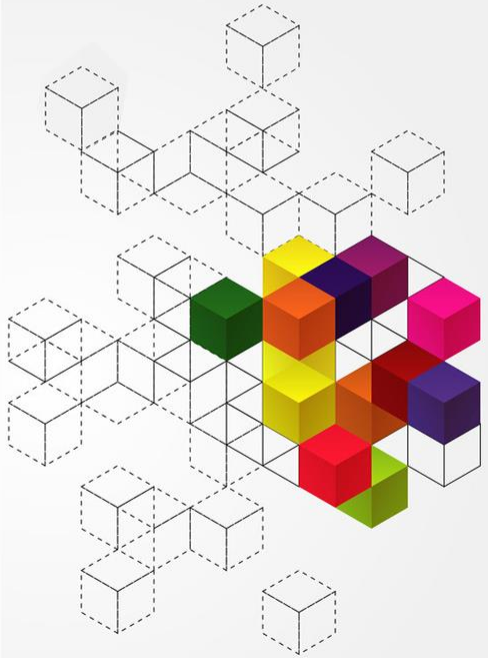
内容纲要

5.6 调度实现示例

一、Linux调度算法

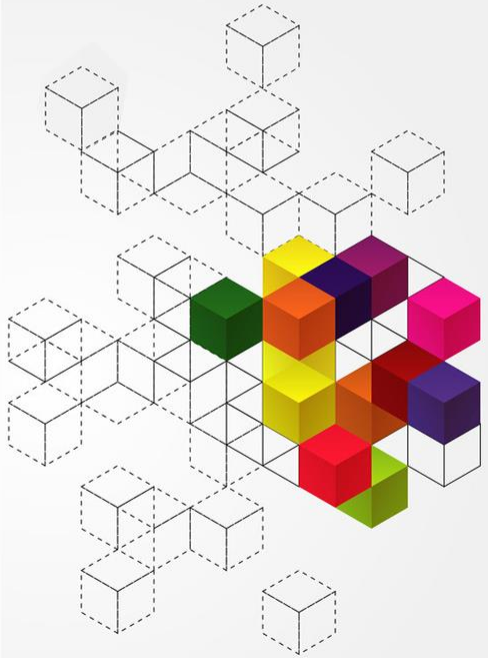
二、Windows调度

三、Solaris调度



一、Linux调度

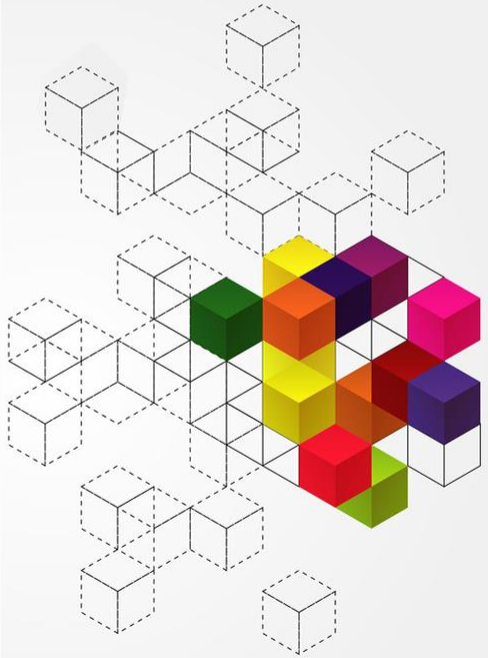
- Scheduler Data Structure (Linux 2.6+)
  - Expire Array and Active Array



