華中農業大學
HUAZHONG AGRICULTURAL UNIVERSITY

计算机操作系统

倪福川
fcni_cn@mail.hzau.edu.cn

华中农业大学 信息学院

目 录

第三章　处理机调度与死锁

3.1 处理机调度的层次和调度算法的目标

处理机调度：对处理机资源进行分配。

处理机调度算法：根据处理机分配策略所规定的处理机分配算法。

多道批处理系统，一个作业从提交到获得处理机执行，直至作业运行完毕，经历三级级处理机调度。

3.1.1 处理机调度的层次

1. 高级调度(High Level Scheduling)

2. 低级调度(Low Level Scheduling)

3. 中级调度(Intermediate Scheduling)

1. 高级调度

- **作业调度**、长程调度 、接纳调度
- **作用**：把外存上处于后备队列中的作业调入内存，为它们创建进程、分配资源、排在就绪队列上，准备执行。

分时系统、实时系统通常不需要作业调度（批处理）

2. 低级调度

- **进程调度、短程调度 。**
- **作用：**决定就绪队列中的哪个进程应获得处理机，然后由分派程序执行把处理机分配给该进程。

在OS中都必须配置。 内存到内存

3. 中级调度

挂起

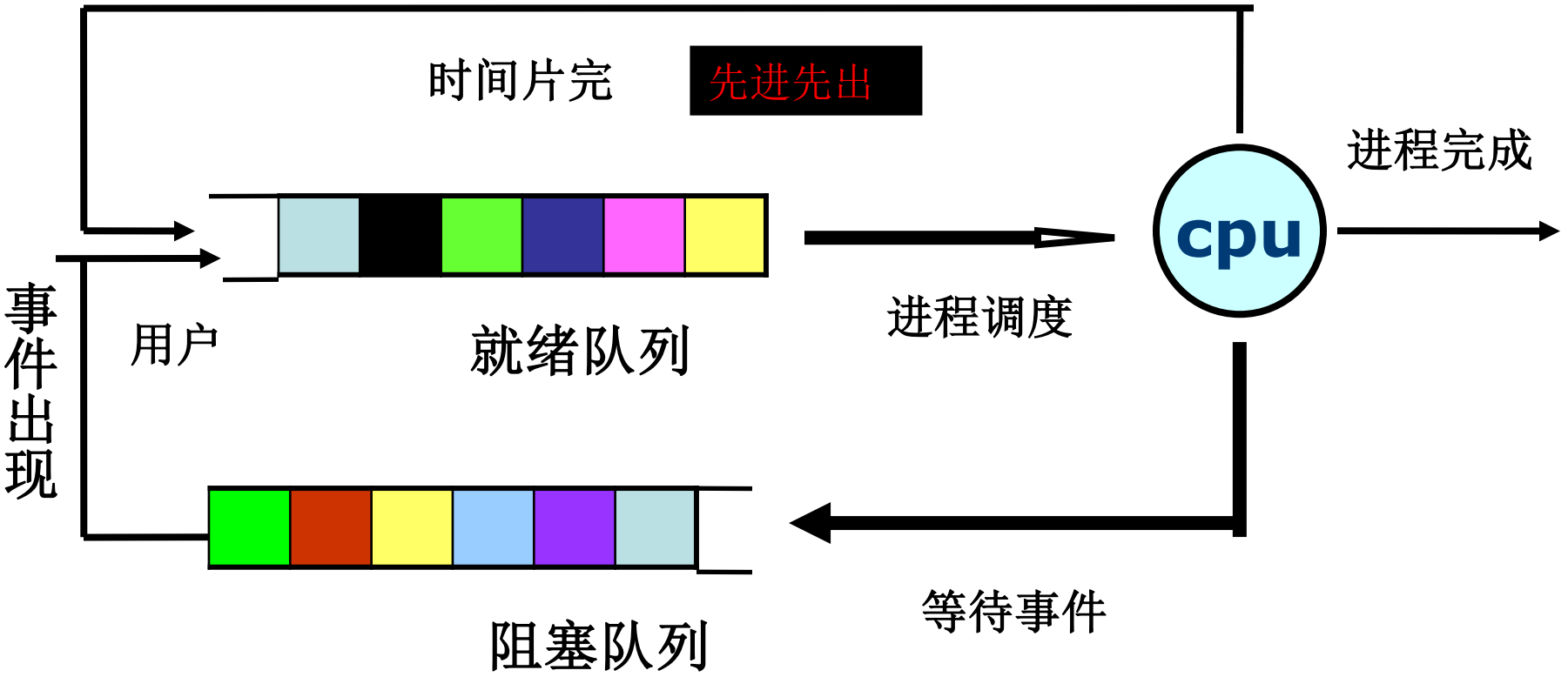
- **对换**、中程调度
- **作用**：使暂时不能运行的进程从内存调至外存，进入就绪驻外存状态或挂起状态。把外存上又具备运行条件的就绪进程，重新调入内存，并修改为就绪状态，挂在就绪队列上。

