深圳大学
Shenzhen University

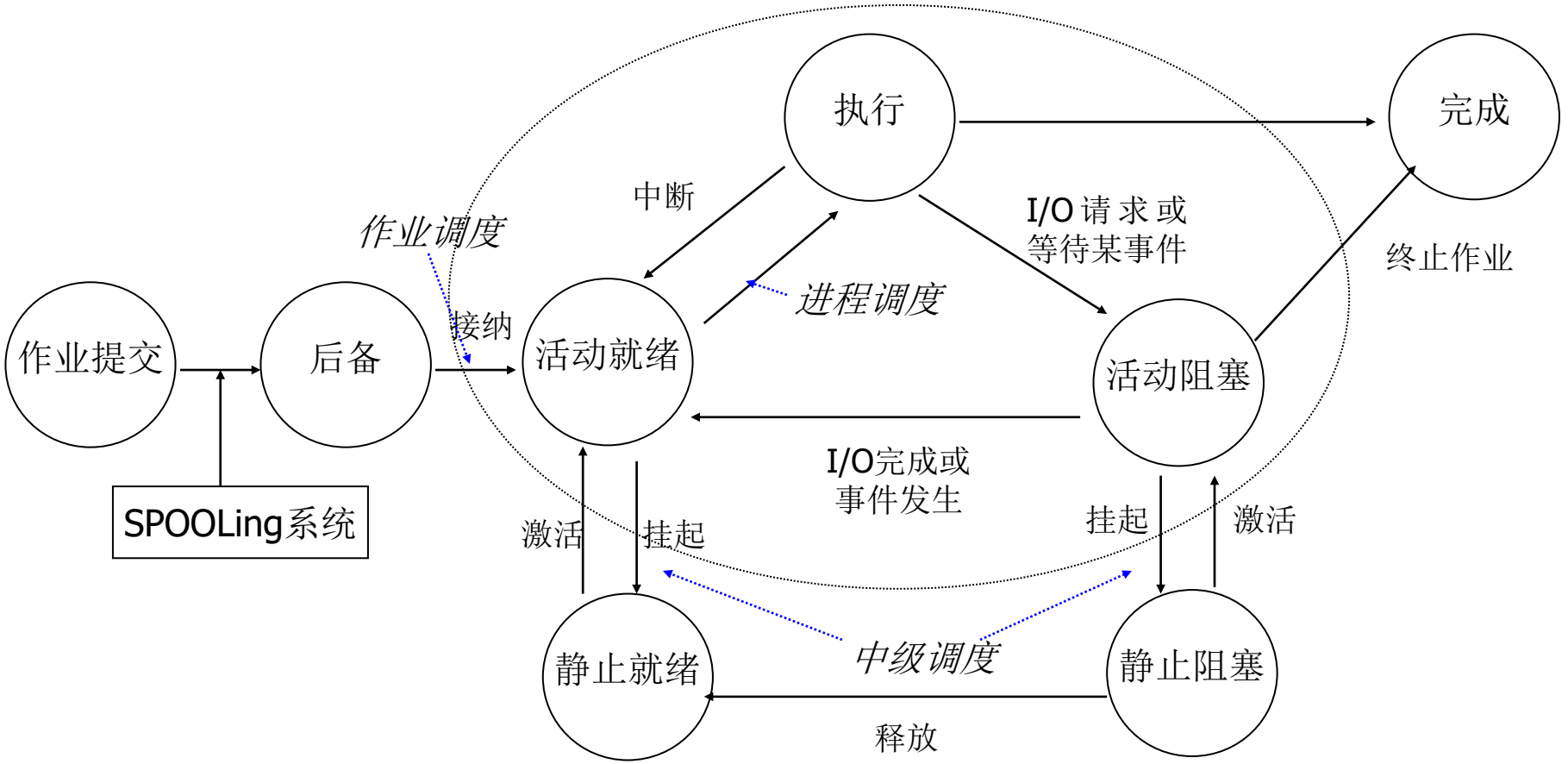
操作系统

第四讲 处理机调度与死锁

谭舜泉
计算机与软件学院

第4章　处理机调度与死锁

第一节　调度级别



第4章　处理机调度与死锁

第一节　调度级别

一、高级调度

- 高级调度即作业调度
- 它决定允许哪些作业参与竞争CPU和其它系统资源
- 将一个或一批作业从后备状态变为运行状态
- 高级调度是为作业分配虚拟处理机，也称宏观调度