一、Linux调度

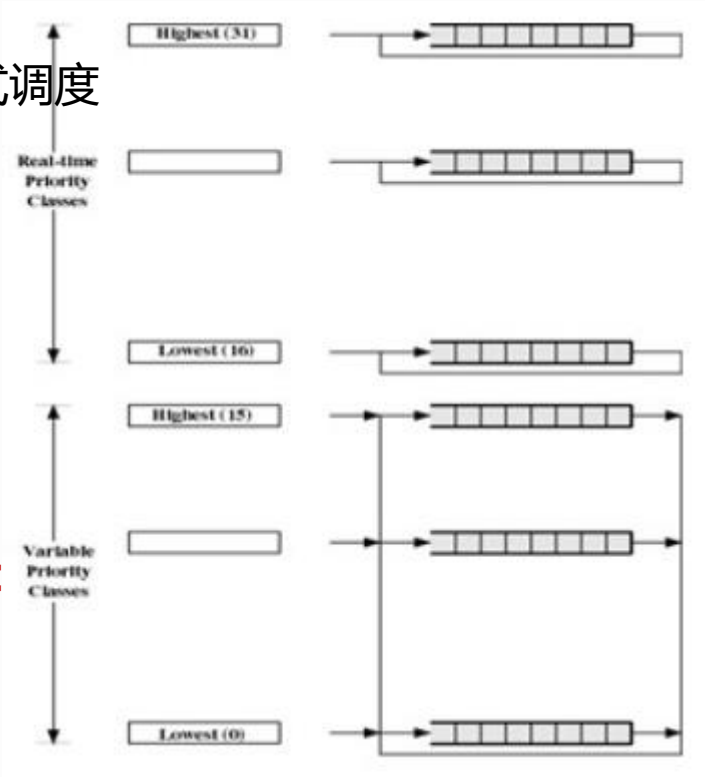
- Linux调度优先级
  - 用优先数0-140表示
  - 优先数越大，优先级越低

numeric priority	relative priority		time quantum
0	highest	real-time tasks	200 ms
•			
•			
•			
99			
100		other tasks	
•			
•			
•			
140	lowest		10 ms



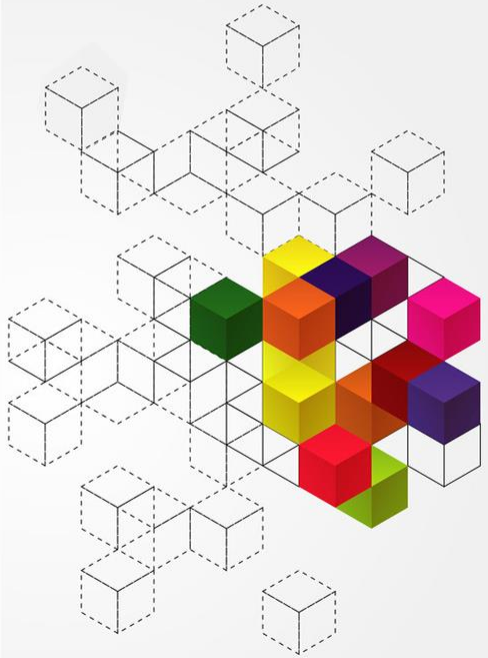
二、Windows调度

- Windows NT-based OS use a multilevel feedback queue.
  - 基于优先级的抢占式调度
  - 使用多级反馈队列