目的 ：提高内存利用率和系统吞吐率

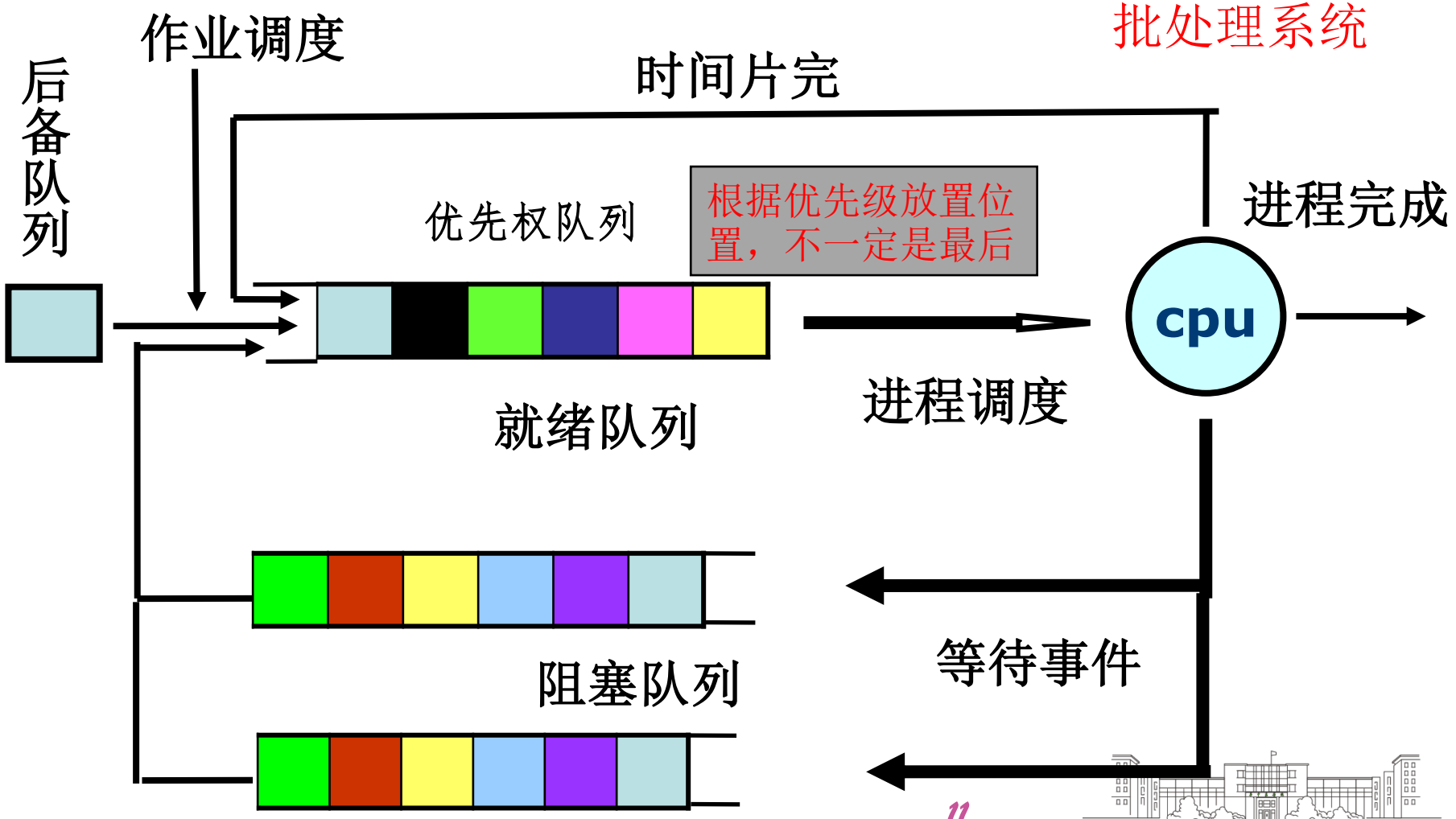
调度队列模型

- 仅有进程调度的调度队列模型
- 有高级和低级调度的调度队列模型
- 同时有三级调度的调度队列模型

仅有进程调度的调度队列模型



有高级和低级调度的调度队列模型



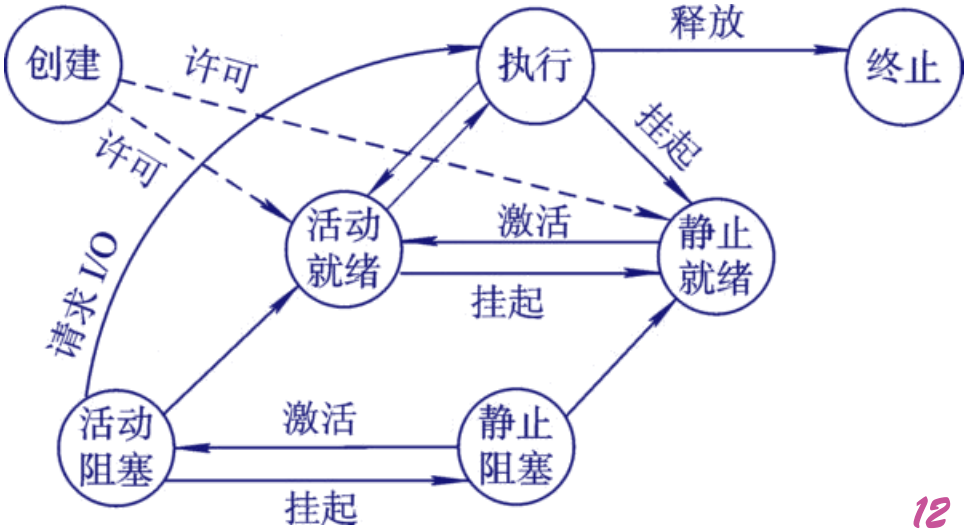
有三级调度的调度队列模型

调出时，可使进程状态由活动就绪转变为静止就绪，由活动阻塞转变为静止阻塞；

挂起

在中级调度使静止就绪转变为活动就绪。

激活



选择调度方式和调度算法的准则

面向用户的准则
- 周转时间短
- 响应时间快
- 截止时间的保证
- 优先权准则

面向系统的准则
- 系统吞吐量高
- 处理机利用率好
- 资源的平衡利用

3.1.2 处理机调度算法的目标

1. 处理机调度算法的共同目标

(1) 资源利用率

$$\text{CPU 的利用率} = \frac{\text{CPU有效工作时间}}{\text{CPU有效工作时间} + \text{CPU空闲等待时间}}$$

(2) 公平性 进程都获得合理CPU ，不会饥饿。

相对性：同类进程获得相同服务，不同类进程提供不同服务。(紧急程度或重要性不同)

(3) 平衡性

保持资源平衡性使用。(都处于忙碌)

多种类型进程，如计算型，I/O型。

(4) 策略强制执行

制订的策略须准确执行

作业周转时间

从作业被提交给系统开始，到作业完成为止的时间间隔。

包括四部分：

- 在外存后备队列上等待调度的时间
- 进程在就绪队列上等待调度的时间
- 进程在CPU上执行的时间
- 进程等待I/O操作完成的时间

2. 批处理系统的目标

(1) 平均周转时间短

用户：作业周转时间最短。

计算机系统管理者：平均周转时间最短

有效提高系统资源利用率，使多数用户满意。

平均周转时间：

$$T = \frac{1}{n}\left[\sum_{i=1}^{n} T_i\right]$$

应使作业周转时间和作业的平均周转时间尽可能短。否则，会使许多用户等待时间过长，引起用户特别是短作业用户的不满。

带权周转时间:作业的周转时间T与系统为它提供服务的时间T_s之比，

$$W = T/T_s。$$

描述各进程在其周转时间中，等待和执行时间的具体分配状况

平均带权周转时间:

$$W = \frac{1}{n}\sum_{i=1}^{n}\frac{T_i}{T_s}$$

T_s **真正运行时间**

在CPU上执行的时间，比周转时间短或相等（理想情况下）

越小越好

I/O繁忙型作业

CPU繁忙型作业

(2) 系统吞吐量高

吞吐量:在单位时间内系统所完成的作业数。

与批处理作业的平均长度有关。

如果只为获得高的系统吞吐量，应尽量多选择短作业运行。

评价批处理系统性能的重要指标

(3) 处理机利用率高

大、中型计算机的重要指标，

调度方式和算法　重要作用。

如单纯使处理机利用率高，应尽量多选择计算量大的作业。

这些要求存在着一定矛盾

3. 分时系统的目标

(1) 响应时间

从用户通过键盘提交一个请求开始直至系统首次产生响应为止的时间间隔

- 从键盘输入的请求信息传送到处理机的时间
- 处理机对请求信息进行处理的时间
- 将响应信息回送到终端显示器的时间。

(2) 均衡性

系统响应时间快慢与用户所请求服务的复杂性相适应

4. 实时系统的目标

(1) 截止时间的保证

截止时间：某任务必须开始执行的最迟时间，或必须完成的最迟时间。

严格的实时系统，其调度方式和调度算法必须能保证这一点

(2) 可预测性

3.2 作业与作业调度

作业是用户提交给系统的一项相对独立的工作。

操作员把用户提交的作业通过相应的输入设备输入到磁盘存储器，并保存在一个后备作业队列中。再由作业调度程序将其从外存调入内存。

3. 2. 1　批处理系统中的作业

1. 作业和作业步

(1) 作业(Job)　程序 数据 和 运行控制说明

(2) 作业步(Job Step)　加工步骤：　编译/连接/运行

2. 作业控制块(Job Control Block，JCB)

作业在系统中存在的标志，保存对作业管理和调度所需信息。

JCB包含：

作业标识、用户名称、用户账号、

作业类型(CPU 繁忙型、I/O 繁忙型、批量型、终端型)、

作业状态、调度信息(优先级、作业运行时间)、

资源需求(预计运行时间、要求内存大小等)、

资源使用情况等。

3. 作业运行的三个阶段和三种状态

(1) 收容阶段 “后备状态”

输入磁盘　建立JCB　放入作业后备队列

(2) 运行阶段 “运行状态”

分配资源　建立进程　放入就绪队列

(3) 完成阶段 “完成状态”

回收JCB　资源　输出

3.2.2 作业调度的主要任务

根据JCB信息，检查系统资源能否满足需求，按照调度算法，从外存的后备队列中选取作业，调入内存，创建进程、分配资源，排在就绪队列上等待调度。

接纳调度(Admission Scheduling)

执行作业调度时，需做出两个决定：

1. 接纳多少个作业
2. 接纳哪些作业

3.2.3 先来先服务(FCFS)和短作业优先(SJF)调度

1. 先来先服务(FCFS)调度算法

按照作业到达先后次序来调度，优先考虑等待时间最长的作业，而不管作业所需执行时间长短。

从后备作业队列中选择几个最先进入该队列的作业，调入内存，配资源和创建进程，放入就绪队列。

按时间顺序服务，适用于高级和低级调度

最简单，作业调度/进程调度。

3.2.3 先来先服务(FCFS)和短作业优先(SJF)调度

1. 先来先服务(FCFS)调度算法

按作业到达先后次序来调度，优先考虑等待时间最长的作业，而不管所需执行时间长短。

有利长作业（进程）而不利于短作业（进程）；

有利CPU繁忙型作业，不利于I/O繁忙型作业。

没有或很少I/O操作，基本是运算，带权周转时间接近1

带权周转时间越大，I/O繁忙型作业；带权周转时间小，CPU繁忙型作业

2. 短作业优先的调度算法

（short job first，SJF）

为使短作业(进程)能比长作业优先执行

以作业的长短来计算优先级，作业越短，其优先级越高。作业的长短以作业所要求的运行时间来衡量。

调度算法：从就绪队列中选一估计运行时间最短的进程，分配处理机使它立即执行直到完成，或发生某事件而被阻塞放弃处理机时，再重新调度。

可用于作业调度和进程调度。

2) 短作业优先算法缺点

(1) 必须预知作业运行时间：偏长估计。

(2) 对长作业非常不利

长作业周转时间增长；可能出现饥饿

(3)完全未考虑作业紧迫程度，不能保证紧迫性作业（进程）被及时处理。

完全忽视作业的等待时间　长作业长期不被调度

3.2.4 优先级调度算法和高响应比优先调度算法

1. 优先级调度算法

(priority-scheduling algorithm, PSA)