第4章　处理机调度与死锁

第一节　调度级别

一、高级调度需要解决的问题

1、接纳多少作业

- 周转时间：太多会增加周转时间
- 吞吐量：太少会降低计算机吞吐量

2、接纳哪些作业

- 调度算法，如先来先服务，短作业优先等

第4章　处理机调度与死锁

第一节　调度级别

一、高级调度

3、高级调度功能

- 根据调度算法和计算机当前状态，挑选一个或多个后备作业投入运行
- 为选中的作业分配基本的内存和设备资源
- 为选中的作业建立进程，将进程实体装入内存

第4章　处理机调度与死锁

第一节　调度级别

一、高级调度

4、高级调度时机

- 如果当前运行的作业数目少于计算机系统允许的最大作业数目，且存在后备作业
- 当一作业运行终止而被撤销后，且存在后备作业
- 在分时系统中，当一用户被核准注册

第4章　处理机调度与死锁

第一节　调度级别

二、中级调度

- 中级调度决定哪些进程可参与竞争CPU
- 中级调度将进程从活动态（活动就绪、活动阻塞）变为静止的挂起态（静止就绪、静止阻塞）；或相反
- 中级调度实际上是实现“挂起”和“激活”操作
- 中级调度也称为进程交换调度，通常仅用于分时系统

第4章　处理机调度与死锁

第一节　调度级别

三、低级调度

- 低级调度即进程调度
- 低级调度决定哪个进程可获得CPU
- 低级调度从活动就绪队列中挑选一个进程，将它变为运行态，同时启动CPU执行该进程
- 低级调度也称微观调度

