操作系统
第五讲

张涛

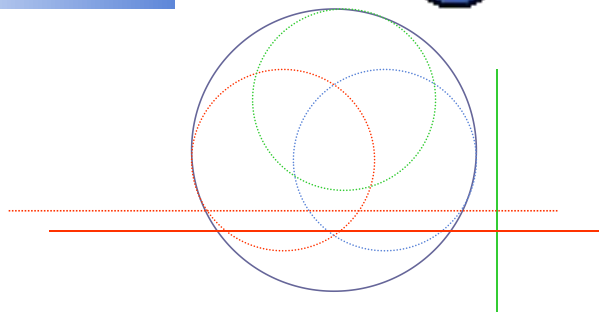
软件与微电子学院

Review

进程概念的引入

进程的表示和状态转换

进程的控制



3.4 进程调度

处理机调度的基本概念

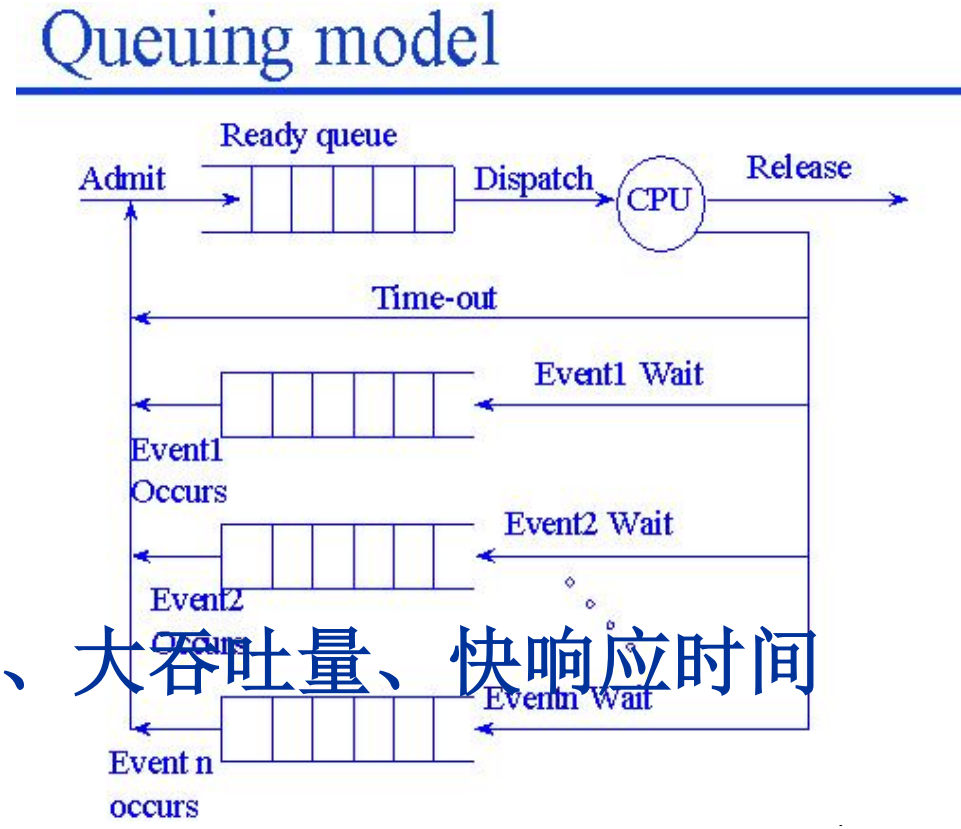
调度算法

实时调度

多处理机调度

3.4.1 处理机调度的基本概念

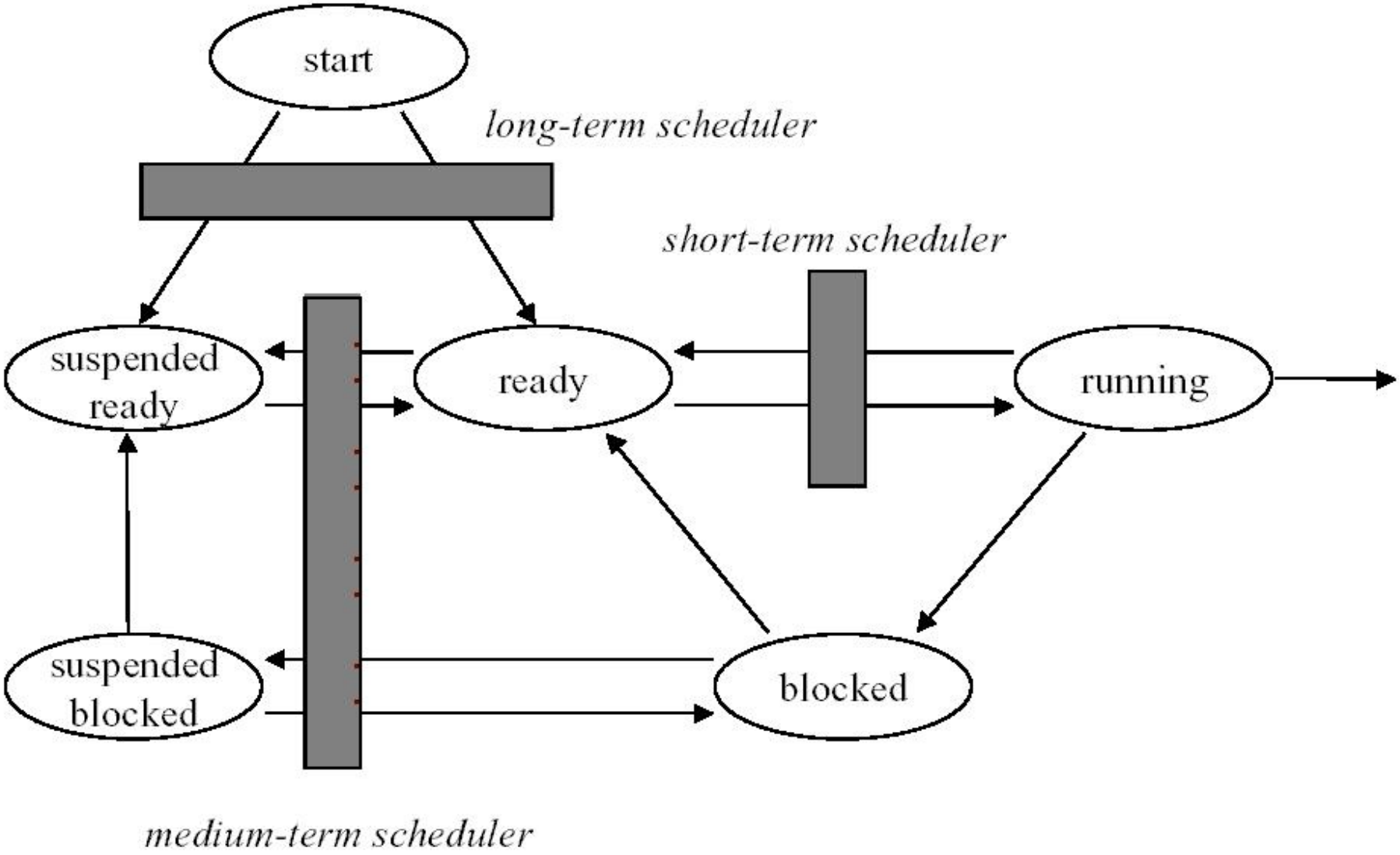
- 要解决的问题
  - WHAT：按什么原则分配CPU
    - —进程调度算法
  - WHEN：何时分配CPU
    - —进程调度的时机
  - HOW：如何分配CPU
    - —进程调度方式