32个优先级（1-15：variable class, 16-31:real time）

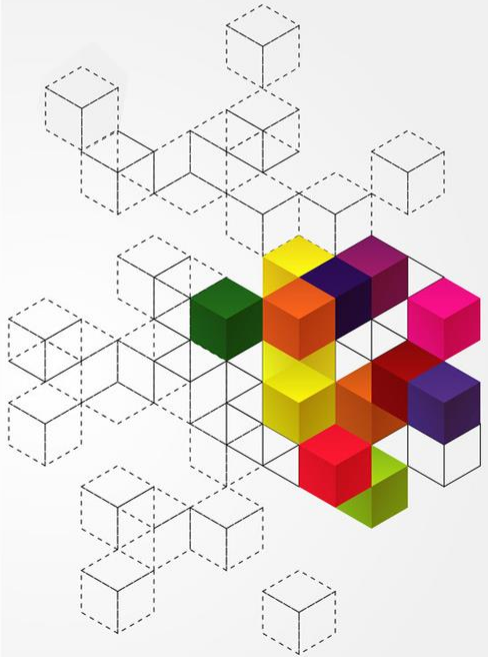
优先级0：赋予memory-management thread



二、Windows调度

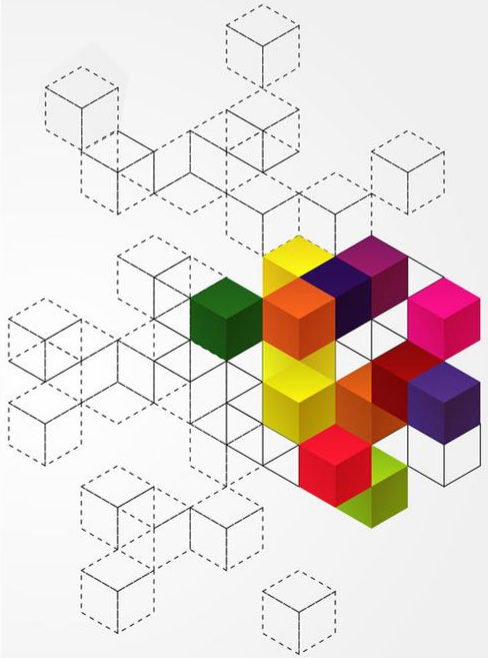
- Windows NT内核的优先级设定
  - 两个维度：base priority and relative priority

	real-time	high	above normal	normal	below normal	idle priority
time-critical	31	15	15	15	15	15
highest	26	15	12	10	8	6
above normal	25	14	11	9	7	5
normal	24	13	10	8	6	4
below normal	23	12	9	7	5	3
lowest	22	11	8	6	4	2
idle	16	1	1	1	1	1



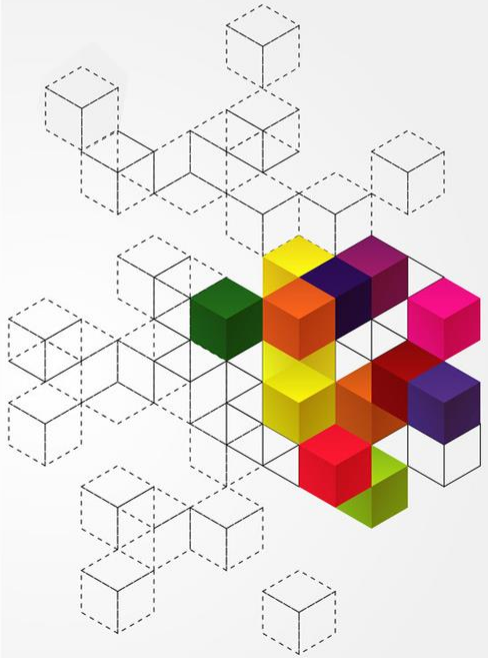
二、Windows调度

- Windows NT内核的线程优先级
  - 每个线程初始会被赋予一个base priority
  - 在动态优先级的范围内，当线程的时间片用完后，其优先级会被调低
    - 但不会低于base priority