第4章　处理机调度与死锁

第一节　调度级别

三、低级调度

1、低级调度方式

- 非抢占方式

进程获得CPU后，一直执行完成或自动阻塞

第4章　处理机调度与死锁

第一节　调度级别

三、低级调度

1、低级调度方式

- **抢占方式**

  进程获得CPU后，允许其它进程依据一定的原则抢占CPU资源

(1)、时间片原则

(2)、优先权原则

(3)、短作业（进程）优先原则

第4章　处理机调度与死锁

第一节　调度级别

三、低级调度

2、低级调度时机

- **现行进程执行完成或自动阻塞**
- **在采用抢占方式的系统中，发生了某种抢占事件**

第4章　处理机调度与死锁

第一节　调度级别

四、仅有低级调度的调度队列模型

- 每个进程执行时：

1、进程在时间片内已完成，进入完成状态

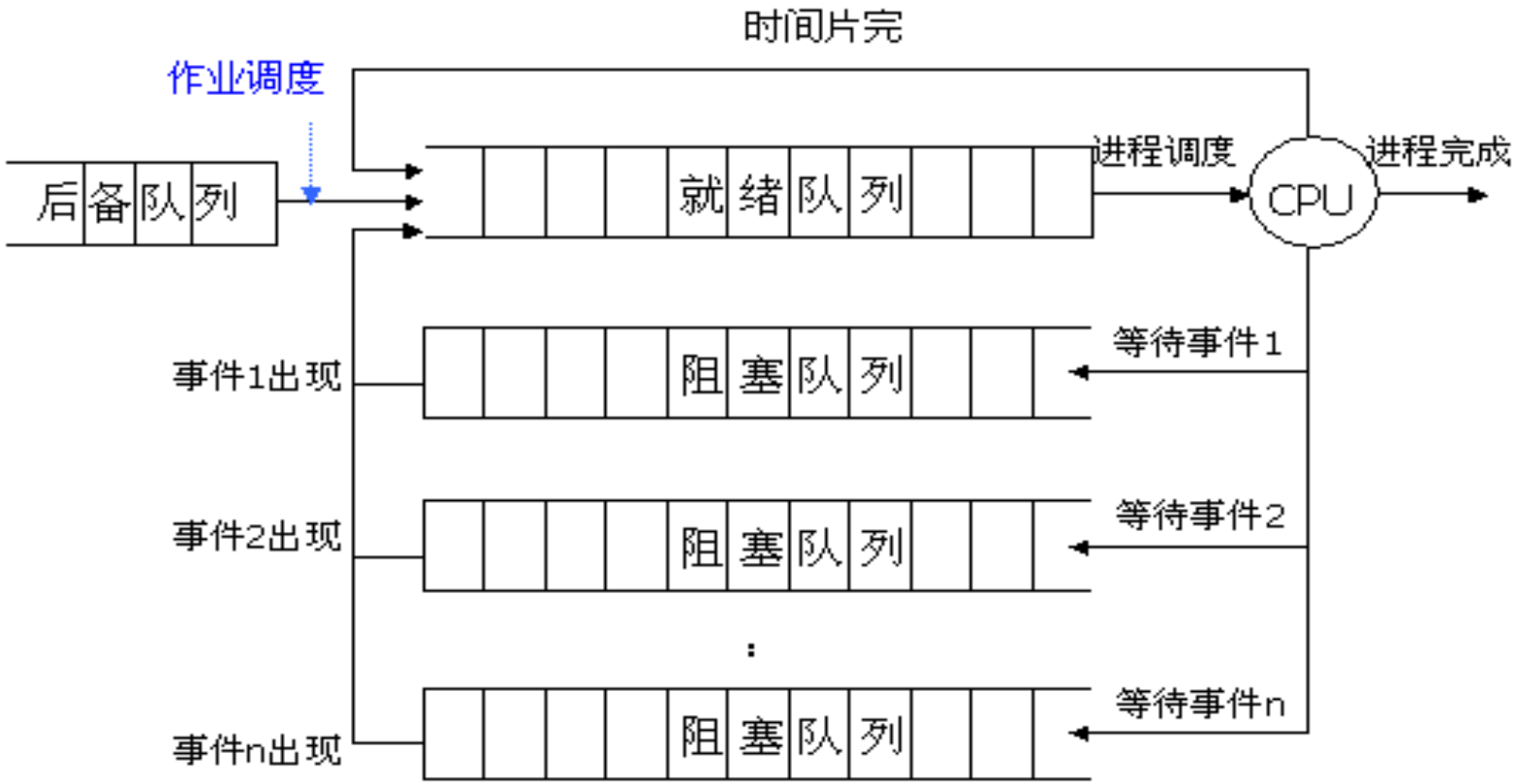
2、未完成，放在就绪队列后

3、因某事件，进入阻塞状态

第4章　处理机调度与死锁

第一节　调度级别

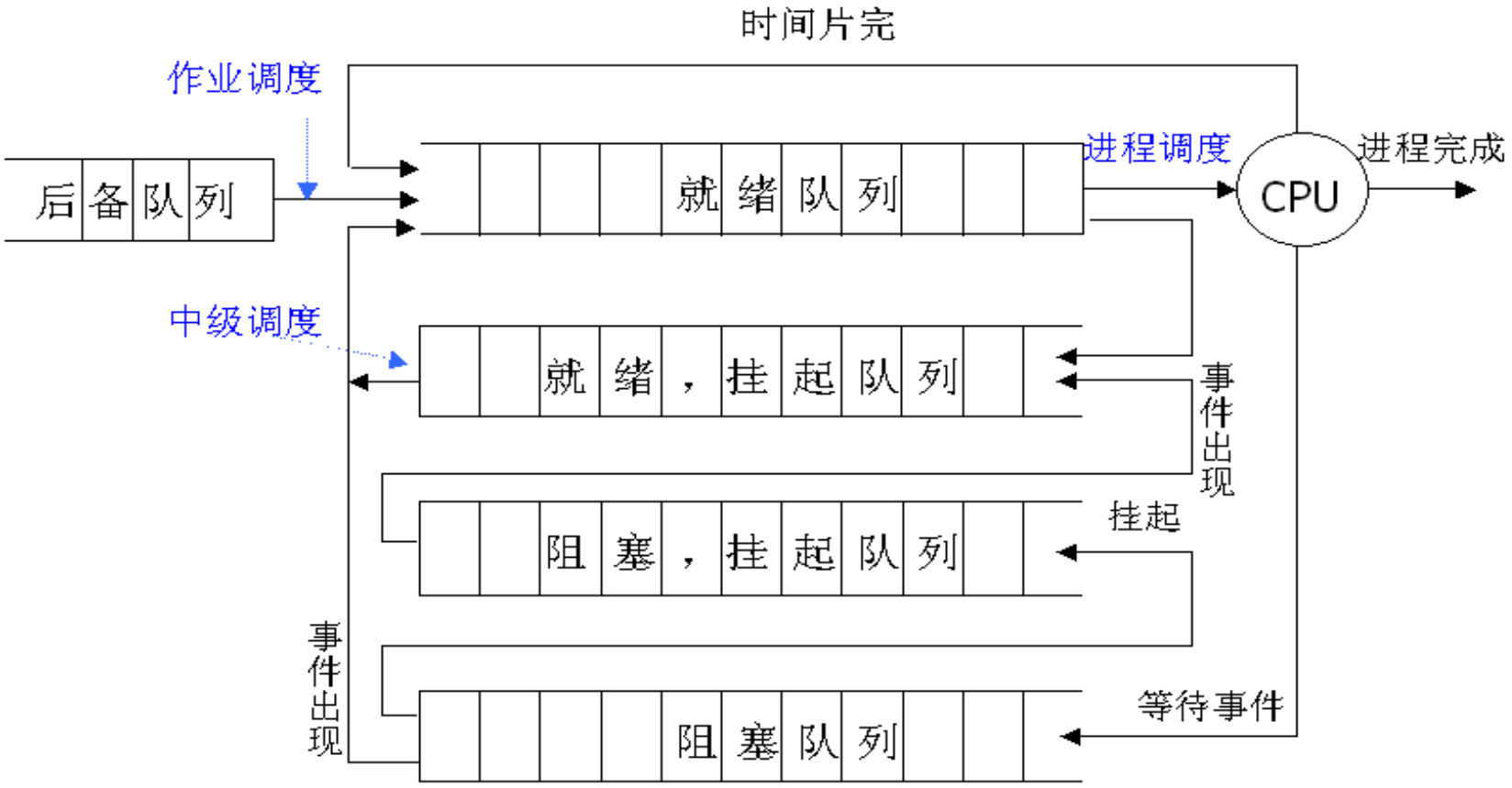
五、具有高级和低级进程调度的调度队列模型



第4章　处理机调度与死锁

第一节　调度级别

六、具有三级调度的调度队列模型



第4章　处理机调度与死锁

第二节　调度原则与评估标准

一、调度评估标准

1、周转时间

- 作业（进程）i从提交（进入时刻）到完成的时间称为该作业的周转时间T_i

$$T_i = \text{完成时刻} - \text{进入时刻}$$

第4章　处理机调度与死锁

第二节　调度原则与评估标准

一、调度评估标准

2、平均周转时间

- 平均周转时间为n个作业（进程）周转时间的平均值
- n个作业（进程）的平均周转时间：

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

第4章　处理机调度与死锁

第二节　调度原则与评估标准

一、调度评估标准

3、带权周转时间

- 作业(进程)周转时间T_i与实际运行时间T_{si}之比称为该作业的带权周转时间W_i

$$W_i = \frac{T_i}{T_{si}}$$

第4章　处理机调度与死锁

第二节　调度原则与评估标准