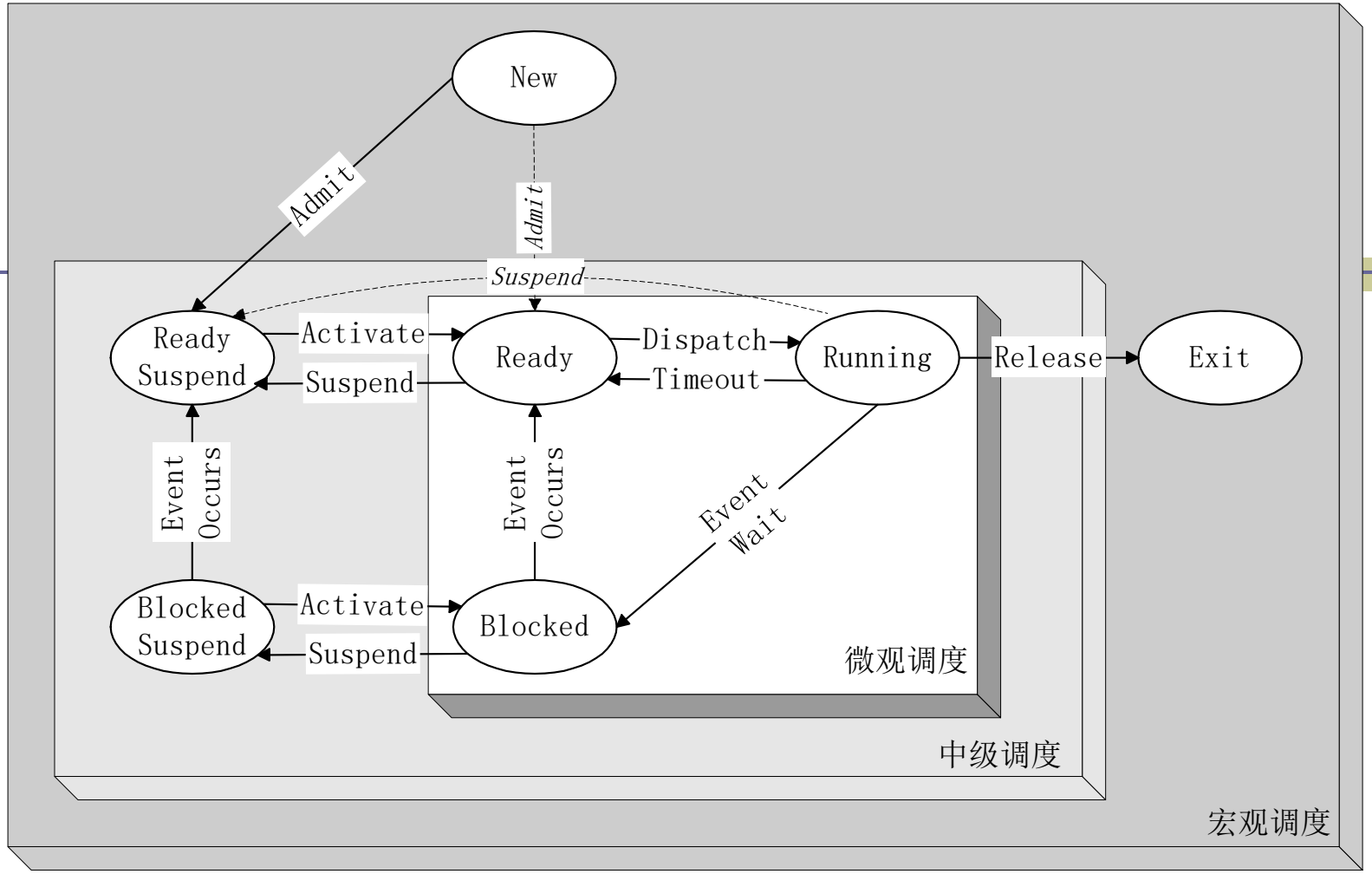


- 目标：高CPU的利用率、大吞吐量、快响应时间

3.4.1.1 调度的类型

- 处理机调度可以分成三级：
  - 高级调度
    - 也称为作业调度或宏观调度，从用户工作流程的角度，一次提交的若干个流程，其中每个程序按照进程调度。时间上通常是分钟、小时或天。New－>Ready suspend, Running －>Exit
  - 中级调度
    - 涉及进程在内外存间的交换，从存储器资源管理的角度来看，把进程的部分或全部换出到外存上，将当前进程所需部分换入到内存。 Ready <－>Ready suspend, Blocked <－>Blocked suspend
  - 低级调度，
    - 也称进程调度、微观调度，从处理机资源分配的角度来看，处理机需要经常选择就绪进程或线程进入运行状态。Ready <－>Running


start
long-term scheduler
short-term scheduler
suspended ready
ready
running
suspended blocked
blocked
medium-term scheduler