作业等待时间，作业长短作为优先级都不能反映作业的紧迫程度。

基于作业的紧迫程度，赋予作业优先级，保证优先级高的作业先运行。

优先权的类型

静态优先权　　一般不会改变

在创建进程时确定，在进程的整个运行期间保持不变

整数来表示　　简单，系统开销小；不精确

动态优先权

随进程推进或随其等待时间的增加而改变的，以便获得更好的调度性能

确定优先权依据：

（1）进程类型：系统进程高于用户进程

（2）进程对资源的要求：要求少的进程应赋予较高优先权。

（3）用户要求。由用户进程的紧迫程度及所付费多少来确定。

2. 高响应比优先调度算法

(Highest Response Ratio Next，HRRN)

既考虑作业的等待时间，又考虑作业运行时间，既照顾短作业，又不致使长作业长时间等待，改善处理机调度性能。

动态优先级

$$\text{优先权} = \frac{\text{等待时间} + \text{要求服务时间}}{\text{要求服务时间}}$$

高响应比优先算法的实现

优先级相当于响应比R_P。

$$R_P = \frac{\text{等待时间} + \text{要求服务时间}}{\text{要求服务时间}} = \frac{\text{响应时间}}{\text{要求服务时间}}$$

作业的响应时间：等待时间与服务时间之和

动态优先级，随等待时间延长而增加，长作业优先级在等待期间不断地增加，等到足够的时间后，必然有机会获得处理机。

高响应比优先算法分析

- 作业等待时间相同，有利于短作业。
- 要求服务时间相同，先来先服务。
- 长作业也可获得处理机。

先来的等待时间就长

优点：兼顾长短作业。

缺点：计算响应比，增加系统开销

3.3 进程调度

OS必不可少的调度。

在三种类型的OS中，都配置进程调度。

对系统性能影响最大的一种处理机调度，调度的算法也较多。

3.3.1 进程调度的任务、机制和方式

1. 进程调度的任务

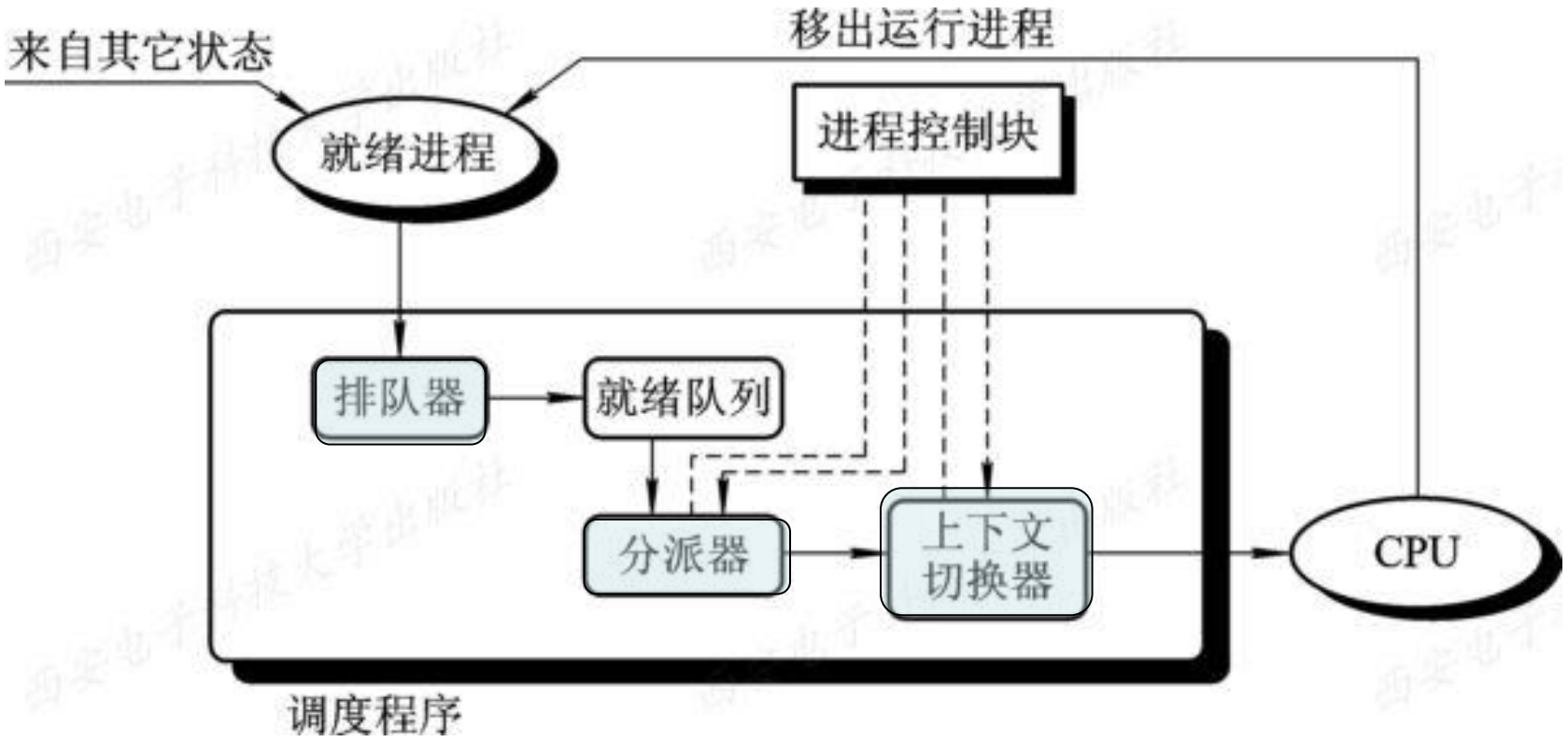
进程调度的任务：

(1) 保存处理机的现场信息。

(2) 按某种算法选取进程。

(3) 把处理器分配给进程。

2. 进程调度机制



(1) 排队器　(2) 分派器　(3) 上下文切换器　P91

3. 进程调度方式

1) 非抢占方式(Nonpreemptive Mode)

一旦分配处理机，一直运行，直至完成或阻塞

运行过程，不因时钟中断或任何其它原因，剥夺处理机

2) 抢占方式(Preemptive Mode)