一、调度评估标准

4、平均带权周转时间

- 平均带权周转时间为n个作业（进程）带权周转时间的平均值
- n个作业（进程）的平均带权周转时间：

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

第4章　处理机调度与死锁

第二节　调度原则与评估标准

一、调度评估标准

3、平均等待时间

- 进程i从进入就绪队列到获得CPU的时间称为该进程的等待时间WT_i
- n个进程的平均等待时间：

$$WT = \frac{1}{n}\left[\sum_{i=1}^{n} WT_i\right]$$

第4章　处理机调度与死锁

第二节　调度原则与评估标准

二、调度原则

1、面向用户的原则

- 周转时间短
- 响应时间快
- 截止时间的保证
- 优先权准则

第4章　处理机调度与死锁

第二节　调度原则与评估标准

二、调度原则

2、面向系统的原则

- **系统吞吐量高**
- **处理机利用率好**
- **各类资源的平衡使用**

第4章　处理机调度与死锁

第三节　调度算法

一、调度算法

- 处理机调度实际上是一种资源(处理机)分配
- 调度算法是指根据系统的资源分配策略所规定的资源分配算法
- 目标不同、系统不同，则调度算法各异

第4章　处理机调度与死锁

第三节　调度算法

二、先来先服务（FCFS）调度算法

- **作业调度：**

  从后备队列中，选择一个或多个最先进入该队列的作业

- **进程调度**

  从就绪队列中，选择一个最先进入该队列的进程，把处理机分配给它，一直运行完或自动阻塞

第4章　处理机调度与死锁

第三节　调度算法

二、先来先服务（FCFS）调度算法

- **作业调度举例**

作业	进入时刻	运行时间	开始时刻	完成时刻	周转时间	带权周转时间
A	8.00	2.00	8.00	10.00	2.00	1.00
B	8.50	0.50	10.00	10.50	2.00	4.00
C	9.00	0.10	10.50	10.60	1.60	16.00
D	9.50	0.20	10.60	10.80	1.30	6.50
平均周转时间T=1.725单位时间 平均带权周转时间W=6.875					6.90	27.50

第4章　处理机调度与死锁

