西安交通大学
软件学院

操作系统原理

Operating System Principle

田丽华

5-1　CPU调度

Basic Concepts
基本概念

Maximum CPU utilization obtained with multiprogramming

（通过多道程序设计得到CPU的最高利用率）

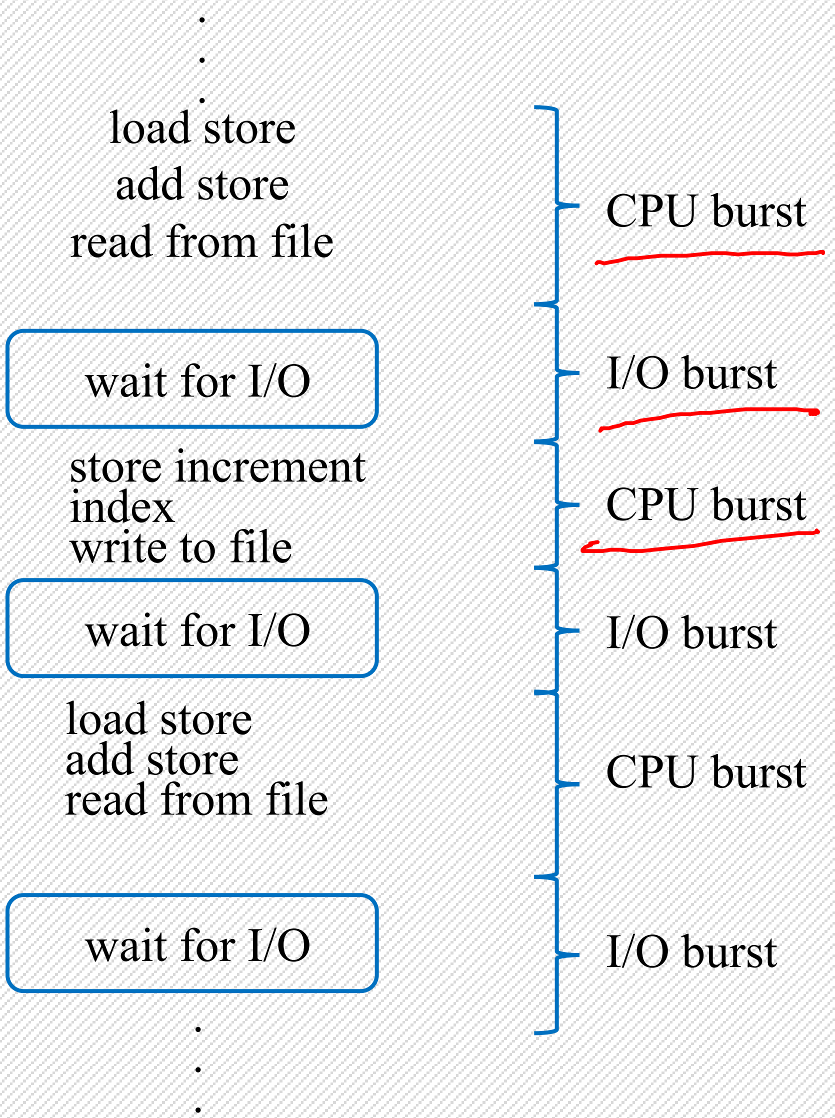
CPU–I/O Burst Cycle – Process execution consists of a cycle of CPU execution and I/O wait.

（CPU-I/O脉冲周期 - 进程的执行包括进程在CPU上执行和等待I/O）

进程的执行以CPU脉冲开始,其后跟着I/O脉冲.进程的执行就是在这两个状态之间进行转换.