调度程序根据某种原则，暂停某执行进程，将处理机重新分配给另一进程　　现代OS广泛采用

- 批处理机系统：防止长进程长时间占用处理机，确保公平
- 分时系统：只有抢占方式才可能实现人—机交互。
- 实时系统：满足实时任务需求，但复杂，开销大

抢占原则　①优先权　②短进程优先　③时间片

3.3.2 轮转调度算法

1. 轮转法的基本原理

将就绪进程按FCFS策略排成一个就绪队列。

每隔一定时间产生一次中断，去激活进程调度程序，把CPU分配给队首进程，并执行一个时间片。

保证所有就绪进程在确定的时间段内，都能获得一个时间片的处理机时间

2. 进程切换时机

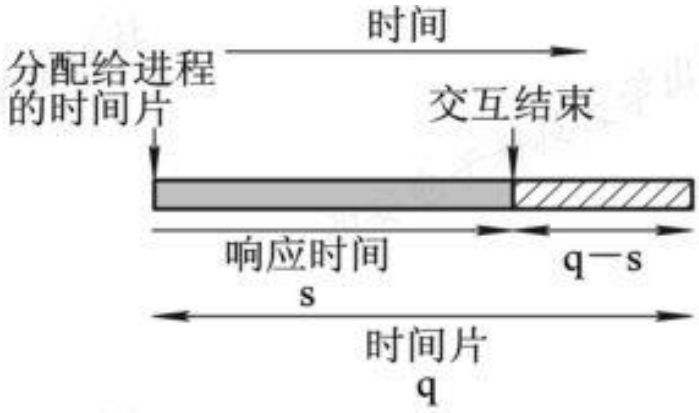
RR调度算法，进程切换情况：

① 时间片未用完，运行进程完成，激活调度程序，调度就绪队列中首进程，启动新时间片。

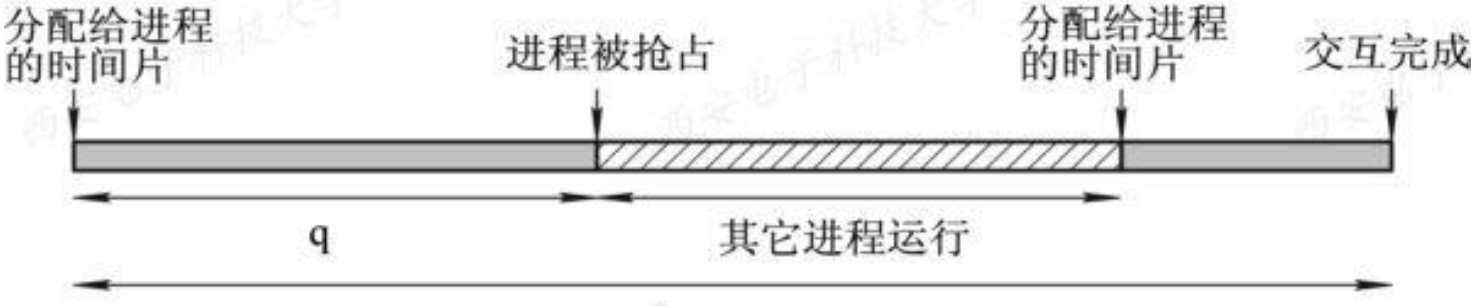
② 时间片用完，计时器中断处理程序被激活，如进程未完，调度程序将把它送往就绪队列末尾。

3. 时间片大小的确定

RR，时间片大小对系统性能有很大的影响



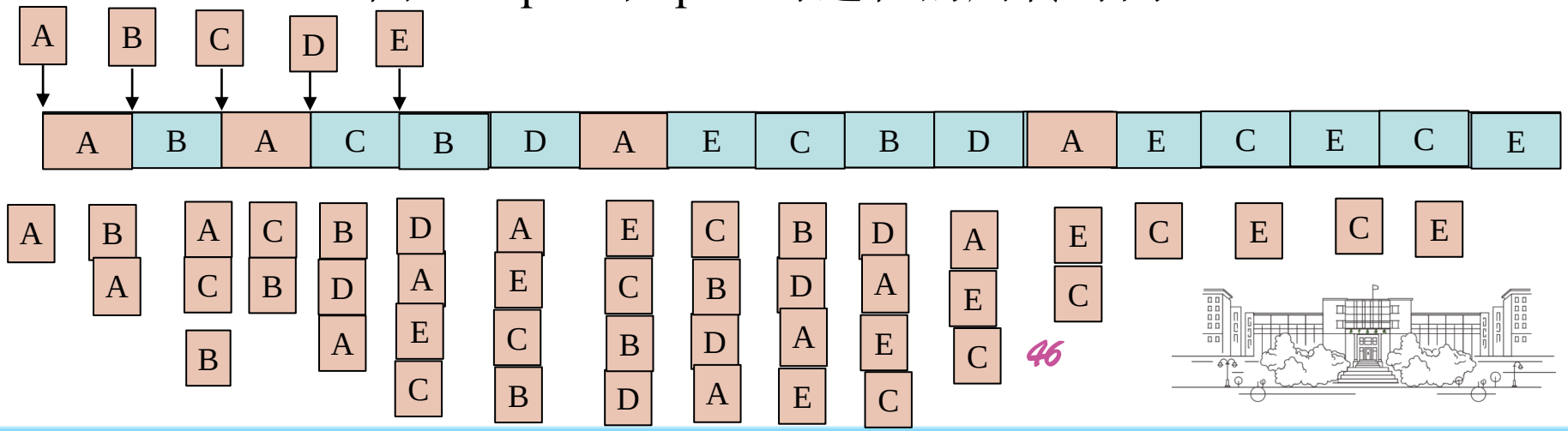
(a) 时间片大于交互时间



(b) 时间片小于交互时间

作业 / 情况 / 时间片	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	4	2	4	
RR q=1	完成时间	15	12	16	9	17	
	周转时间	15	11	14	6	13	11.8
	带权周转时间	3.75	3.67	3.5	3	3.33	3.46
RR q=4	完成时间	4	7	11	13	17	
	周转时间	4	6	9	10	13	8.4
	带权周转时间	1	2	2.25	5	3.33	2.5

图3-3 q = 1和q = 4时进程的周转时间



3.3.3 优先级调度算法

优先级调度算法的类型

优先级进程调度算法，是把处理机分配给就绪队列中优先级最高的进程。

考虑调度方式：

(1) 非抢占式优先级调度算法。

(2) 抢占式优先级调度算法。

3.3.4　多队列调度算法

仅设置一个进程就绪队列，固定的、单一，无法满足系统中不同用户对进程调度策略的不同要求。

在多处理机系统中，多级队列调度算法

3.3.5 多级反馈队列(multileved feedback queue)调度算法

1. 调度机制

(1) 设置多个就绪队列。

(2) 每个队列都采用FCFS算法

(3) 按队列优先级调度。

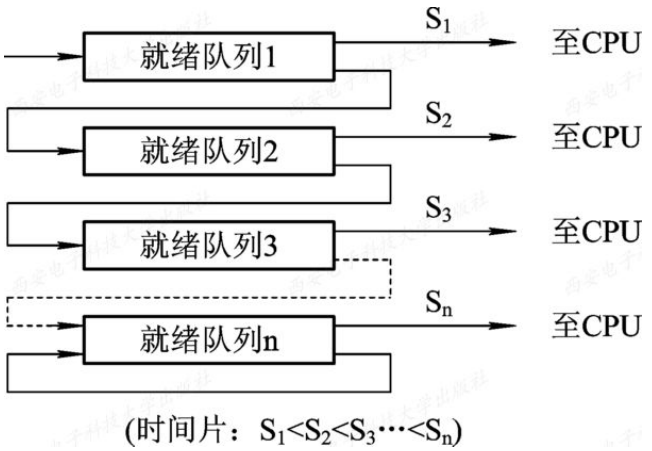
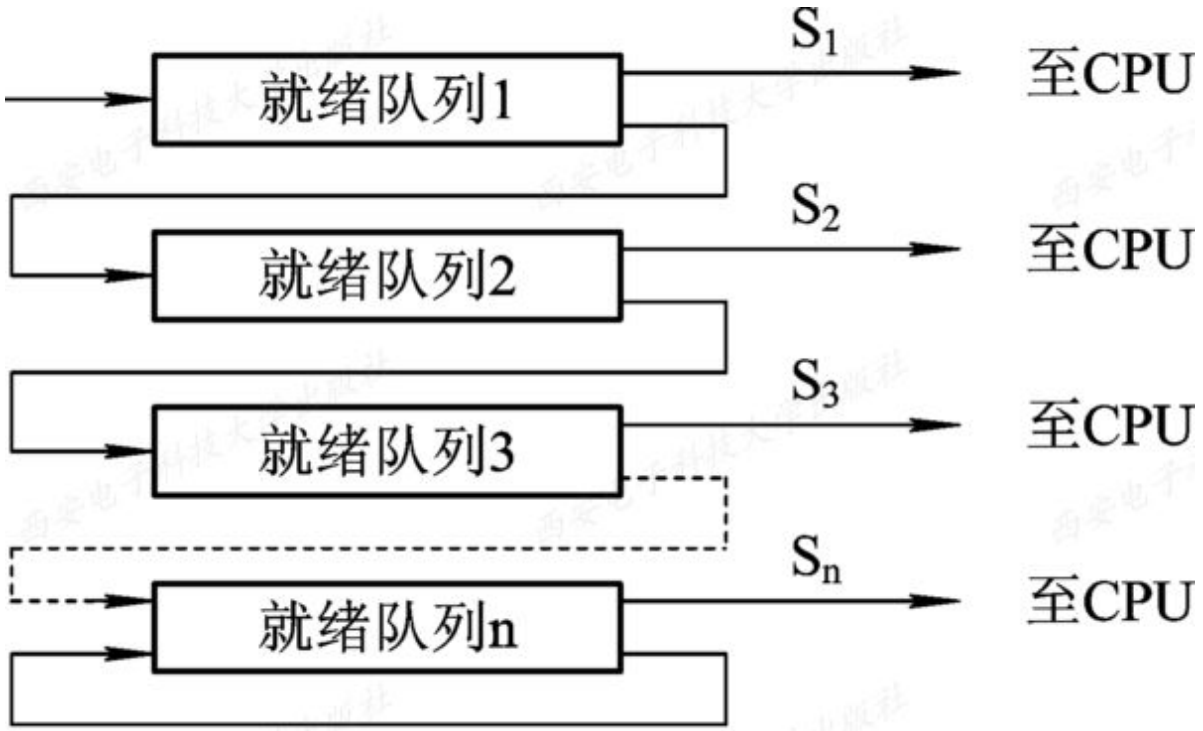