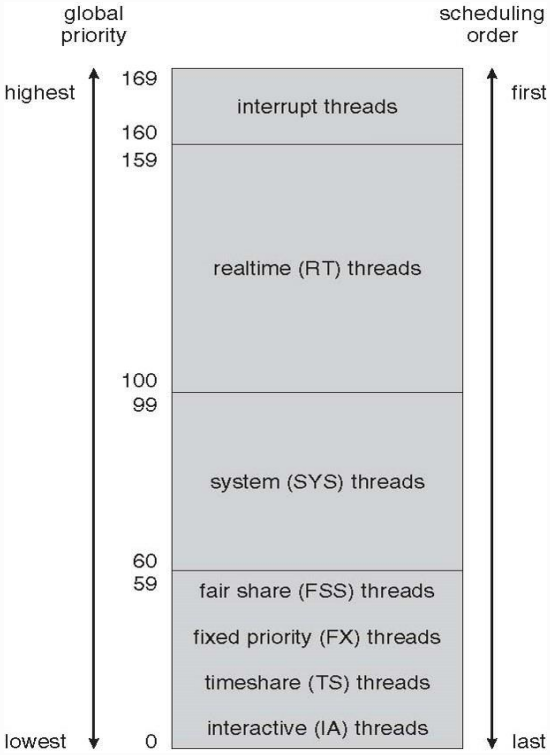
二、Windows调度

- Windows NT内核的线程优先级
  - 当线程从阻塞状态被唤醒后，其优先级会被调高
  - 优先级调高的幅度，根据线程阻塞时等待的事件不同而有区别
    - A thread that was waiting for keyboard I/O would get a large increase
    - A thread that was waiting for a disk operation would get a moderate increase
  - CPU密集性任务的优先级在这个过程中会被逐渐调低，而I/O密集性任务的优先级会逐渐增高

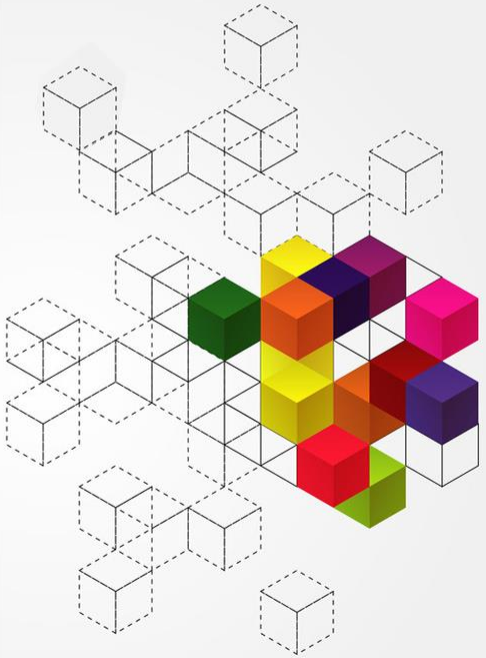


三、Solaris调度

- Solaris采用多级反馈队列调度



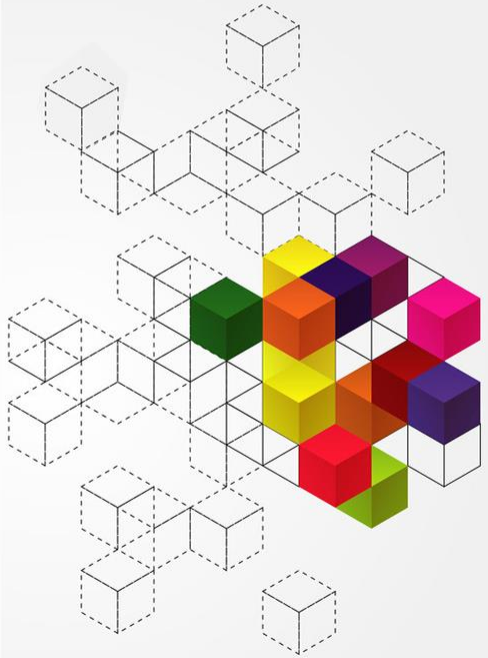
- **6类线程**
  - **Time sharing (default) 分时**
  - **Interactive 交互**
  - **Real time 实时**
  - **System 系统**
  - **Fair Share 公平轮转**
  - **Fixed priority 固定优先级**