- **进程调度的任务**是控制协调进程对CPU的竞争，即按一定的调度算法从就绪队列中选中一个进程，把CPU的使用权交给被选中的进程
- 微观调度，低级调度，Ready <–>Running

王海鹏印

3.4.1.2 调度的性能准则

- 面向用户的调度性能准则
  - 周转时间：作业从提交到完成（得到结果）所经历的时间。包括：在收容队列中等待，CPU上执行，就绪队列和阻塞队列中等待等。
  - 平均周转时间

$$T = \left(\sum_{i=1}^{n} T_i\right) \Big/ n$$

  - 平均带权周转时间

$$w = \left(\sum_{i=1}^{n} w_i\right) \Big/ n = \left(\sum_{i=1}^{n} \frac{T_i}{t_{Ri}}\right) \Big/ n$$

n —— 作业流中的作业数

T_i —— 第i个作业周转时间

t_{Ri} —— 作业i的运行时间

- 响应时间：用户输入一个请求（如击键）到系统给出首次响应（如屏幕显示）的时间——分时系统
- 截止时间：开始截止时间和完成截止时间
- 公平性：不因作业或进程本身的特性而使上述指标过分恶化。如长作业等待很长时间。
- 优先级：可以使关键任务达到更好的指标。

面向系统的调度性能准则

- 吞吐量：单位时间内所完成的作业数，跟作业本身特性和调度算法都有关系——批处理系统
- 处理机利用率：——大中型主机
- 各种设备的均衡利用：如CPU繁忙的作业和I/O繁忙的作业搭配——大中型主机

调度算法本身的调度性能准则

- 易于实现
- 执行开销比

调度算法设计目标

All systems

Fairness - giving each process a fair share of the CPU
Policy enforcement - seeing that stated policy is carried out
Balance - keeping all parts of the system busy

Batch systems

Throughput - maximize jobs per hour
Turnaround time - minimize time between submission and termination
CPU utilization - keep the CPU busy all the time

Interactive systems

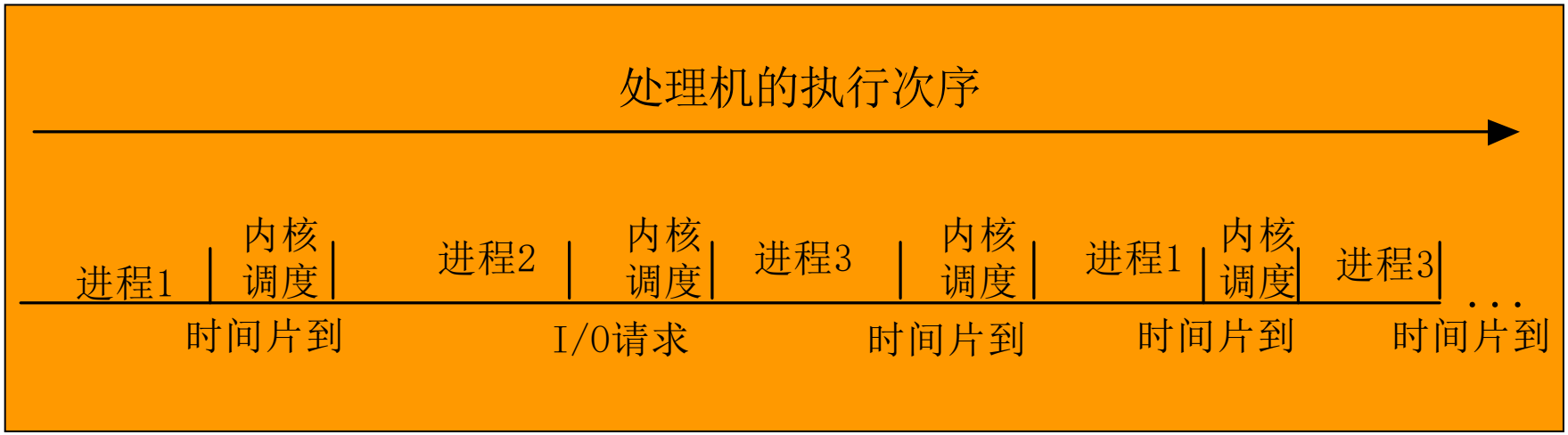
Response time - respond to requests quickly
Proportionality - meet users' expectations

Real-time systems

Meeting deadlines - avoid losing data
Predictability - avoid quality degradation in multimedia systems

3.4.1.3 进程调度的时机

- 现运行进程完成任务正常结束或因出现错误异常结束；
- 时间片到 (按时间片运行)；
- 进程提出I/O请求 —— 阻塞，调新进程；
- 执行原语操作，进入阻塞状态；
- 具有更高优先级的进程进入就绪队列，要求使用处理机（可剥夺调度）。



3.4.1.4 进程调度的方式

- 非剥夺调度(nonpreemptive scheduling)
  - 某一进程被调度运行后，除非由于它自身的原因不能运行，否则一直运行下去。
- 剥夺调度(preemptive scheduling)
  - 当有比正在运行的进程优先级更高的进程就绪时，系统可强行剥夺正在运行进程的CPU，提供给具有更高优先级的进程使用。