图3-4是多级反馈队列算法的示意图。

图3-4 多级反馈队列调度算法



(时间片：$S_1<S_2<S_3\cdots<S_n$)

(2) 每个队列都采用FCFS算法。

新进程先放入第一队列末尾，按FCFS等待调度。当轮到该进程执行时，如它能在该时间片内完成，便可撤离系统。

否则，它在一个时间片结束时尚未完成，调度程序将进程转入第二队列的末尾等待调度；

如果它在第二队列中运行一个时间片后仍未完成，再依次将它放入第三队列，……，依此类推。

当进程最后被降到第n队列后，在第n队列便采取按RR方式运行。

(3) 按队列优先级调度。

首先调度最高优先级队列中的诸进程运行，仅当第一队列空闲时才调度第二队列中的进程运行；

仅当第1～(i-1)所有队列均空时，才会调度第i队列中的进程运行。

如果处理机正在第i队列中为某进程服务时又有新进程进入任一优先级较高的队列，须立即把正在运行的进程放回到第i队列的末尾，而把处理机分配给新到的高优先级进程。

2. 调度算法的性能

在多级反馈队列调度算法中，如果规定第一个队列的时间片略大于多数人机交互所需之处理时间时，便能较好地满足各种类型用户的需要。

(1) 终端型用户。

(2) 短批处理作业用户。

(3) 长批处理作业用户。

3.3.6 基于公平原则的调度算法

1. 保证调度算法

明确的性能保证，调度公平性。

有n个相同类进程同时运行，为公平起见，须保证每个进程都获得相同的处理机时间1/n。

必须具有功能：

(1) 已执行时间。

(2) 应获得处理机时间

(3) 获得处理机时间比率，即实际执行处理时间和应获得处理机时间之比。

(4) 比较各进程获得处理机时间的比率。

(5) 选择比率最小的进程，分配处理机，进程运行，直到超过最接近它的进程比率为止。

2. 公平分享调度算法

分配给每个进程相同处理机时间，对诸进程体现一定程度公平

如果各用户拥有进程数不同，对用户不公平

CFS（完全公平调度算法）

根据各个进程的权重分配运行时间

运行时间计算公式为:

分配给进程的运行时间 = *调度周期* 进程权重 / 所有进程权重之和

3.4 实 时 调 度

为保证系统能正常工作，实时调度必须能满足实时任务对截止时间的要求。实现实时调度应具备一定的条件。 两类不同性质的实时任务：

硬实时任务HRT

要求计算机系统必须在用户给定的时限内完成

软实时任务SRT

允许计算机系统在用户给定的时限左右处理完毕。

3.4.1 实现实时调度的基本条件

1. 提供必要的信息

任务信息：

(1) 就绪时间

(2) 开始截止时间和完成截止时间

(3) 处理时间

(4) 资源要求

(5) 优先级

2. 系统处理能力强

若处理机处理能力不强，则有可能因处理机忙不过，而致使实时任务不能得到及时处理，发生难以预料的后果。

假定系统中有m个周期性的硬实时任务HRT，它们的处理时间可表示为C_i，周期时间表示为P_i，则在单处理机情况下，必须满足的限制条件系统才是可调度的：

$$\sum_{i=1}^{m} \frac{C_i}{P_i} \leq 1$$

提高系统处理能力途径：

一是采用单处理机系统，但须增强其处理能力，以显著地减少对每一个任务的处理时间；

二是采用多处理机系统。假定系统中的处理机数为N，则应将上述的限制条件改为：

$$\sum_{i=1}^{m} \frac{C_i}{P_i} \leq N$$

3. 采用抢占式调度机制

在含有HRT任务的实时系统中，广泛采用抢占机制。以满足HRT任务对截止时间的要求，但调度机制比较复杂。

4. 具有快速切换机制

(1) 对中断的快速响应能力。

具有快速硬件中断；禁止中断时间间隔尽量短，

(2) 快速的任务分派能力。

系统每个运行功能单位适当小

3.4.2 实时调度算法的分类

① 根据实时任务性质，

硬实时调度算法；软实时调度算法；

② 按调度方式，

非抢占调度算法；抢占调度算法。

1. 非抢占式调度算法

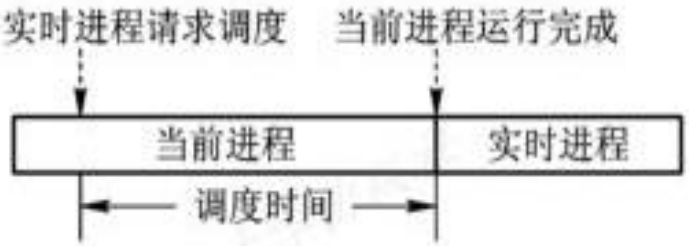
(1) 非抢占式轮转调度算法。

轮转队列　不太严格实时系统

(2) 非抢占式优先调度算法。

优先级

(a) 非抢占轮转调度



(b) 非抢占优先权调度

2. 抢占式调度算法

抢占发生时间不同：

(1) 基于时钟中断的抢占式优先级调度算法。

时钟中断发生后 抢占

(2) 立即抢占的优先级调度算法。

只要未处于临界区，立即抢占

(c) 基于时钟中断抢占的优先权抢占调度

(d) 立即抢占的优先权调度

3.4.3 最早截止时间优先EDF算法

(Earliest Deadline First)