三、Solaris调度

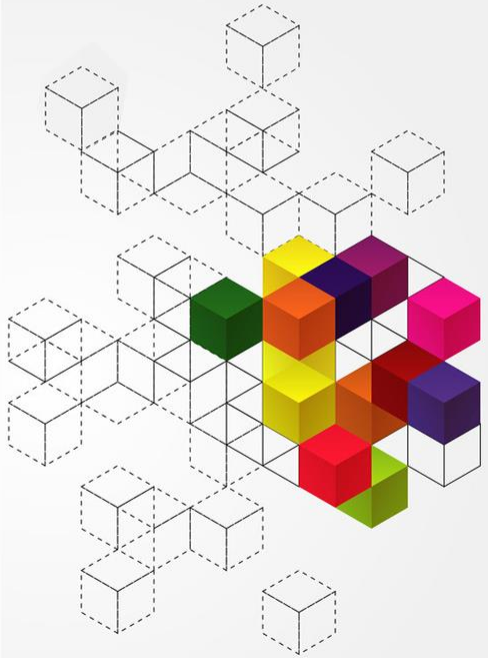
- 进程优先级的动态调整

priority	time quantum	time quantum expired	return from sleep
0	200	0	50
5	200	0	50
10	160	0	51
15	160	5	51
20	120	10	52
25	120	15	52
30	80	20	53
35	80	25	54
40	40	30	55
45	40	35	56
50	40	40	58
55	40	45	58
59	20	49	59