3.4.2 调度算法

- 先来先服务
- 短作业优先
- 时间片轮转算法
- 基于优先级的调度算法
- 多级队列算法
- 多级反馈队列算法

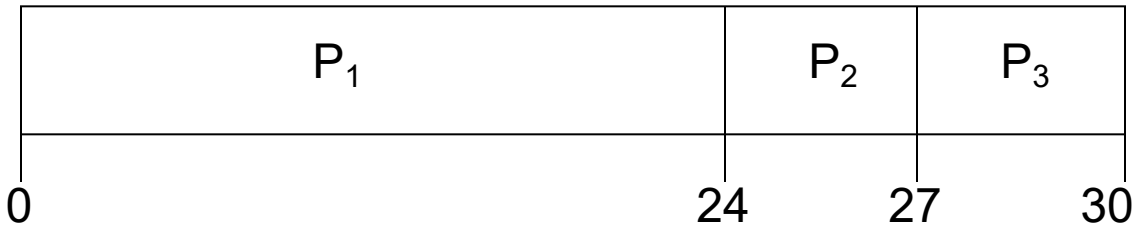
先来先服务
FCFS, First Come First Service

- 按照作业提交或进程变为就绪状态的先后次序分派CPU；
- 当前作业或进程占用CPU，直到执行完或阻塞，才出让CPU（非抢占方式）；在作业或进程唤醒后（如I/O完成），并不立即恢复执行，通常等到当前作业或进程出让CPU。
- 特点：
  - 比较有利于长作业，而不利于短作业。
  - 有利于CPU繁忙的作业，而不利于I/O繁忙的作业。
  - 最简单的算法。

- Example:

Process	Burst Time
P_1	24
P_2	3
P_3	3

- Suppose that the processes arrive in the order: P_1 , P_2 , P_3 The Gantt Chart（甘特图） for the schedule is:

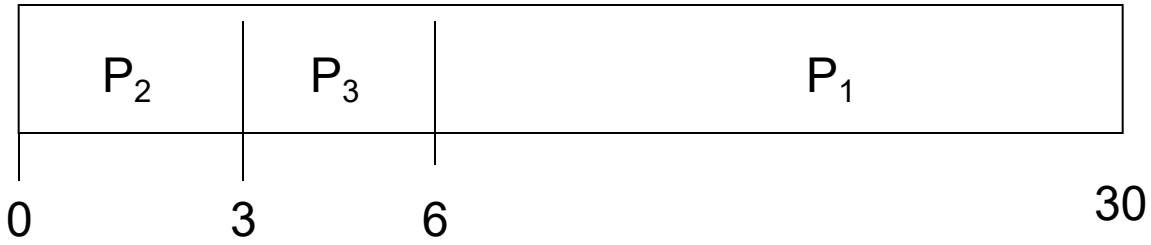


- Waiting time for P_1 = 0; P_2 = 24; P_3 = 27
- Average waiting time: (0 + 24 + 27)/3 = 17
- ***Convoy effect***: short process behind long process

Suppose that the processes arrive in the order

$$P_2 , P_3 , P_1 .$$

- The Gantt chart for the schedule is:



- Waiting time for $P_1 = 6; P_2 = 0; P_3 = 3$
- Average waiting time: (6 + 0 + 3)/3 = 3
- Much better than previous case.

最短作业优先
SJF, Shortest Job First

- 对预计执行时间短的作业（进程）优先分派处理机。通常后来的短作业不抢先正在执行的作业。
- 又称为“最短进程优先”SPN（Shortest Process Next）；
- SJF对于给定的进程集合，平均周转时间最小

Example of SJF

Process	Arrival Time	Burst Time
P1	0.0	7
P2	2.0	4
P3	4.0	1
P4	5.0	4

Non-Preemptive

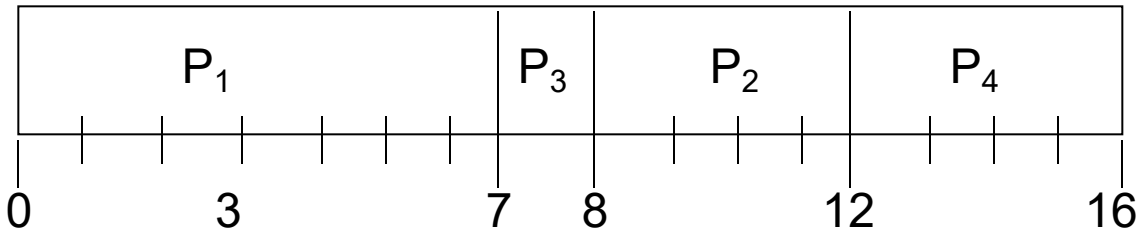
Preemptive

Please give the Waiting Time of each process

Example of Non-Preemptive SJF

Process	Arrival Time	Burst Time
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

- SJF (non-preemptive)

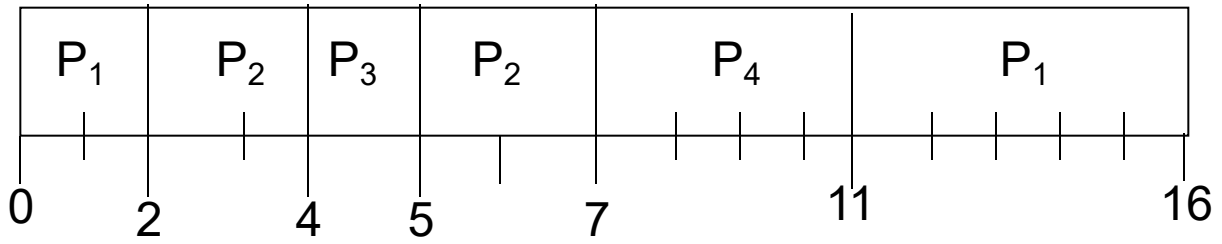


- Average waiting time = (0 + 6 + 3 + 7)/4 = 4

Example of Preemptive SJF

Process	Arrival Time	Burst Time
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

- SJF (preemptive)



- Average waiting time = (9 + 1 + 0 +2)/4 = 3

SJF的特点

- 优点：
  - 比FCFS改善平均周转时间和平均带权周转时间
  - 提高系统的吞吐量；
- 缺点：
  - 对长作业非常不利，可能长时间得不到执行；
  - 未能依据作业的紧迫程度来划分执行的优先级；
  - 难以准确估计作业（进程）的执行时间，影响调度性能。

作业情况 / 调度算法	进程名	A	B	C	D	E	平均
	到达时间	0	1	2	3	4	
	服务时间	4	3	5	2	4	
FCFS (a)	完成时间						
	周转时间						
	带权周转时间						
SJF (b)	完成时间						
	周转时间						
	带权周转时间						

Try your best!