Alternating Sequence of CPU And I/O Bursts

cpu和I/O burst的交替序列



CPU burst distribution

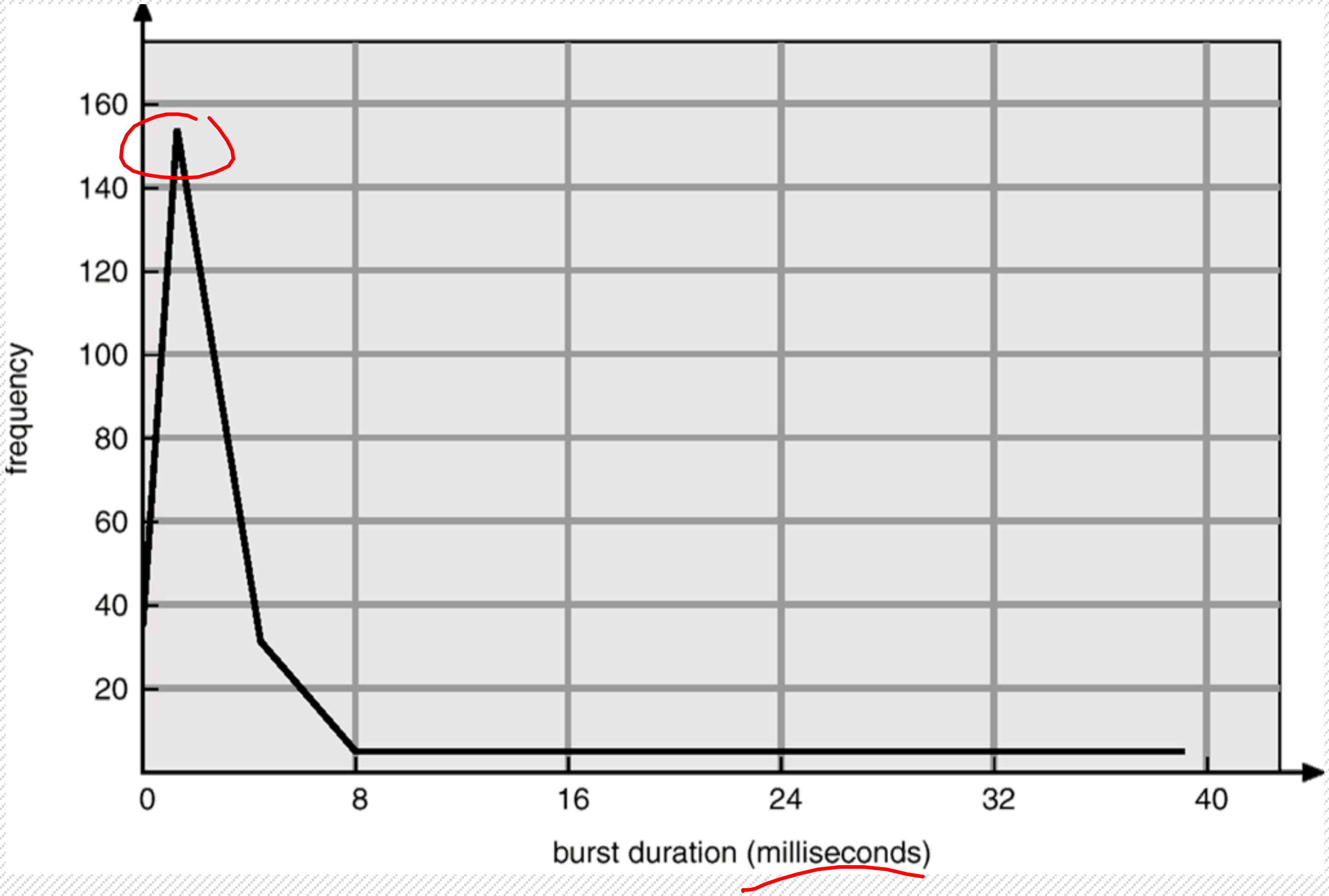
cpu-burst

CPU脉冲的分布,在系统中,存在许多短CPU脉冲,只有少量的长CPU脉冲

比如:I/O型作业具有许多短CPU脉冲,而CPU型作业则会有几个长CPU脉冲,这个分布规律对CPU调度算法的选择是非常重要的.

Histogram of CPU-burst Times

Cpu-burst次数的直方图

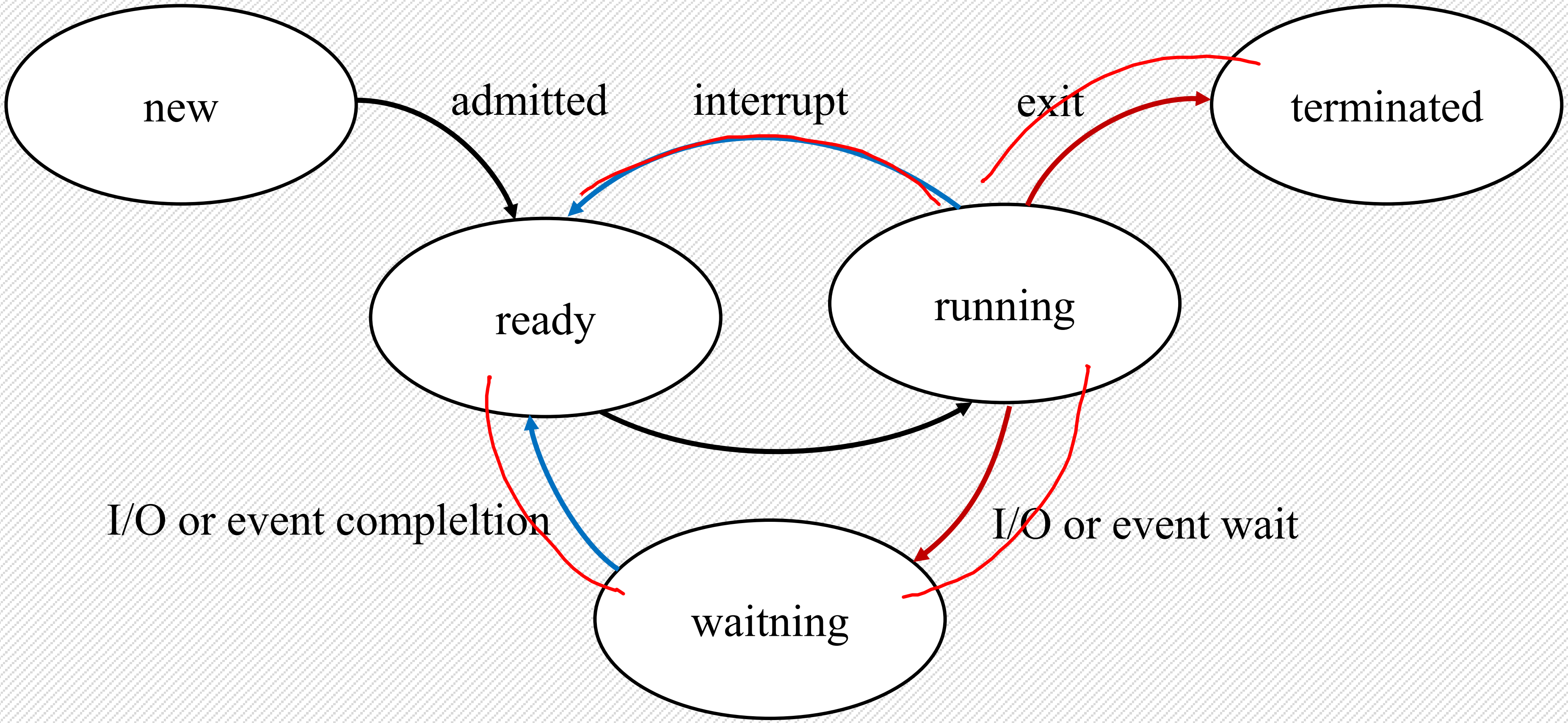


CPU Scheduler
cpu调度

- Selects from among the processes in memory that are ready to execute, and allocates the CPU to one of them.
- 当CPU空闲时，OS就选择内存中的某个就绪进程，并给其分配CPU

CPU Scheduler

cpu调度器



CPU Scheduler

cpu调度器

CPU scheduling decisions may take place under the following circumstances:

（CPU调度可能发生在以下情况下）：

1.Switches from running to waiting state（从运行转到等待）.