第三节　调度算法

二、先来先服务（FCFS）调度算法

- **进程调度举例**

进程	进入时刻	运行时间	开始时刻	完成时刻	周转时间	带权周转时间
1	0	21	0	21	21	1
2	0	6	21	27	27	4.5
3	0	3	27	30	30	10
平均周转时间T=26单位时间(ms) 平均带权周转时间W=5.17					78	15.5

第4章　处理机调度与死锁

第三节　调度算法

二、先来先服务（FCFS）调度算法

- **FCFS特性**

以非抢占方式工作，可用于作业与进程调度

第4章　处理机调度与死锁

第三节　调度算法

二、先来先服务（FCFS）调度算法

- 优点：

  简单，有利于CPU繁忙型作业（进程），有利于长时间作业（进程）

- 缺点：

  对短时间作业（进程）不利，对I/O繁忙型作业（进程）不利，对紧迫作业（进程）不利

第4章　处理机调度与死锁

第三节　调度算法

三、短作业(进程)优先（SF）调度算法

- **作业调度：**

  从后备队列中，选择一个或多个*估计*执行时间最短的作业

- **进程调度**

  从就绪队列中，选择一个*估计*需CPU时间最短的进程，把处理机分配给它

第4章　处理机调度与死锁

第三节　调度算法

三、短作业(进程)优先（SF）调度算法

- **作业调度举例**

作业	进入时刻	运行时间	开始时刻	完成时刻	周转时间	带权周转时间
A	8.00	2.00	8.00	10.00	2.00	1.00
B	8.50	0.50	10.30	10.80	2.30	4.60
C	9.00	0.10	10.00	10.10	1.10	11.00
D	9.50	0.20	10.10	10.30	0.80	4.00
平均周转时间T=1.55单位时间 平均带权周转时间W=5.15					6.2	20.60

第4章　处理机调度与死锁

第三节　调度算法

三、短作业(进程)优先（SF）调度算法

- **进程调度举例**

进程	进入时刻	运行时间	开始时刻	完成时刻	周转时间	带权周转时间
1	0	21	9	30	30	10/7
2	0	6	3	9	9	1.5
3	0	3	0	3	3	1
平均周转时间T=14单位时间(ms) 平均带权周转时间W=1.31					42	3.93