SJF的变形

- 最短剩余时间优先，SRT，Shortest Remaining Time
  - 允许比当前进程剩余时间更短的进程来抢占
- 最高响应比优先HRRN（Highest Response Ratio Next）
  - 响应比R = (等待时间 + 要求执行时间) / 要求执行时间
  - 是FCFS和SJF的折衷

作业调度实例

（5）作业调度算法应用例子1

- 假设在单道批处理环境下有四个作业，已知它们进入系统的时间、估计运行时间

应用先来先服务、最短作业优先和最高响应比优先作业调度算法，分别计算出作业的平均周转时间和带权的平均周转时间

作业	进入时间	估计运行时间（分钟）
JOB1	8：00	120
JOB2	8：50	50
JOB3	9：00	10
JOB4	9：50	20

作业调度实例

作业	进入时间	估计运行时间（分钟）	开始时间	结束时间	周转时间（分钟）	带权周转时间
JOB1	8：00	120	8：00	10：00	120	1
JOB2	8：50	50	10：00	10：50	120	2.4
JOB3	9：00	10	10：50	11：00	120	12
JOB4	9：50	20	11：00	11：20	90	4.5
作业平均周转时间 T＝112.5 作业带权平均周转时间 W＝4.975					450	19.9

先来先服务调度算法计算结果

作业调度实例

作业	进入时间	估计运行时间（分钟）	开始时间	结束时间	周转时间（分钟）	带权周转时间
JOB1	8：00	120	8：00	10：00	120	1
JOB2	8：50	50	10：30	11：20	150	3
JOB3	9：00	10	10：00	10：10	70	7
JOB4	9：50	20	10：10	10：30	40	2
作业平均周转时间 T＝95 作业带权平均周转时间 W＝3.25					380	13

最短作业优先作业算法计算结果

作业调度实例

作业	进入时间	估计运行时间（分钟）	开始时间	结束时间	周转时间（分钟）	带权周转时间
JOB1	8：00	120	8：00	10：00	120	1
JOB2	8：50	50	10：10	11：00	130	2.6
JOB3	9：00	10	10：00	10：10	70	7
JOB4	9：50	20	11：00	11：20	90	4.5
作业平均周转时间 T = 102.5 作业带权平均周转时间 W =3.47					410	15.1

最高响应比优先作业算法计算结果

作业调度实例

（6）作业调度算法应用例子2

在两道环境下有四个作业

已知它们进入系统的时间、估计运行时间

系统采用短作业优先作业调度算法，作业被调度运行后不再退出

当一新作业投入运行后，可按照作业运行时间长短调整作业执行的次序

请给出这四个作业的执行时间序列，并计算出平均周转时间及带权平均周转时间

作业	进入时间	估计运行时间（分钟）
JOB1	**10：00**	**30**
JOB2	**10：05**	**20**
JOB3	**10：10**	**5**
JOB4	**10：20**	**10**

作业调度实例

作业	进入时间	估计运行时间（分钟）	开始时间	结束时间	周转时间（分钟）	带权周转时间
JOB1	10：00	30	10：00	11：05	65	2.167
JOB2	10：05	20	10：05	10：25	20	1
JOB3	10：10	5	10：25	10：30	20	4
JOB4	10：20	10	10：30	10：40	20	2
作业平均周转时间 T＝31.25 作业带权平均周转时间 W＝2.29					125	9.167

两道批处理系统中

最短作业优先作业算法计算结果

作业调度实例

四个作业的执行时间序列为：

JOB1：10：00—10：05，10：40—11：05

JOB2：10：05—10：25

JOB3：10：25—10：30

JOB4：10：30—10：40

两道批处理系统中

最短作业优先作业算法计算结果（续1）

作业调度实例

两道批处理系统中

最短作业优先作业算法分析过程

10：00，JOB1进入，只有一作业，JOB1被调入执行

10：05，JOB2到达，最多允许两作业同时进入

所以JOB2也被调入

- 内存中有两作业，哪一个执行？题目规定当一新作业运行后，可按作业运行时间长短调整执行次序
- 即基于优先数可抢占式调度策略

  优先数是根据作业估计运行时间大小来决定的

  由于JOB2运行时间（20分）比JOB1少

  （到10：05，JOB1还需25分钟）

  所以JOB2运行，而JOB1等待

作业调度实例

两道批处理系统中
最短作业优先作业算法分析过程（续1）

10：10，JOB3到达输入井，内存已有两作业

JOB3不能马上进入内存；

10：20，JOB4也不能进入内存

10：25，JOB2运行结束，退出，内存中剩下JOB1

输入井中有两作业JOB3和JOB4，如何调度？

- 作业调度算法：最短作业优先

  因此JOB3进入内存

  比较JOB1和JOB3运行时间

  JOB3运行时间短，故JOB3运行

  同样，JOB3退出后，下一个是JOB4

  JOB4结束后，JOB1才能继续运行

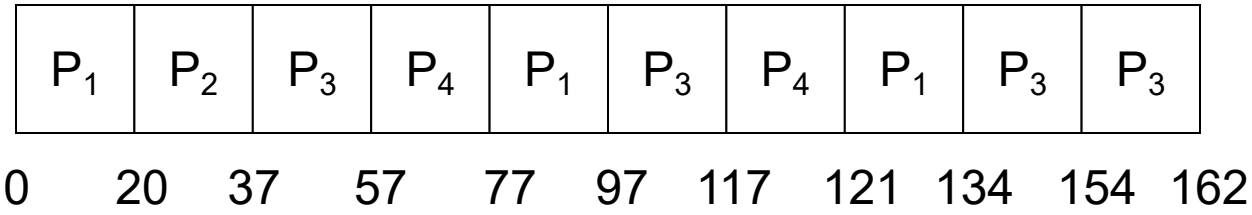
时间片轮转算法
RR, Round Robin

- 基本思路：
  - 通过时间片轮转，提高进程并发性和响应时间特性，从而提高资源利用率；
- 执行过程：
  - 将系统中所有的就绪进程按照FCFS原则，排成一个队列。
  - 每次调度时将CPU分派给队首进程，让其执行一个时间片。时间片的长度从几个ms到几百ms。
  - 在一个时间片结束时，发生时钟中断。调度程序暂停当前进程的执行，将其送到就绪队列的末尾，并通过上下文切换执行当前的队首进程。
  - 进程可以未使用完一个时间片，就出让CPU（如阻塞）。