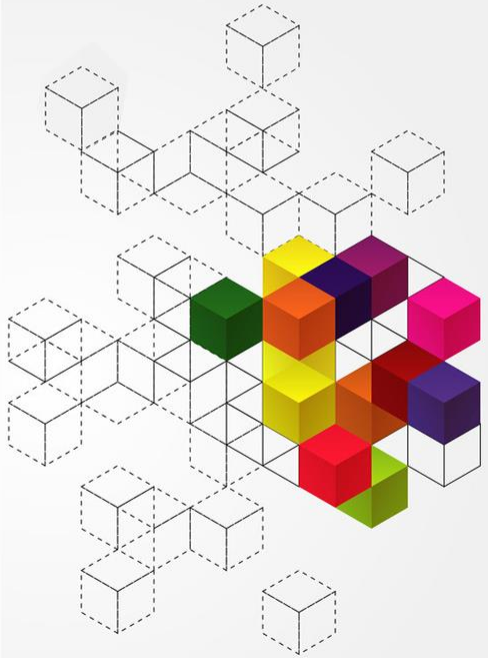
本讲小结

- Linux调度实现
- Windows调度
- Solaris调度



CPU调度课后习题

- 请对Linux的调度算法发展史做个调研，写成一篇短文。



E、CPU调度练习

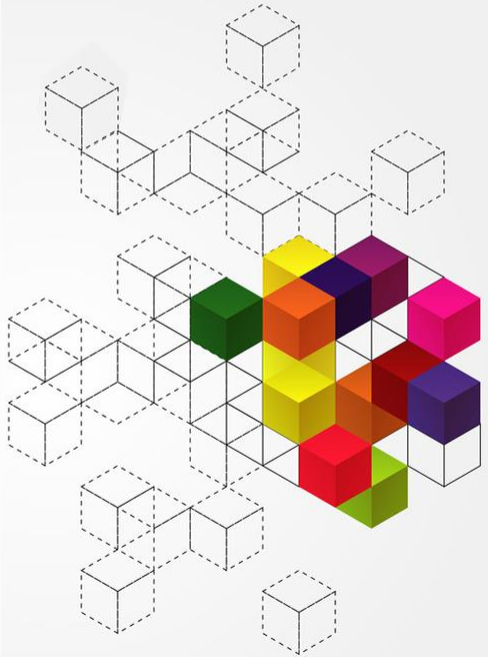
- 进程调度是从（ ）中选择一个进程投入运行。

A. 就绪队列

B. 等待队列

C. 作业后备队列

D. 提交队列



E、CPU调度练习

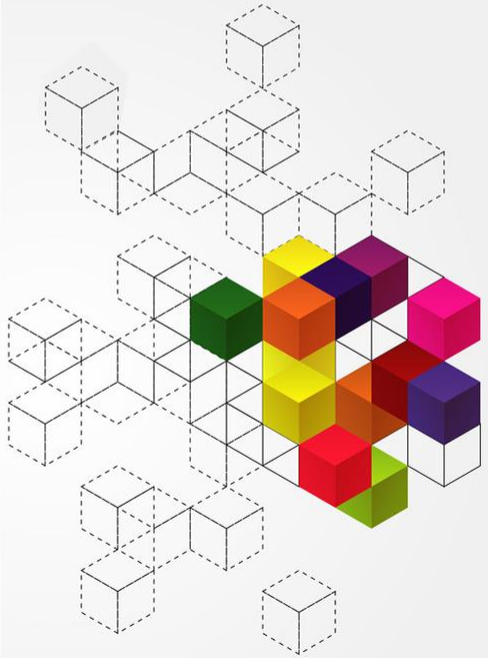
- 操作系统短程调度主要负责（ ）。

A. 选作业进入内存

B. 选一进程去占用CPU

C. 选择进程交换到外存

D. 撤销一进程



E、CPU调度练习

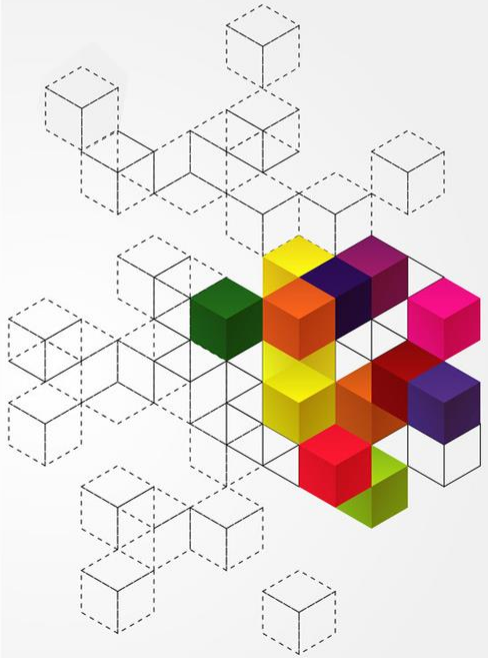
- “可抢占”和“不可抢占”的优先级调度算法相比，（ ）。

A. 前者实现开销小

B. 前者实现开销大

C. 两者实现开销大致相同

D. 两者实现开销不好比较



E、CPU调度练习

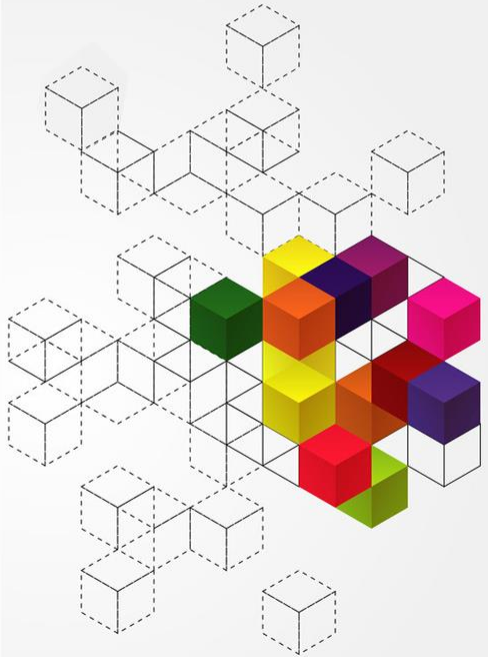