第4章　处理机调度与死锁

第三节　调度算法

三、短作业(进程)优先（SF）调度算法

- **SF特性**

  可以抢占方式和非抢占方式工作

  可用于作业调度与进程调度

第4章　处理机调度与死锁

第三节　调度算法

三、短作业(进程)优先（SF）调度算法

- **优点：**

  有利于短时作业（进程）

  在抢占方式中，最短时作业（进程）将以最快的速度完成

第4章　处理机调度与死锁

第三节　调度算法

三、短作业(进程)优先（SF）调度算法

- **缺点：**

(1)、对长时间作业（进程）不利

(2)、未考虑作业（进程）的紧迫程度

(3)、在抢占方式中，最短指总需要时间最短还是剩余时间最短（而且是估计值）

(4)、在抢占方式下，即使一个长作业（进程）正在运行，但也可能会被长时间地延迟

第4章　处理机调度与死锁

第三节　调度算法

四、高响应比优先（HRN）调度算法

- **响应比R_P**

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

$$= 1 + \frac{\text{已等待时间}}{\text{要求服务时间}}$$

要求服务时间　＝　运行时间

第4章　处理机调度与死锁

第三节　调度算法

四、高响应比优先（HRN）调度算法

- **作业调度：**
  从后备队列中，选择一个响应比Rp最高的作业

- **进程调度**
  从就绪队列中，选择一个响应比Rp最高的进程，把处理机分配给它

第4章　处理机调度与死锁

第三节　调度算法

四、高响应比优先（HRN）调度算法

- **作业调度举例**

作业	进入时刻	运行时刻	开始时刻	完成时刻	响应比1	响应比2	响应比3	响应比4	周转时间	带权周转时间
A	8.00	2.00	8.00	10.00	1				2.00	1.00
B	8.50	0.50	10.10	10.60	0	4	4.2		2.10	4.20
C	9.00	0.10	10.00	10.10	0	11			1.10	11.00
D	9.50	0.20	1060	10.80	0	3.5	4	6.5	1.30	6.50
平均周转时间T=1.625单位时间　　平均带权周转时间W=5.675									6.5	22.70

第4章　处理机调度与死锁

第三节　调度算法

四、高响应比优先（HRN）调度算法

- **进程调度举例**

进程	进入时刻	运行时间	开始时刻	完成时刻	周转时间	带权周转时间
1	0	21	9	30	30	10/7
2	0	6	3	9	9	1.5
3	0	3	0	3	3	1
平均周转时间T=14单位时间(ms) 平均带权周转时间W=1.31					42	3.93