2.Switches from running to ready state（从运行转到就绪）.

3.Switches from waiting to ready（从等待转到就绪）.

4.Terminates（终止运行）.

Scheduling under 1 and 4 is nonpreemptive （发生在1、4两种情况下的调度称为非抢占式调度）.

All other scheduling is preemptive （其他情况下发生的调度称为抢占式调度）.

CPU Scheduling Scheme

cpu调度方案

非抢占方式(nonpreemptive)

- 把处理机分配给某进程后，便让其一直执行，直到该进程完成或发生某事件而被阻塞时，才把处理机分配给其它进程，不允许其他进程抢占已经分配出去的处理机。
- 优点:实现简单、系统开销小，适用于大多数批处理系统环境
- 缺点:难以满足紧急任务的要求，不适用于实时、分时系统要求

抢占方式（Preemptive mode）

- 允许调度程序根据某个原则，去停止某个正在执行的进程，将处理机重新分配给另一个进程。

CPU Scheduling Scheme

cpu调度方案

抢占的原则

- **时间片原则**:各进程按时间片运行，当一个时间片用完后，便停止该进程的执行而重新进行调度。这个原则适用于分时系统。
- **优先权原则**:通常对一些重要的和紧急的进程赋予较高的优先权。当这种进程进入就绪队列时，如果其优先权比正在执行的进程优先权高，便停止正在执行的进程，将处理机分配给优先权高的进程，使之执行
- **短作业优先原则**:当新到达的作业比正在执行的作业明显短时，将暂停当前长作业的执行，将处理机分配给新到的短作业，使之执行。

西安交通大学
软件学院

操作系统原理

Operating System Principle

田丽华

5-2 FCFS

CPU调度

01 调度程序采用什么算法选择一个进程（作业）？

02 如何评价调度算法的性能?

Scheduling Criteria

调度准则

调度准则

CPU utilization – keep the CPU as busy as possible（CPU利用率 – 使CPU尽可能的忙碌）

Throughput – the number of processes that complete their execution per time unit（吞吐量 – 单位时间内运行完的进程数）

Turnaround time – the interval from submission to completion（周转时间 – 进程从提交到运行结束的全部时间）

Waiting time – amount of time a process has been waiting in the ready queue（等待时间 – 进程在就绪队列中等待调度的时间片总和）

Scheduling Criteria

调度准则

Response time – amount of time it takes from when a request was submitted until the first response is produced, not output (for time-sharing environment)

（响应时间 – 从进程提出请求到首次被响应的时间段[在分时系统环境下不是输出完结果的时间]）

调度算法影响的是等待时间，而不能影响进程真正使用CPU的时间和I/O时间

Scheduling Criteria
调度准则

Optimization Criteria

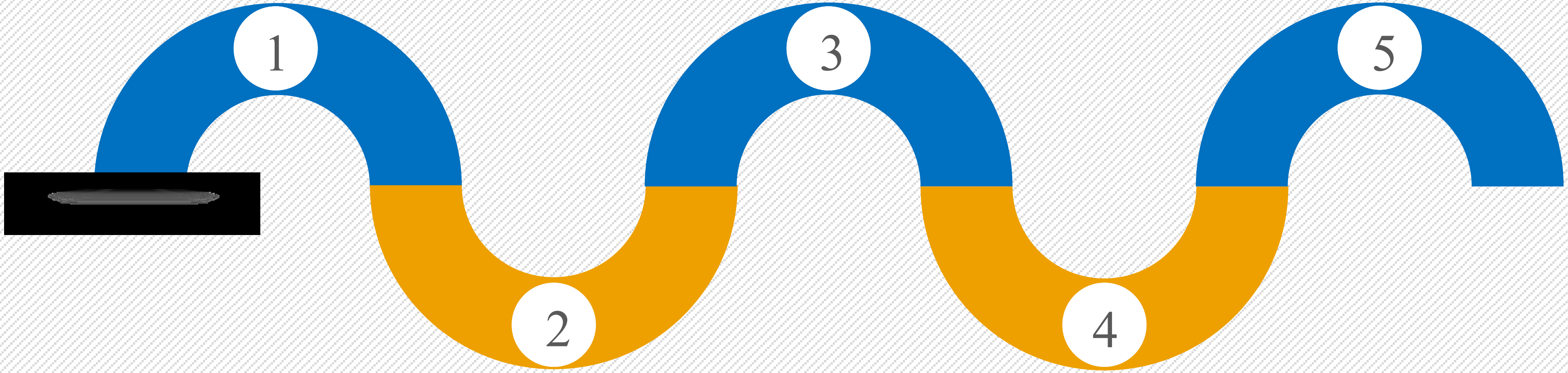
Max CPU utilization
(最大的CPU利用率)

Max throughput
(最大的吞吐量)

Min turnaround time
(最短的周转时间)

Min waiting time
(最短的等待时间)

Min response time
(最短的响应时间)



Scheduling Algorithm

调度算法

- 先来先服务(FCFS)
- 短作业优先(SJF)
- 优先权调度(Priority Scheduling)
- 时间片轮转(Round Robin)
- 多级队列调度(Multilevel Queue)
- 多级反馈队列调度算法(Multilevel Feedback Queue)

First-Come, First-Served (FCFS) Scheduling

- 先来先服务First-Come-First-Served:

01 最简单的调度算法

02 可用于作业或进程调度

03 算法的原则是按照作业到达后备作业队列（或进程进入就绪队列）的先后次序来选择作业（或进程）

First-Come, First-Served (FCFS) Scheduling

- FCFS算法属于非抢占方式:一旦一个进程占有处理机，它就一直运行下去，直到该进程完成或者因等待某事件而不能继续运行时才释放处理机。
- FCFS算法易于实现，表面上很公平，实际上有利于长作业，不利于短作业；有利于CPU繁忙型，不利于I/O繁忙型。