根据任务的截止时间确定任务的优先级，优先级高的排在队首。

抢占式调度方式

非抢占调度方式

1. 非抢占式调度方式用于非周期实时任务 P100

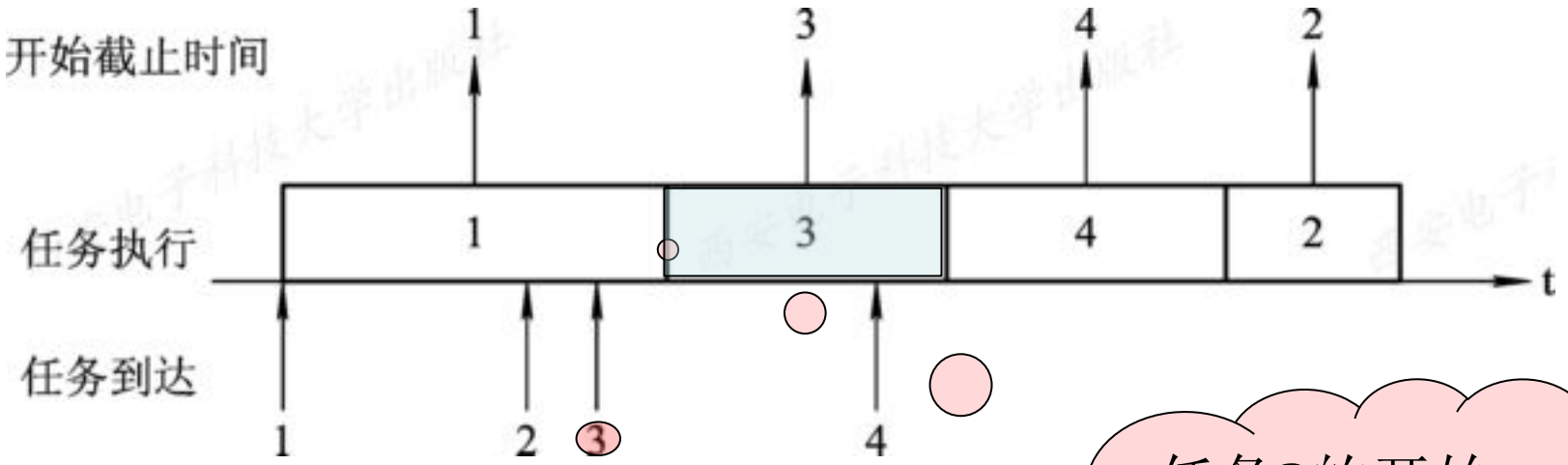


图3-6 EDF算法用于非抢占调度方式

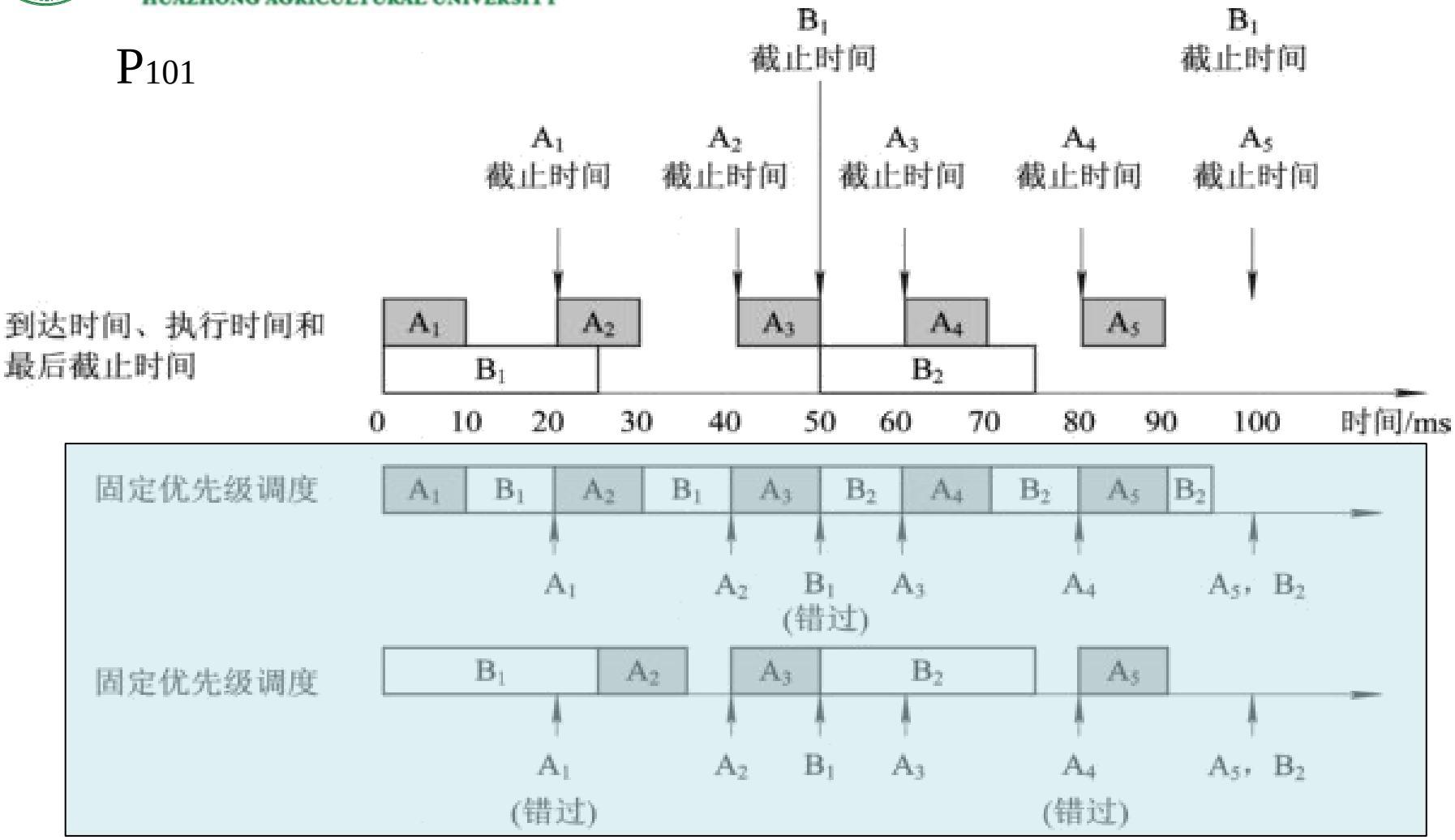
任务3的开始截止时间早于任务2

2. 抢占式调度方式用于周期实时任务

例：有两个周期任务，任务A和任务B的周期时间分别为20 ms和50 ms，每个周期的处理时间分别为10 ms和25 ms。

通常的优先级调度失效。

P101



通常的优先级调度失效

图3-7 最早截止时间优先算法用于抢占调度方式之例

P101

最早截止时间优先

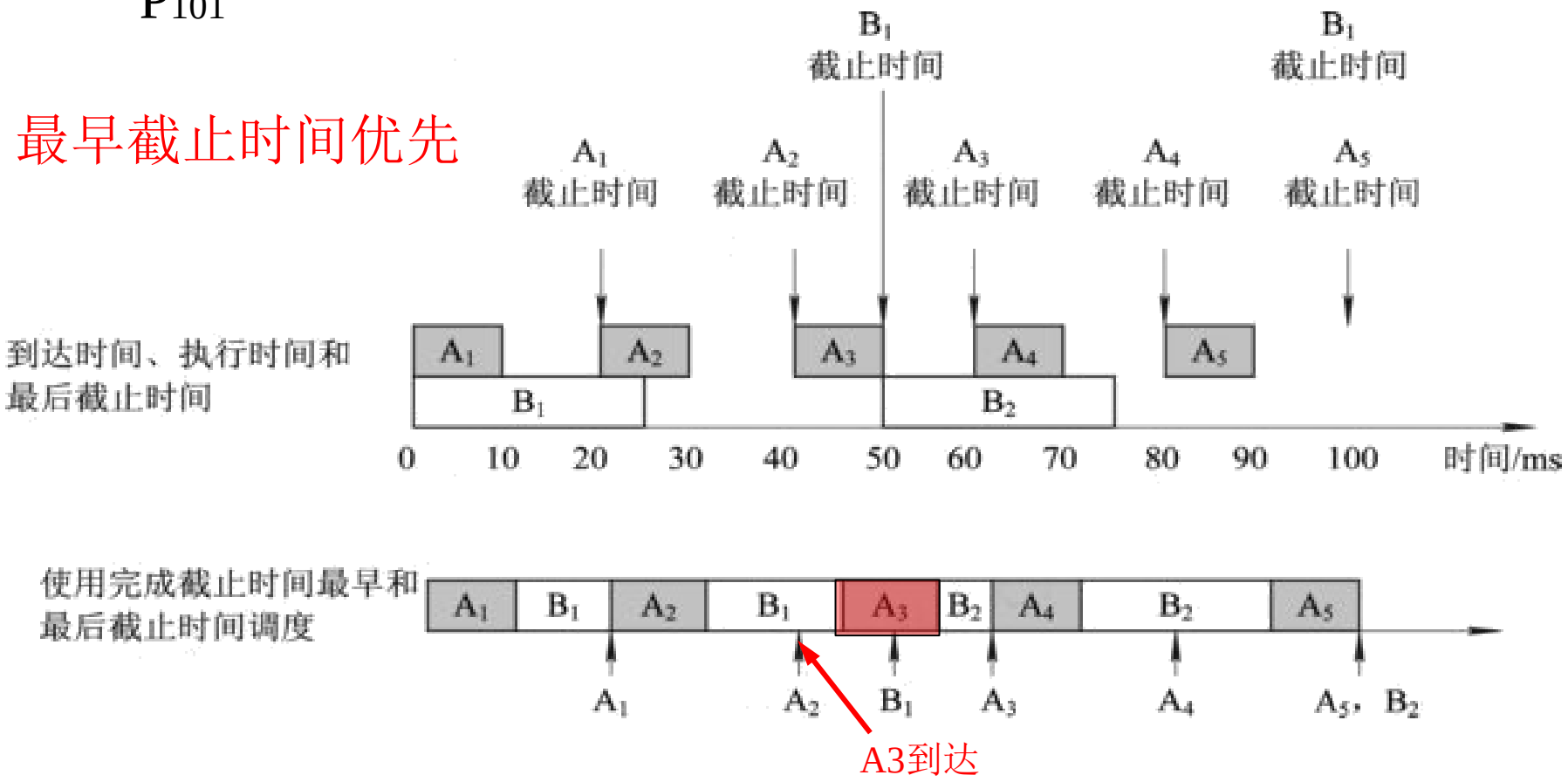


图3-7 最早截止时间优先算法用于抢占调度方式之例

3.4.4 最低松弛度优先LLF算法 (Least Laxity First)