- （ ）优先权是在创建进程时确定的，确定之后在整个进程运行期间不再改变。

A. 先来先服务

B. 静态

C. 动态

D. 短作业



E、CPU调度练习

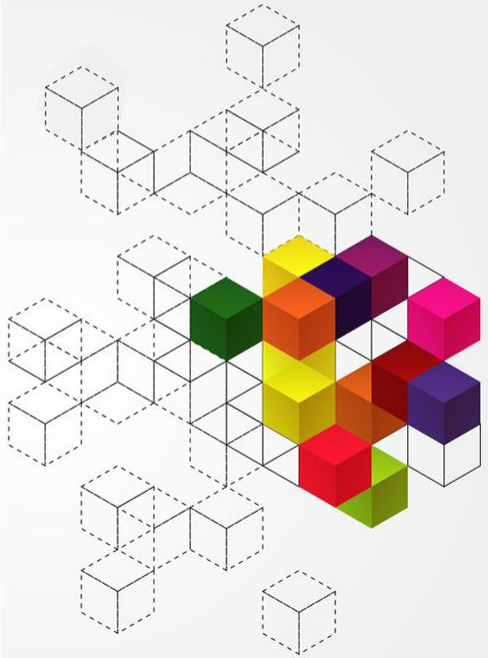
- 若进程P一旦被唤醒就能够投入运行，系统可能为（ ）。

A. 分时系统，进程P的优先级最高

B. 抢占调度方式，就绪队列上的所有进程的优先级皆比P的低

C. 就绪队列为空队列

D. 抢占调度方式，P的优先级高于当前运行的进程



E、CPU调度练习

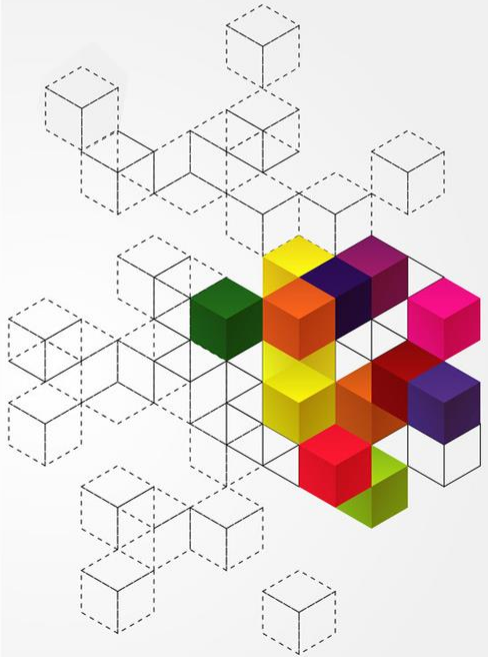
- 一个进程P被唤醒后，（ ）。

A. P就占有了CPU

B. P的PCB就被移到就绪队列的队首

C. P的优先级一定是最高

D. P的状态变为就绪



E、CPU调度练习

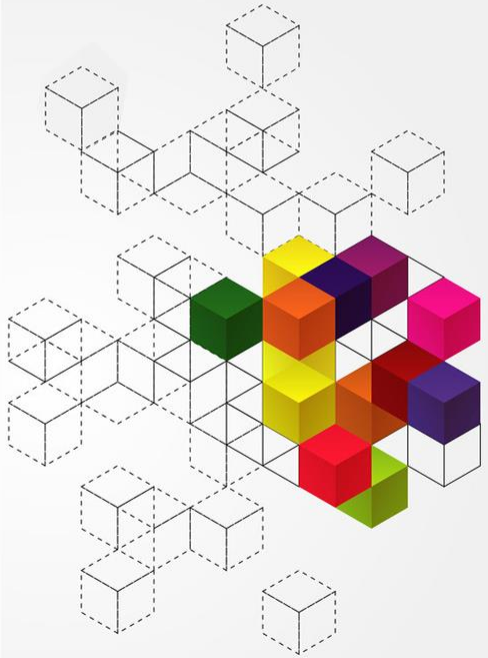
- 为照顾紧迫型作业，应采用（ ）。

A. 先来先服务调度算法

B. 短作业优先调度算法

C. 时间片轮转调度算法

D. 优先权调度算法



E、CPU调度练习

- 操作系统调度的基本对象是什么？

在未引入线程的操作系统中，调度的基本单位是进程；

在引入了线程概念的操作系统中，调度的基本单位是线程（内核级线程）