Scheduling Algorithm
调度算法

First-Come, First-Served (FCFS) Scheduling

Example：

Process	Burst Time
P_1	24
P_2	3
P_3	3

Suppose that the processes arrive in the order（假定进程到达顺序如下）: P_1 , P_2 , P_3

The Gantt Chart for the schedule is（该调度的Gantt图为）:

First-Come, First-Served (FCFS) Scheduling

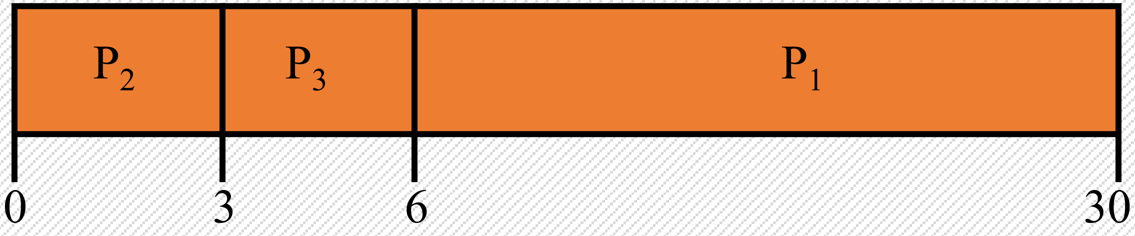
Waiting time（等待时间） for $P_1 = 0$; $P_2 = 24$; $P_3 = 27$

Average waiting time（平均等待时间）：$(0 + 24 + 27)/3 = 17$

FCFS Scheduling (Cont.)

Suppose that the processes arrive in the order（假定进程到达顺序如下） P_2 , P_3 , P_1 .

The Gantt Chart for the schedule is（该调度的Gantt图为）：



Waiting time（等待时间） for $P_1 = 6$; $P_2 = 0$; $P_3 = 3$

FCFS Scheduling (Cont.)

Suppose that the processes arrive in the order （假定进程到达顺序如下） P_2 , P_3 , P_1 .

Average waiting time （平均等待时间）： $(6 + 0 + 3)/3 = 3$

Much better than previous case（比前例好得多）.

short process behind long process （此种结果产生是由于长进程先于短进程到达）

Convoy effect 护航效应

假设有一个CPU进程和许多I/O型进程

当CPU进程占用CPU运行时， I/O型进程可能完成了其I/O操作，回到就绪队列等待CPU， I/O设备空闲

CPU进程释放CPU后， I/O型进程陆续使用CPU，并很快转为I/O操作， CPU空闲

西安交通大学
软件学院

操作系统原理

Operating System Principle

田丽华

5-3 SJF (Shortest-Job-First)

Shortest-Job-First (SJF) Scheduling

Associate with each process the length of its next CPU burst. Use these lengths to schedule the process with the shortest time.
（关联到每个进程下次运行的CPU脉冲长度，调度最短的进程）

Shortest-Job-First (SJF) Scheduling

Two schemes:

nonpreemptive – once CPU given to the process it cannot be preempted until completes its CPU burst（非抢占式调度 – 一旦进程拥有CPU，它的使用权限只能在该CPU 脉冲结束后让出).

Preemptive – if a new process arrives with CPU burst length less than remaining time of current 法executing process, preempt. This scheme is know as the Shortest-Remaining-Time-First (SRTF).（抢占式调度 – 发生在有比当前进程剩余时间片更短的进程到达时，也称为最短剩余时间优先调度）

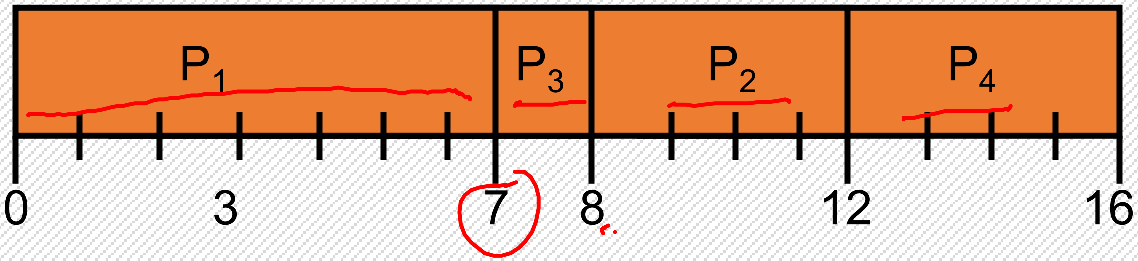
Shortest-Job-First (SJF) Scheduling

SJF is optimal – gives minimum average waiting time for a given set of processes.（SJF是最优的 – 对一组指定的进程而言，它给出了最短的平均等待时间）

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

01 SJF (non-preemptive)

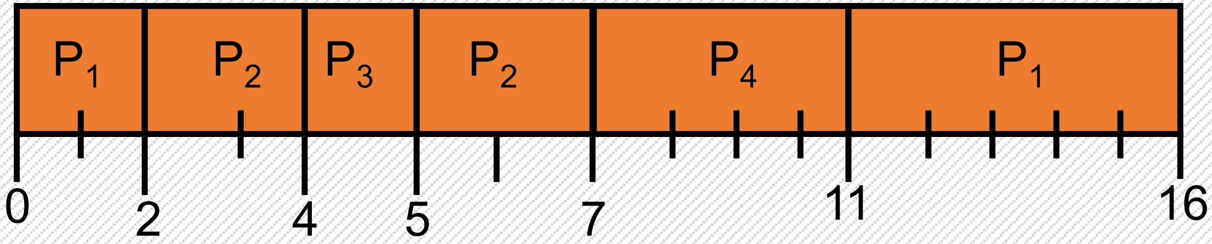


02 Average waiting time = (0 + 6 + 3 + 7)/4 =4

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

01 SJF (preemptive)



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

Determining Length of Next CPU Burst

- Although the SJF algorithm is optimal, it cannot be implemented at the level of short_term CPU scheduling.
- Can only estimate the length（其长度只能估计）.
- Can be done by using the length of previous CPU bursts, using exponential averaging （可以通过先前的CPU脉冲长度及计算指数均值进行）.