根据任务紧急(或松弛)程度确定任务优先级。任务紧急程度愈高，赋予该任务的优先级就愈高，以使之优先执行。

松弛度=必须完成的时间-其本身的运行时间-当前时间

其本身运行的时间指任务运行结束还需多少时间，如果任务已经运行了一部分，则：

其本身运行的时间=任务的处理时间-任务已经运行的时间

按松弛度排序的就绪队列，松弛度低位于队首。

主要用于可抢占调度方式。

3.4.4 最低松弛度优先LLF算法（Least Laxity First）

1. 主要用于可抢占调度方式中，当一任务的最低松弛度减为0时，它必须立即抢占CPU，以保证按截止时间的要求完成任务。

2. 计算关键时间点(任务执行完、)的各进程周期的松弛度，当进程在当前周期截止时间前完成了任务，则在该进程进入下个周期前，无需计算它的松弛度。

3. 当出现多个进程松弛度相同且为最小时，按照“最近最久未调度”的原则进行进程调度。

有两个周期性实时任务A和B，任务A要求每20 ms执行一次，执行时间为10 ms，任务B要求每50 ms执行一次，执行时间为25 ms。

任务A和B每次必须完成的时间分别为：A_1、A_2、A_3、…和B_1、B_2、B_3、…，

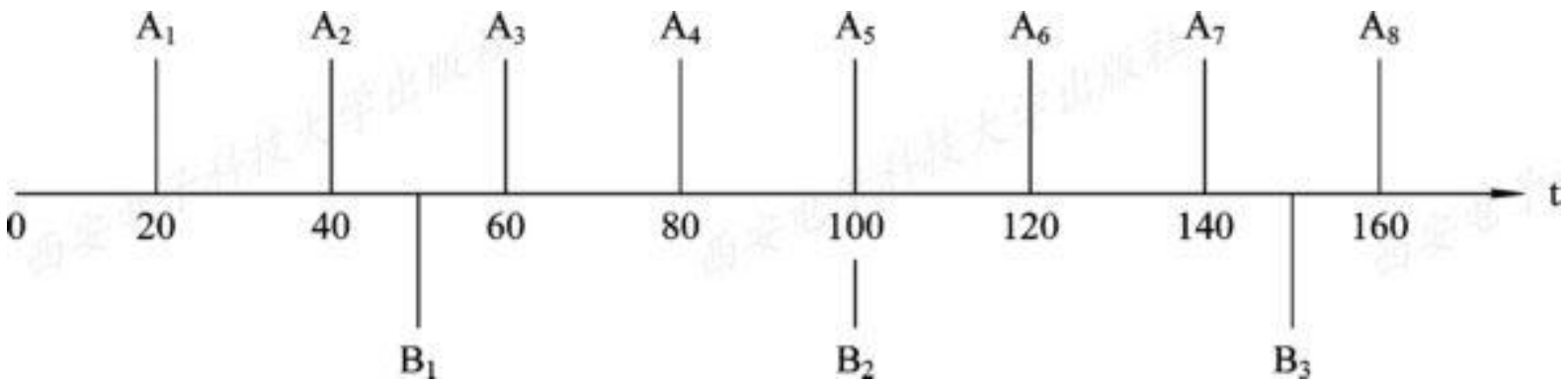


图3-8 A和B任务每次必须完成的时间

最低松弛度优先的抢占式调度

P102

松弛度=必须完成时间-本身需运行时间-当前时间

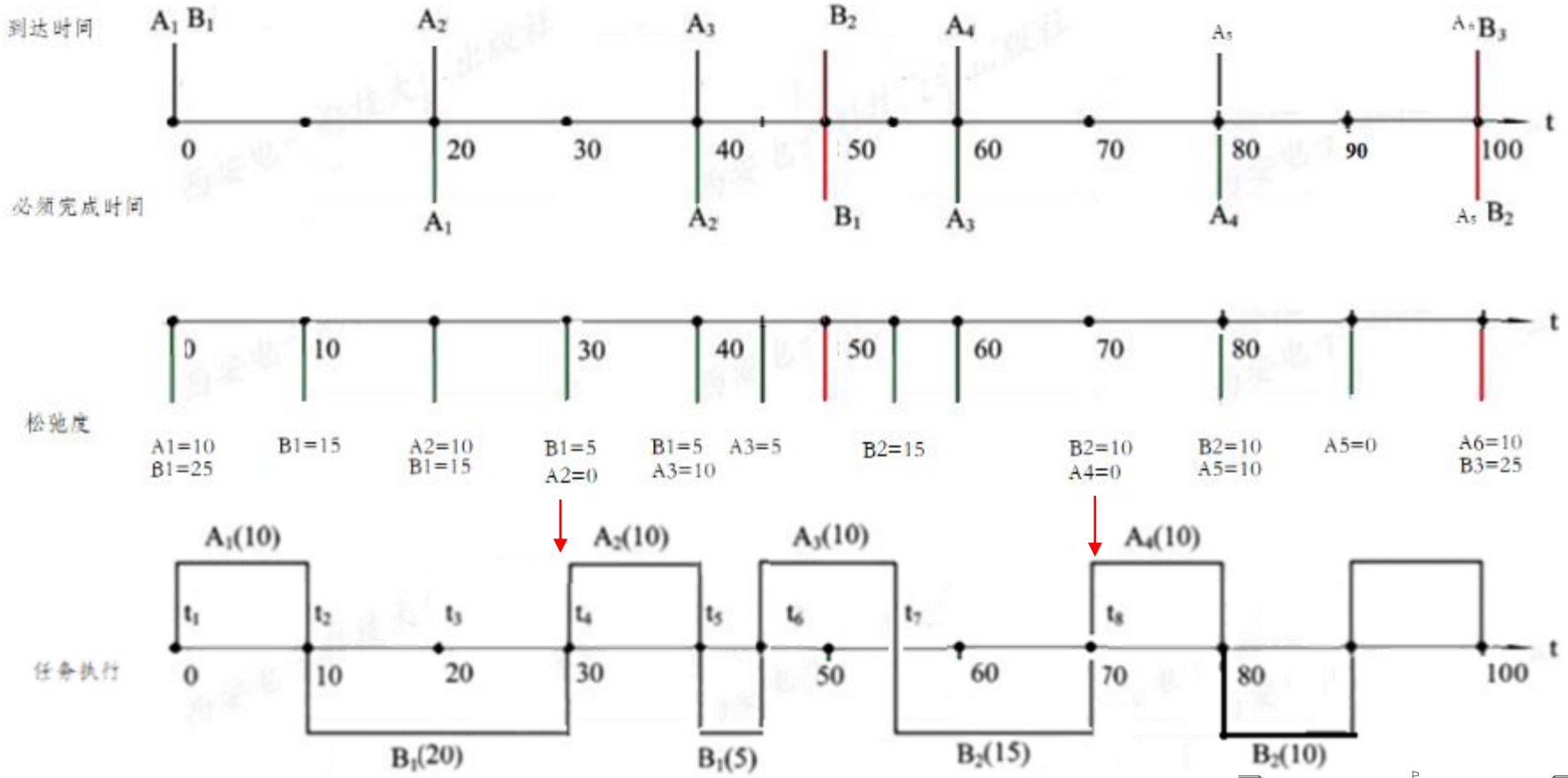


图3-9　利用LLF算法进行调度的情况

3.4.5 优先级倒置

(priority inversion problem)