Example: RR with Time Quantum = 20

Process	Burst Time
P_1	53
P_2	17
P_3	68
P_4	24

- The Gantt chart is:



- Typically, *higher average turnaround* than SJF, but *better response*.

时间片长度的确定

- 时间片长度变化的影响
  - 过长－>退化为FCFS算法，进程在一个时间片内都执行完，响应时间长。
  - 过短－>用户的一次请求需要多个时间片才能处理完，上下文切换次数增加，响应时间长。
- 对响应时间的要求：
  - T(响应时间)=N(进程数目)*q(时间片)
- 时间片长度的影响因素：
  - 就绪进程的数目：数目越多，时间片越小（当响应时间一定时）
  - 系统的处理能力：应当使用户输入通常在一个时间片内能处理完，否则使响应时间，平均周转时间和平均带权周转时间延长。

基于优先级的调度算法
Priority Scheduling

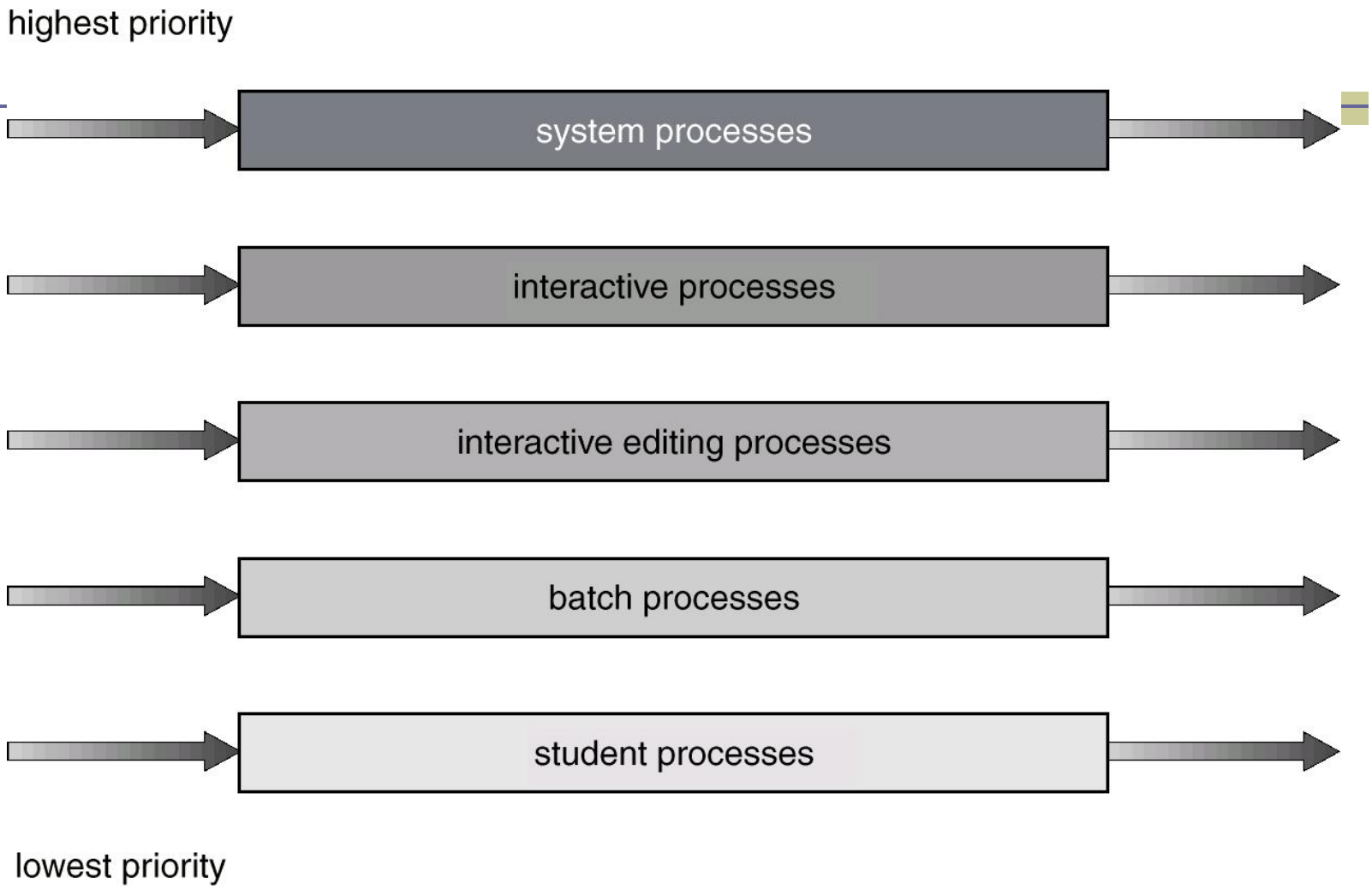
- 基本思想：
  - 系统为每个进程设置一个优先数（对应一个优先级），把所有的就绪进程按优先级从大到小排序，调度时从就绪队列中选择优先级最高的进程投入运行，仅当占用CPU的进程运行结束或因某种原因不能继续运行时，系统才进行重新调度。
- 剥夺方式：
  - 非剥夺（抢占）的优先级调度法
  - 可剥夺（抢占）的优先级调度法