1. t_n = actual lenght of n^{th} CPU burst
2. τ_{n+1} = predicted value for the next CPU burst
3. $\alpha, 0 \le \alpha \le 1$
4. Define: $\tau_{n+1} = \alpha t_n + (1-\alpha)\tau_n$

Scheduling Algorithm

调度算法

Shortest-Job-First (SJF) Scheduling

01 采用SJF有利于系统减少平均周转时间,提高系统吞吐量。

02 一般情况下SJF调度算法比FCFS调度算法的效率要高一些, 但实现相对要困难些。

03 如果作业的到来顺序及运行时间不合适，会出现饥饿现象，例如，系统中有一个运行时间很长的作业JN，和几个运行时间小的作业，然后，不断地有运行时间小于JN的作业的到来，这样，作业JN就因得不到调度而饿死。另外，作业运行的估计时间也有问题。

西安交通大学
软件学院

操作系统原理

Operating System Principle

田丽华

5-4 优先级和 RR时间片轮转

Priority Scheduling

- A priority number (integer) is associated with each process（每个进程都有自己的优先数[整数]）
- The CPU is allocated to the process with the highest priority (smallest integer ≡ highest priority)（CPU分配给最高优先级的进程[假定最小的整数 ≡ 最高的优先级]）.

1 Preemptive（抢占式）

2 Nonpreemptive （非抢占式）

- SJF is a priority scheduling where priority is the predicted next CPU burst time（SJF是以下一次CPU脉冲长度为优先数的优先级调度）.

Priority Scheduling

进程	运行时间	优先权
P1	10	3
P2	1	1
P3	2	4
P4	1	5
P5	5	2

- 调度顺序：P2、P5、P1、P3、P4
- 平均等待时间8.2ms

Priority Scheduling

1. 静态优先权在进程创建时确定，且在整个生命期中保持不变。

2. 静态优先权的问题 Problem ≡ Starvation – low priority processes may never execute

（问题 ≡ 饥饿 – 低优先级的可能永远得不到运行）.

一个很有意思的例子：当MIT的IBM7094机器于1973年关掉时，人们发现一个于1967年提交的一个低优先权的进程还没有得到运行。

Solution ≡ Aging – as time progresses increase the priority of the process

（解决方法 ≡ 老化 – 视进程等待时间的延长提高其优先数）.

Priority Scheduling

- 动态优先权是指进程的优先权可以随进程的推进而改变，以便获得更好的调度性能
- 改变优先权的因素

进程的等待时间

已使用处理机的时间

资源使用情况

Round Robin (RR)

Each process gets a small unit of CPU time *(time quantum),* usually 10-100 milliseconds. After this time has elapsed, the process is preempted and added to the end of the ready queue

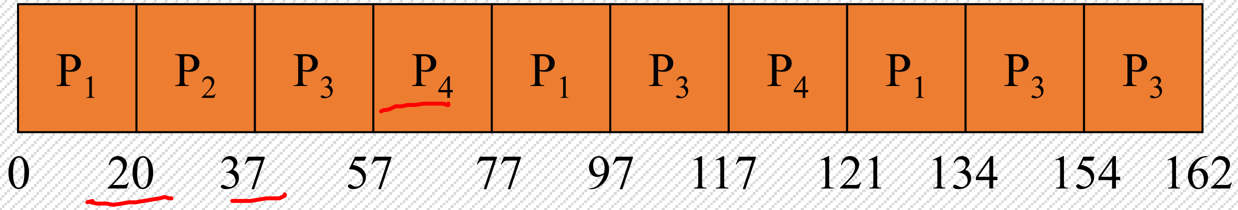
(每个进程将得到小单位的CPU时间[时间片]，通常为10-100毫 秒。时间片用完后，该进程将被抢占并插入就绪队列末尾)

Round Robin (RR)

Example: RR with Time Quantum = 20

Process	Burst Time
P_1	53
P_2	17
P_3	68
P_4	24

01 The Gantt chart is:



02 Typically, higher average turnaround than SJF, but better response（一般来说，RR的平均周转时间比SJF长，但响应时间要短一些）.

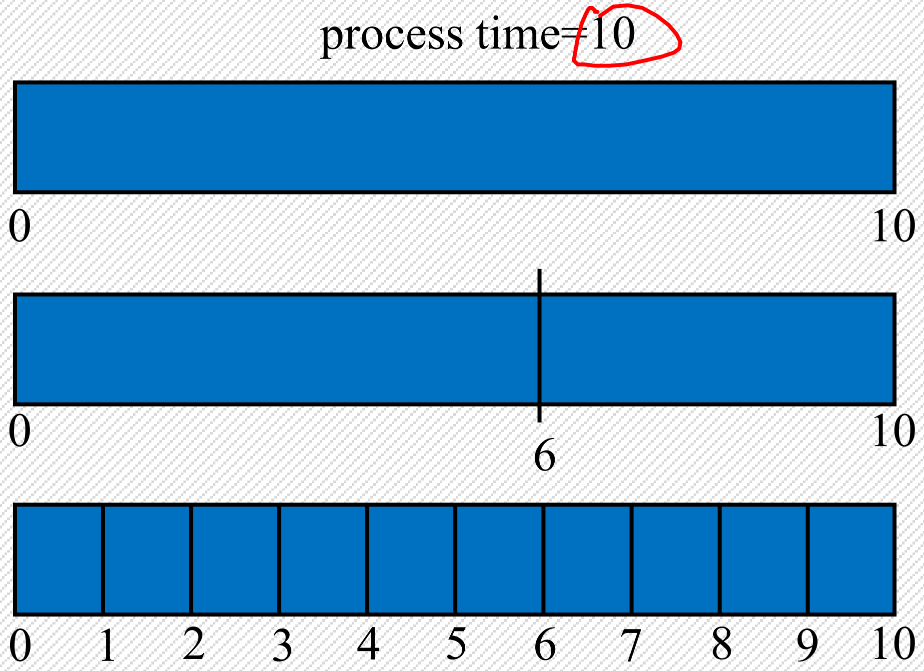
Round Robin (RR)

Performance（特性）

1. q large $\Rightarrow$ FCFS
2. q small $\Rightarrow$ q must be large with respect to context switch, otherwise overhead is too high（q相对于切换上下文的时间而言足够长，否则将导致系统开销过大）.