1. 优先级倒置的形成

OS中广泛采用的优先级调度算法和抢占方式。

“优先级倒置”现象，即高优先级进程(或线程)被低优先级进程(或线程)延迟或阻塞

P102

例: 三个独立进程P1、P2、P3，优先级由高到低。P1、P3共享临界资源进行交互。代码：

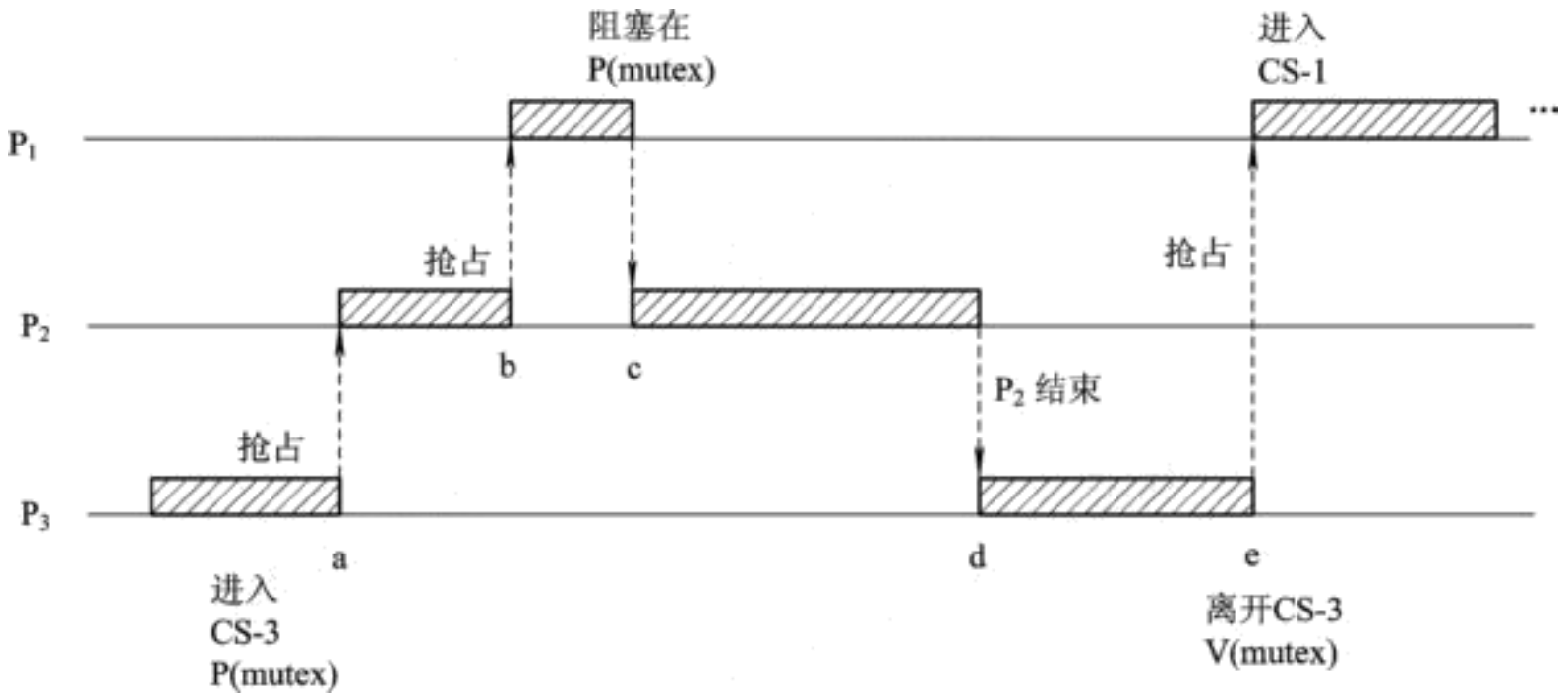
P1：... P(mutex); CS-1; V(mutex) ...;

P2: ... Program2 ...;

P3: ... P(mutex); CS-3; V(mutex) ...;

假设 P3 先执行，列出进程执行顺序？

假如P_3最先执行，顺序为 P3—>P2(抢占)—>P1（阻塞）—>P2(执行结束)—>P3(执行结束)—>P1(执行结束)



问题：P1优先级最高，但最后执行结束

2. 优先级倒置的解决方法

1）进程在进入临界区后所占用的处理就不允许被抢占。简单，高优先级进程等待

2）用动态优先级继承方法

规定：P1阻塞时由P3继承P1的优先级，一直保持到P3退出临界区。

目的：防止P2进程插进来，延缓P3退出临界区。

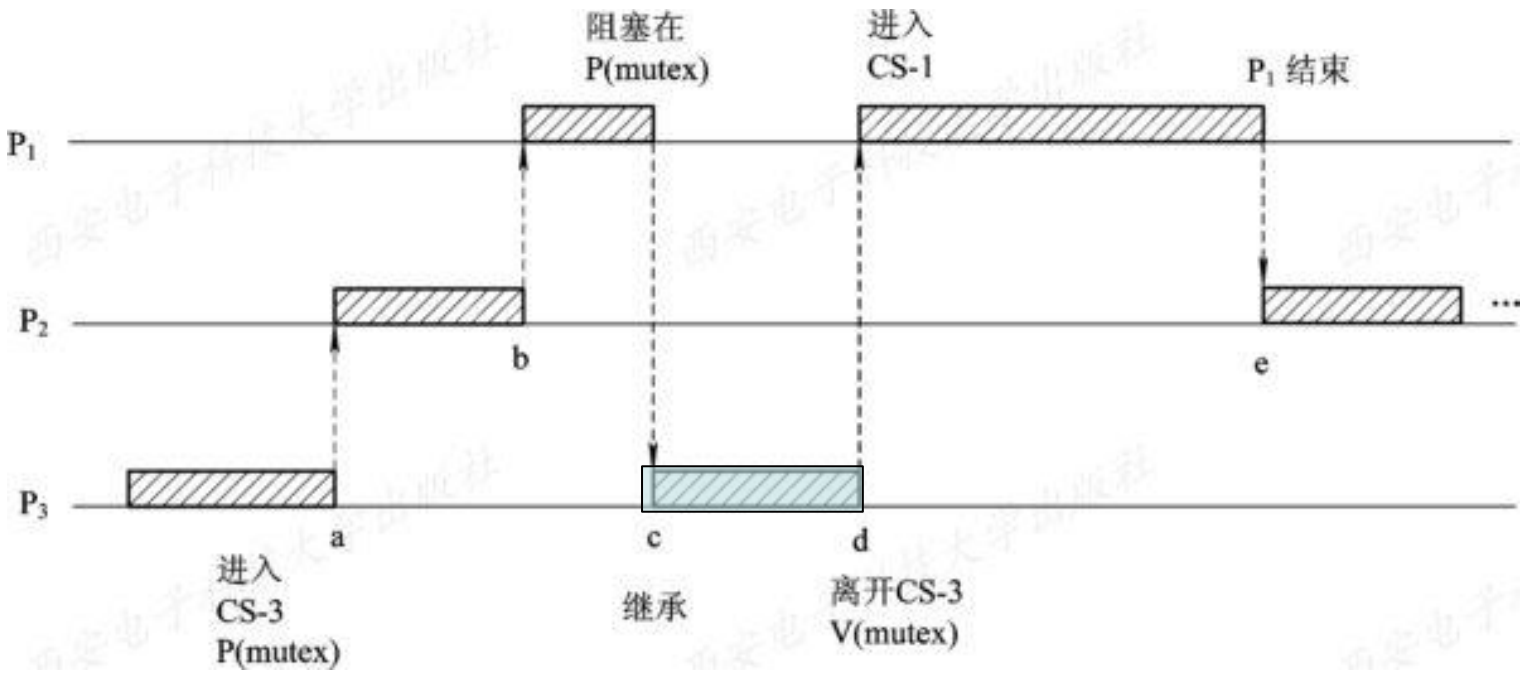


图3-11　采用了动态优先级继承方法的运行情况

进入临界区，　动态优先级继承 避免优先级低的进程插入

设置

在线听课评价:

A 教师准备充分,讲述条理清晰

B 能调动学习的积极性

C 教学方式单一、缺乏互动

D 学生的收获少，获得感不强

E 内容枯燥、听不懂

提交

主观题 10分

设置

HUAZHONG AGRICULTURAL UNIVERSITY

在本课堂上，你想学习那些内容？
你有那些建议？

作答

正常使用主观题需2.0以上版本雨课堂