优先级的类型

- **静态优先级**：创建进程时就确定，直到进程终止前都不改变。通常是一个整数。依据：
  - 进程类型（系统进程优先级较高）
  - 对资源的需求（对CPU和内存需求较少的进程优先级较高）
  - 用户要求（紧迫程度和付费多少）
- **动态优先级**：在创建进程时赋予的优先级，在进程运行过程中可以自动改变，以便获得更好的调度性能。如：
  - 在就绪队列中等待时间延长则优先级提高，使优先级较低的进程在等待足够的时间后，其优先级提高到可被调度执行；
  - 进程每执行一个时间片，就降低其优先级，从而一个进程持续执行时，其优先级降低到出让CPU。

多级队列算法
Multiple-level Queue

- 本算法引入多个就绪队列，通过各队列的区别对待，达到一个综合的调度目标；
- 基本思想：
  - 根据作业或进程的性质或类型的不同，将就绪队列再分为若干个子队列。
  - 每个作业固定归入一个队列。
  - 各队列不同处理：不同队列可有不同的优先级、时间片长度、调度策略等。如：系统进程、用户交互进程、批处理进程等。

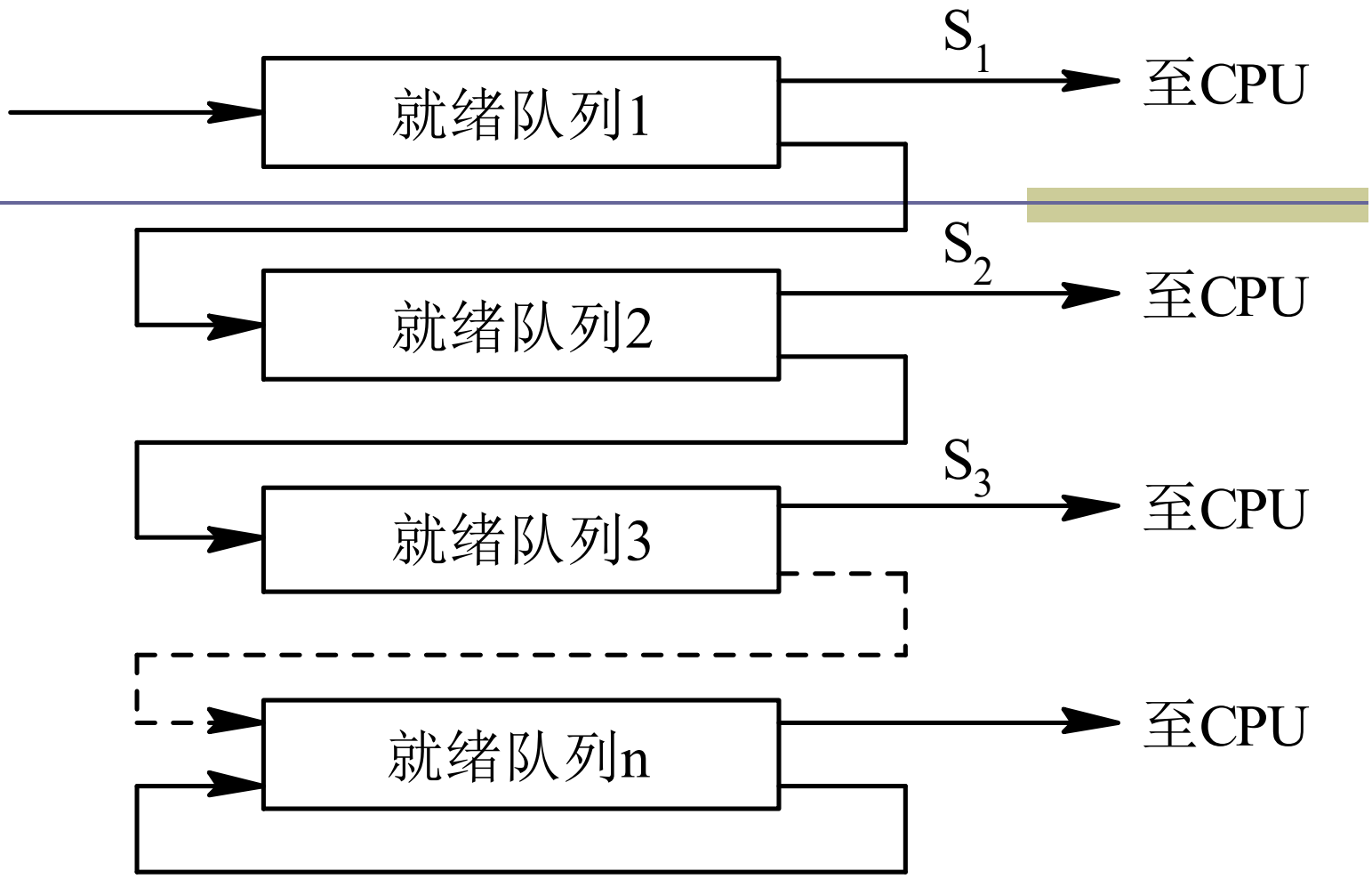

highest priority
system processes
interactive processes
interactive editing processes
batch processes
student processes
lowest priority