Round Robin (RR)

How a Smaller Time Quantum Increases Context Switches

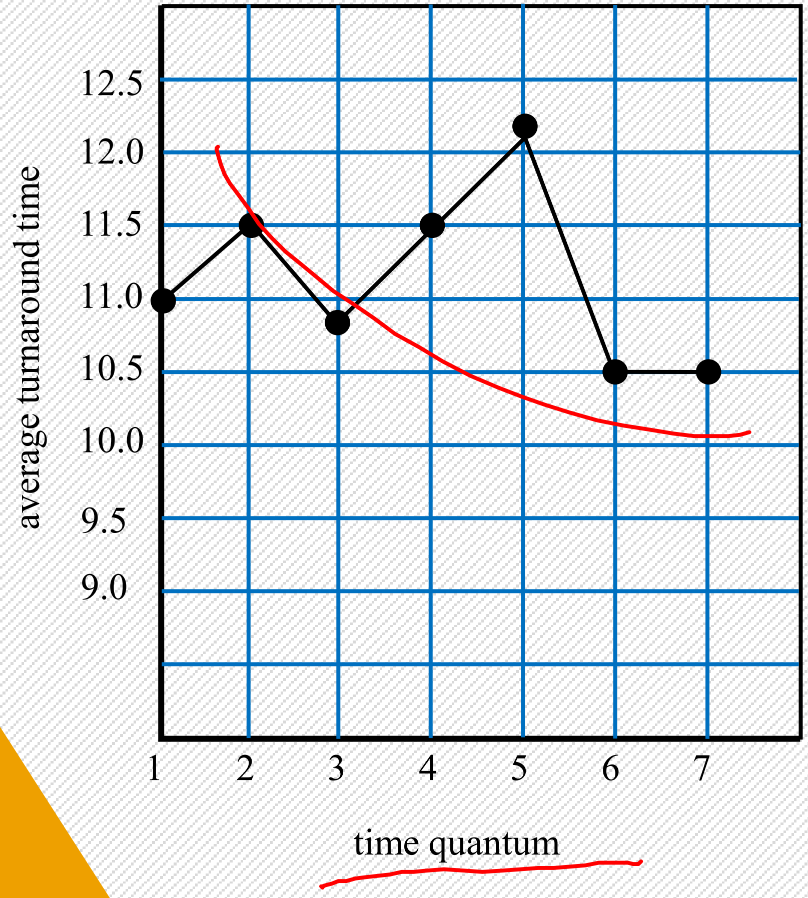


quantum	context switches
12	0
6	1
1	9

Longer quantum yields shorter average turnaround times?

Round Robin (RR)

Turnaround Time Varies With The Time Quantum



process	time
P_1	6
P_2	3
P_3	1
P_4	7

一组进程的平均周转时间并不一定随着时间片的增大而降低。一般来说，如果大多数（80%）进程能在一个时间片内完成，就会改善平均周转时间

西安交通大学
软件学院

操作系统原理

Operating System Principle

田丽华

5-5　多级队列
多级反馈队列

Multilevel Queue
多级队列

按进程的属性来分类，如进程的类型、优先权、占用内存的多少,每类进程组成一个就绪队列，每个进程固定地处于某一个队列，如

- Ready queue is partitioned into separate queues（就绪队列分为）:
  foreground (interactive)（前台）[交互式]
  background (batch) （后台） [批处理]
- Each queue has its own scheduling algorithm（每个队列有自己的调度算法）
  foreground – RR
  background – FCFS
- Scheduling must be done between the queues（调度须在队列间进行）.

Multilevel Queue

多级队列

1

Fixed priority scheduling; i.e., serve all from foreground then from background. Possibility of starvation