第4章　处理机调度与死锁

第三节　调度算法

四、高响应比优先（HRN）调度算法

- **HRN特性**

(1)、可用于作业与进程调度，是FCFS和SF的折衷算法

(2)、Rp与需要运行时间成反比，使短作业（进程）Rp较高

(3)、Rp与等待时间成正比，使等待时间长的作业（进程）Rp较高

(4)、HRN一般采用非抢占方式

第4章　处理机调度与死锁

第三节　调度算法

四、高响应比优先（HRN）调度算法

- **优点：**

  有利于短时作业（进程）

  有利于先来者

第4章　处理机调度与死锁

第三节　调度算法

四、高响应比优先（HRN）调度算法

- **缺点：**

  每次调度前，必须计算Rp，增加系统开销

  未考虑作业（进程）的紧迫程度

第4章　处理机调度与死锁

第三节　调度算法

五、最高优先权（HPF）调度算法

- **作业调度：**

  从后备队列中，选择一个优先权最高的作业

- **进程调度**

  从就绪队列中，选择一个优先权最高的进程，把处理机分配给它

第4章　处理机调度与死锁

第三节　调度算法

五、最高优先权（HPF）调度算法

- **进程调度的非抢占方式**

  系统一旦将CPU分配之后，一直到进程执行完成或自动进入阻塞状态

- **进程调度的抢占方式**

  只要进入就绪队列的进程优先权高于正在CPU上运行现进程的优先权，则将现进程放到就绪队列尾，并运行最高优先权进程

第4章　处理机调度与死锁

第三节　调度算法

五、最高优先权（HPF）调度算法

- **静态优先权**

(1)、在作业（进程）被调度之前赋值，之后一直不变

(2)、优先权的取值可根据：作业（进程）类型，对资源的要求，用户要求等

第4章　处理机调度与死锁

第三节　调度算法

五、最高优先权（HPF）调度算法

- **动态优先权**

(1)、在作业（进程）被调度之前赋初值

(2)、在作业（进程）运行过程中，动态改变优先权

第4章　处理机调度与死锁

第三节　调度算法

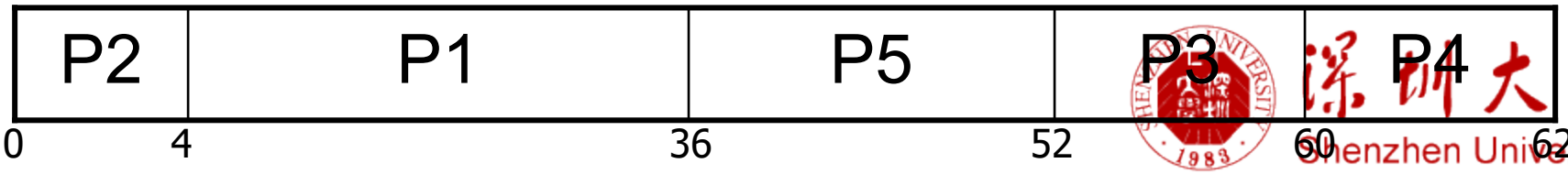
五、最高优先权（HPF）调度算法

- **静态优先权、非抢占方式进程HPF调度举例**

进程	运行时间	优先权	进入时间	周转时间	带权周转时间
P1	32	2	0	36	1.125
P2	4	4	0	4	1
P3	8	1	0	60	7.5
P4	2	0	0	62	31
P5	16	3	16	36	2.25

平均周转时间T=39.6

平均带权周转时间　　W=8.575



第4章　处理机调度与死锁

第三节　调度算法

五、最高优先权（HPF）调度算法

- 特性

(1)、可用于作业与进程调度，但动态优先权算法只用于进程调度

(2)、抢占方式可用于实时系统

第4章　处理机调度与死锁

第三节　调度算法

五、最高优先权（HPF）调度算法

- 特性

  采用非抢占方式

(1)、优先权只依据到达时间，则变为FCFS

(2)、优先权只依据运行时间，则变为SF

(3)、优先权只依据Rp，则变为HRN

第4章　处理机调度与死锁

第三节　调度算法

五、最高优先权（HPF）调度算法

- **优点：**

  可以根据要求，照顾到对系统、用户综合来说最优先的作业（进程）的执行

- **缺点：**

  优先权的计算可能比较复杂，增加系统开销

第4章　处理机调度与死锁

第三节　调度算法

六、时间片轮转（RR）调度算法

- RR调度算法是一种抢占方式的进程调度算法
- RR依据公平服务原则，在一定时间内，为每个进程轮转服务一次
- RR每次为一个进程执行一个时间片

第4章　处理机调度与死锁

第三节　调度算法

六、时间片轮转（RR）调度算法

- **时间片大小的确定**

(1)、系统对响应时间T的要求

(2)、就绪队列中进程的数目N

与系统中的最大终端数相关

T=Nq（T—次轮转时间，N用户数，q时间片大小）

(3)、系统的处理能力

保证每个用户的普通命令能在一个时间片内完成

第4章　处理机调度与死锁

第三节　调度算法

六、时间片轮转（RR）调度算法

- **等长时间片**

  在保证系统对响应时间的要求，满足最大用户需要的前提下，时间片应尽量长

- **不等长时间片**

  可以根据优先权，对优先权高的进程分配较长的时间片，但总的时间片轮转时间应满足要求

第4章　处理机调度与死锁

第三节　调度算法

六、时间片轮转（RR）调度算法

- **RR进程调度举例（q=1）**

进程	到达时间	服务时间	完成时间	周转时间	带权周转时间
A	0	4	15	15	3.75
B	1	3	12	11	3.67
C	2	4	16	14	3.5
D	3	2	9	6	3
E	4	4	17	13	3.25
平均				11.8	3.43