多级反馈队列算法
Multiple-level Queue

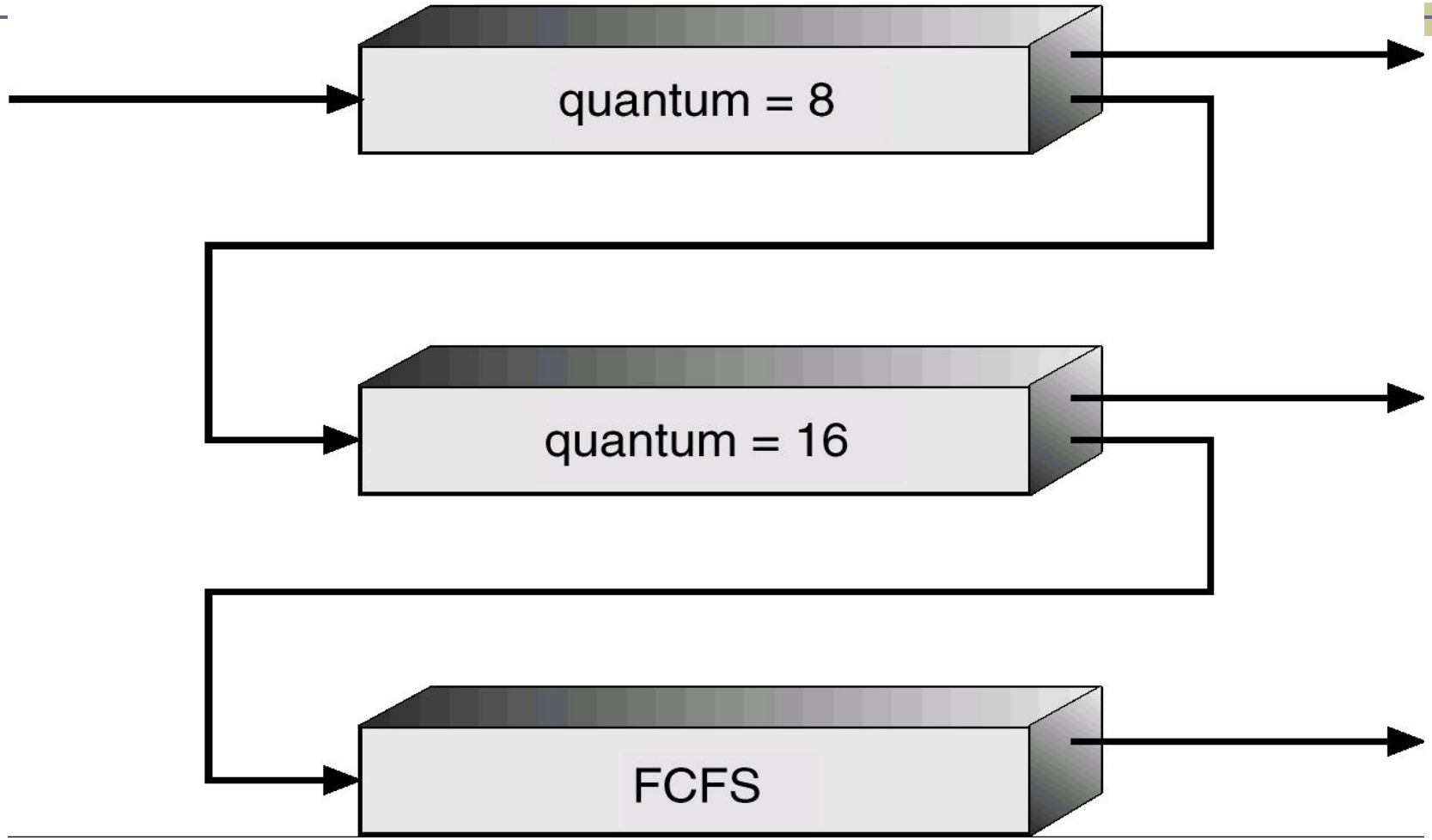
- 多级反馈队列算法是时间片轮转算法和优先级算法的综合和发展。优点：
  - 为提高系统吞吐量和缩短平均周转时间而照顾短进程
  - 为获得较好的I/O设备利用率和缩短响应时间而照顾I/O型进程
  - 不必估计进程的执行时间，动态调节
- 优先级分组法
  - 保留非剥夺式优先级和剥夺式优先级各自的优点，克服其缺点。
  - 方法：组间可剥夺，组内不可剥夺（组内相同优先级则按FCFS处理）

基本思想

- 设置多个就绪队列，分别赋予不同的优先级，如逐级降低，队列1的优先级最高。每个队列执行时间片的长度也不同，规定优先级越低则时间片越长，如逐级加倍
- 新进程进入内存后，先投入队列1的末尾，按FCFS算法调度；若按队列1一个时间片未能执行完，则降低投入到队列2的末尾，同样按FCFS算法调度；如此下去，降低到最后的队列，则按"时间片轮转"算法调度直到完成。
- 仅当较高优先级的队列为空，才调度较低优先级的队列中的进程执行。如果进程执行时有新进程进入较高优先级的队列，则抢先执行新进程，并把被抢先的进程投入原队列的末尾。



（时间片：$S_1 < S_2 < S_3$）



几点说明

- 时间片的变化，进入更低级队列。最终采用最大时间片来执行，减少调度次数。
- I/O次数不多，而主要是CPU处理的进程： I/O型进程：让其进入最高优先级队列，以及时响应I/O交互。通常执行一个小时间片，要求可处理完一次I/O请求的数据，然后转入到阻塞队列。
- 计算型进程：每次都执行完在I/O完成后，放回优先I/O请求时离开的队列，以免每次都回到最高优先级队列后再逐次下降。
- 为适应一个进程在不同时间段的运行特点，I/O完成时，提高优先级；时间片用完时，降低优先级；

- 特点:
  - 短作业优先。
  - 输入/输出进程优先。
  - 运算型进程有较长的时间片。
  - 采用了动态优先级，使用珍贵资源ＣＰＵ的进程优先级不断降低。 采用了可变时间片以适应不同进程对时间的要求，运算型进程将获得较长的时间片。

不同的环境需要不同的调度算法

- 批处理系统
  - FCFS
  - Shortest job first（非剥夺）
  - Shortest remaining time next（SJF的剥夺版）
- 交互式系统
  - Round robin
  - Priority Scheduling
  - Round Robin with Multiple Feedback （CTSS）
  - Lottery scheduling
- 实时系统
  - 硬实时：必须满足绝对的截止时间
  - 软实时：不希望错失截止时间，但可以容忍

What you need to do?

- 复习课本3.4节的内容

See you next time!