（固定优先级调度，即前台运行完后再运行后台。有可能产生饥饿）

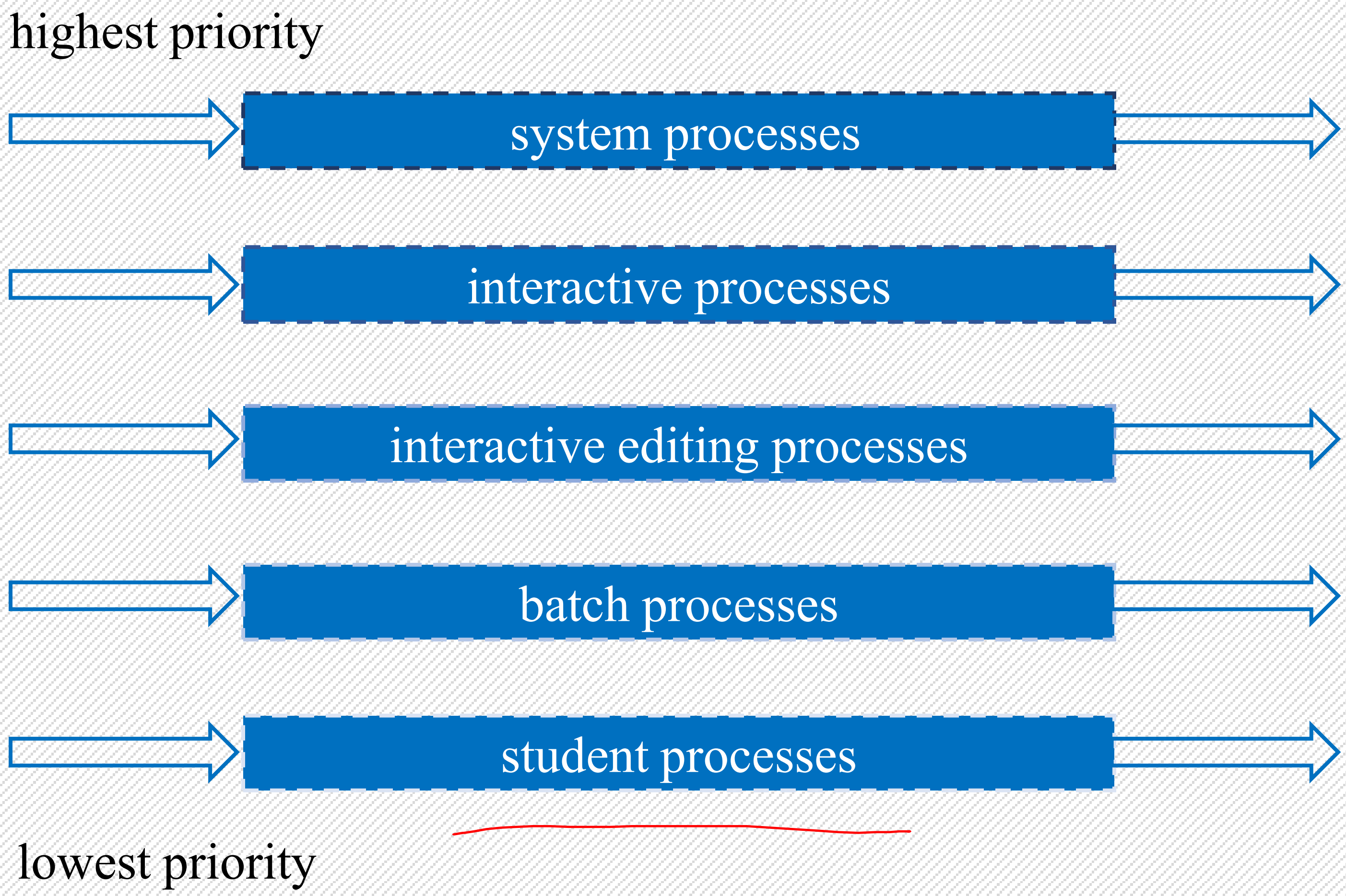
Multilevel Queue
多级队列

2

Time slice – each queue gets a certain amount of CPU time which it can schedule amongst its processes; e.g.,80% to foreground in RR 20% to background in FCFS

给定时间片调度，即每个队列得到一定的CPU时间，进程在给定时间内执行；如，80%的时间执行前台的RR调度，20%的时间执行后台的FCFS调度

Multilevel Queue Scheduling
多级队列调度



Multilevel Feedback Queue

多级反馈队列调度

存在多个就绪队列，具有不同的优先级，各自按时间片轮转法调度

各个就绪队列中时间片的大小各不相同，优先级越高的队列时间片越小。

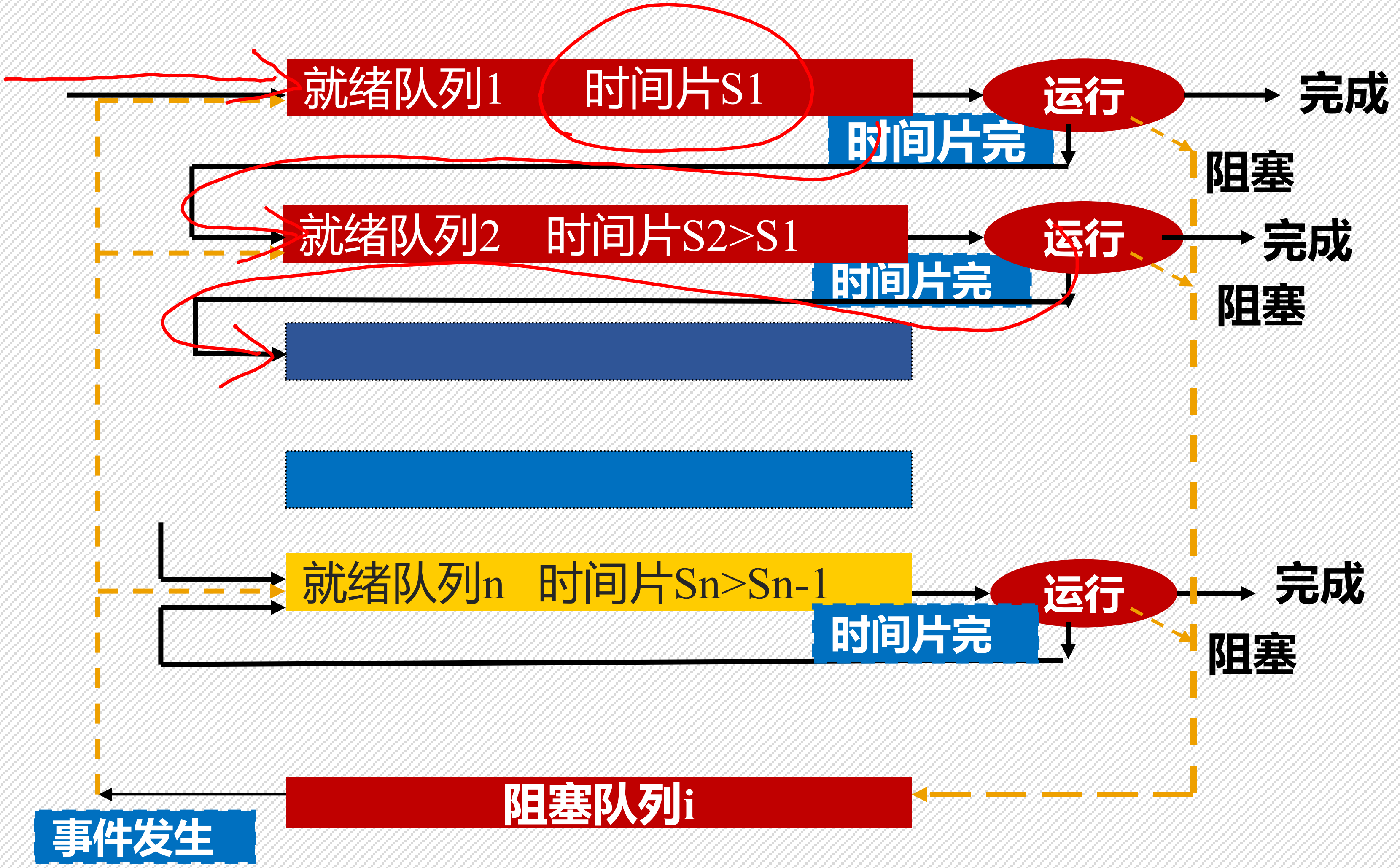
允许进程在队列之间移动

当一个进程执行完一个完整的时间片后被抢占处理器，被抢占的进程优先级降低一级而进入下级就绪队列，如此继续，直至降到进程的基本优先级。而一个进程从阻塞态变为就绪态时要提高优先级

最后会将I/O型和交互式进程留在较高优先级队列

Multi-level feedback queue

图：多级反馈队列



Multi-level Feedback Queue

多级反馈队列调度

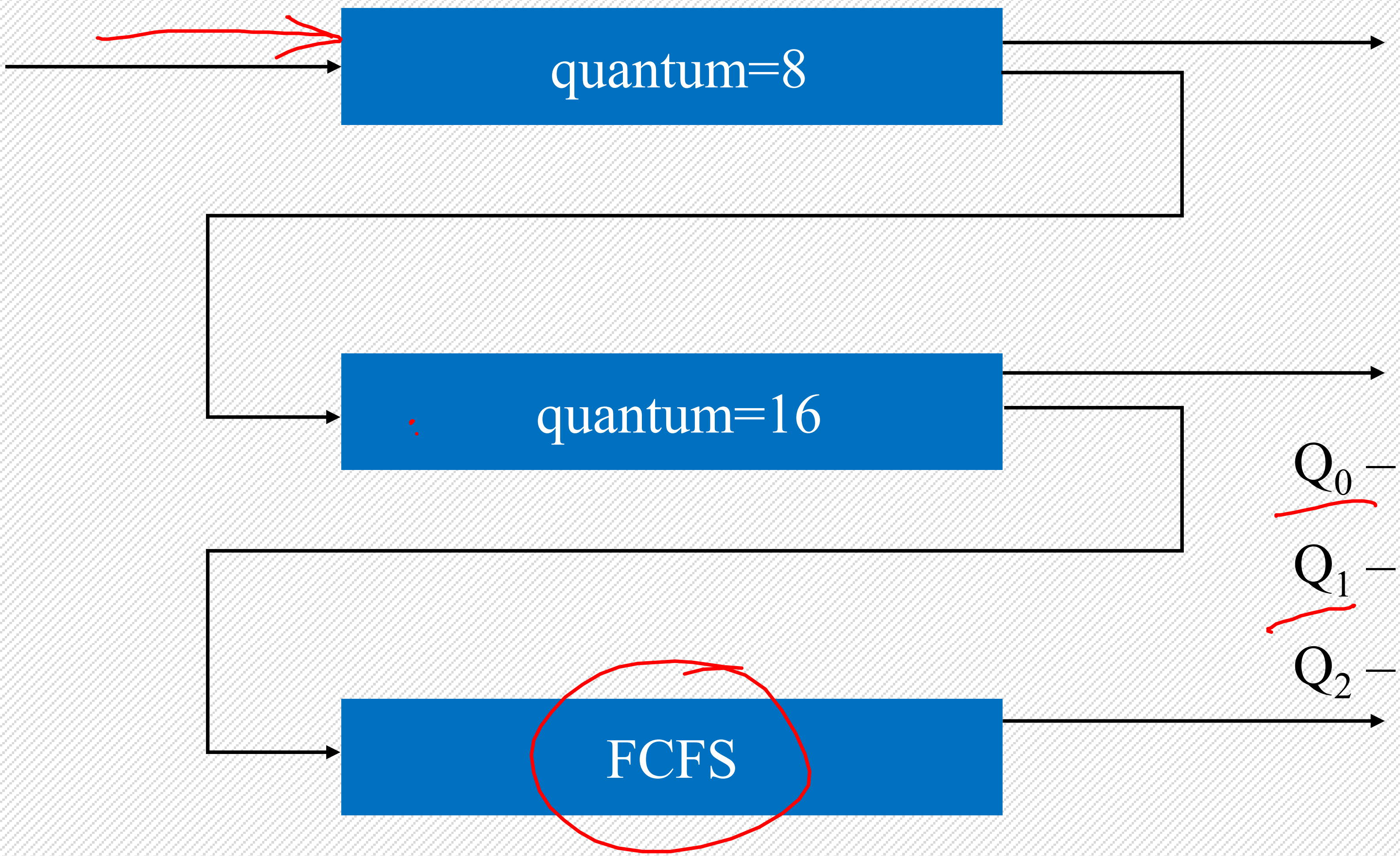
A process can move between the various queues; aging can be implemented this way（进程能在不同的队列间移动；可实现老化）.

Multilevel-feedback-queue scheduler defined by the following parameters（多级反馈队列调度程序由以下参数定义）：

- number of queues **（队列数）**
- scheduling algorithms for each queue **（每一队列的调度算法）**
- method used to determine when to upgrade a process **（决定进程升级的方法）**
- method used to determine when to demote a process **（决定进程降级的方法）**
- method used to determine which queue a process will enter when that process needs service **（决定需要服务的进程将进入哪个队列的方法）**

Multilevel feedback Queues

多级反馈队列调度



Q_0 – time quantum 8 milliseconds

Q_1 – time quantum 16 milliseconds

Q_2 – FCFS