第4章　处理机调度与死锁

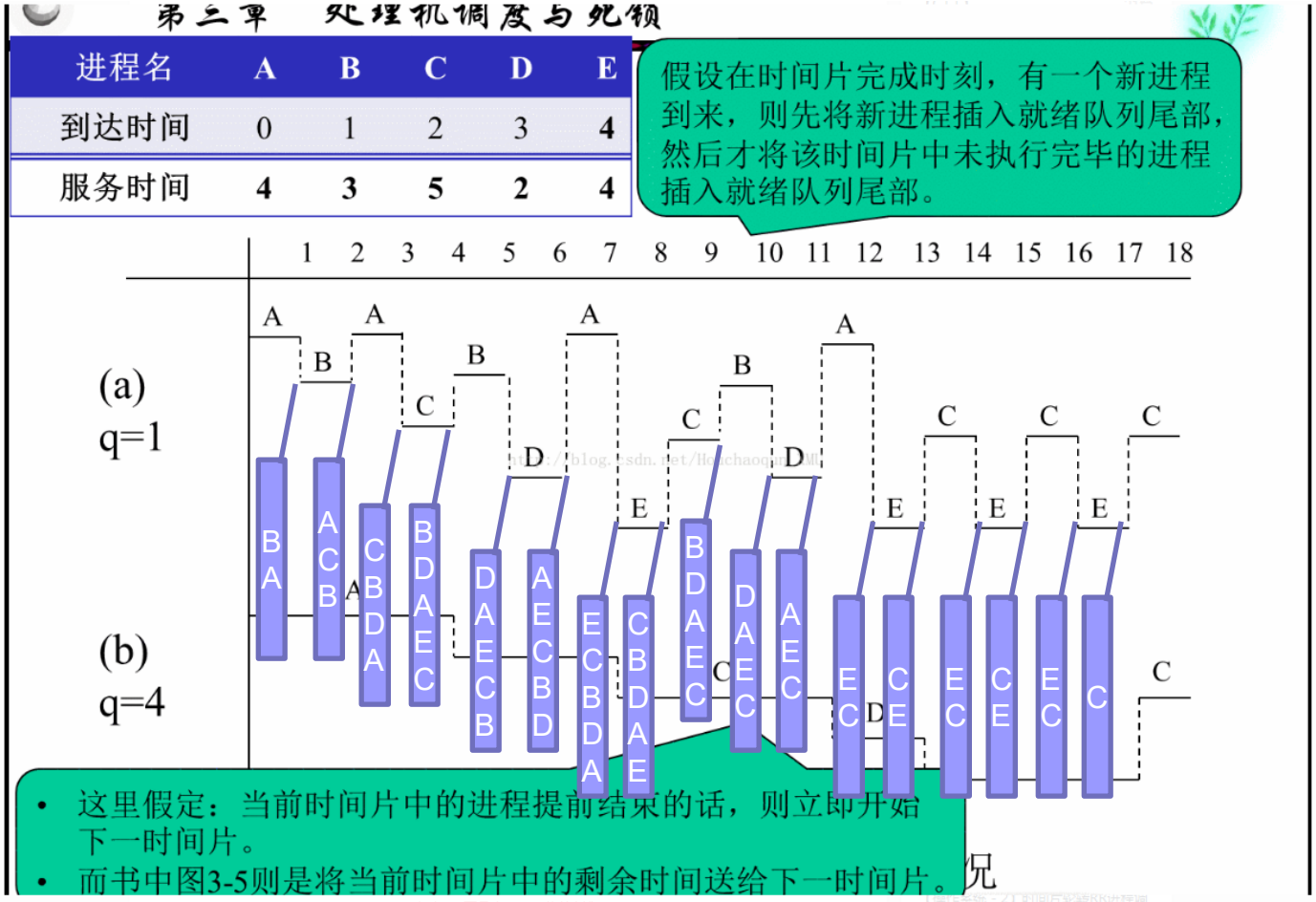
第三节　调度算法

六、时间片轮转（RR）调度算法

- **RR进程调度举例（q=4）**

进程	到达时间	服务时间	完成时间	周转时间	带权周转时间
A	0	4	4	4	1
B	1	3	7	6	2
C	2	4	11	9	2.25
D	3	2	13	10	5
E	4	4	17	13	3.25
平均				8.4	2.7

前面这个例子有没有算错？



https://blog.csdn.net/houchaoqun_xmu/article/details/55540250

深圳大学 Shenzhen University

前面这个例子有没有算错？

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**	
RR q=1	完成时间	12	10	18	11	17	
	周转时间	12	9	16	8	13	11.6
	带权周转时间	3	3	3.2	4	3.25	3.29
RR q=4	完成时间	4	7	18	13	17	
	周转时间	4	6	16	10	13	9.8
	带权周转时间	1	2	3.2	5	3.25	2.89

第4章　处理机调度与死锁

第三节　调度算法

六、时间片轮转（RR）调度算法

- **特性**

  RR用于进程调度，适合于分时系统

  时间片越长，越有利于缩短周转时间

  如果时间片太长，RR退化为FCFS

第4章　处理机调度与死锁

第三节　调度算法

六、时间片轮转（RR）调度算法

- **优点：**

  有利于交互性、事务性进程

  有利于I/O繁忙型的进程

- **缺点：**

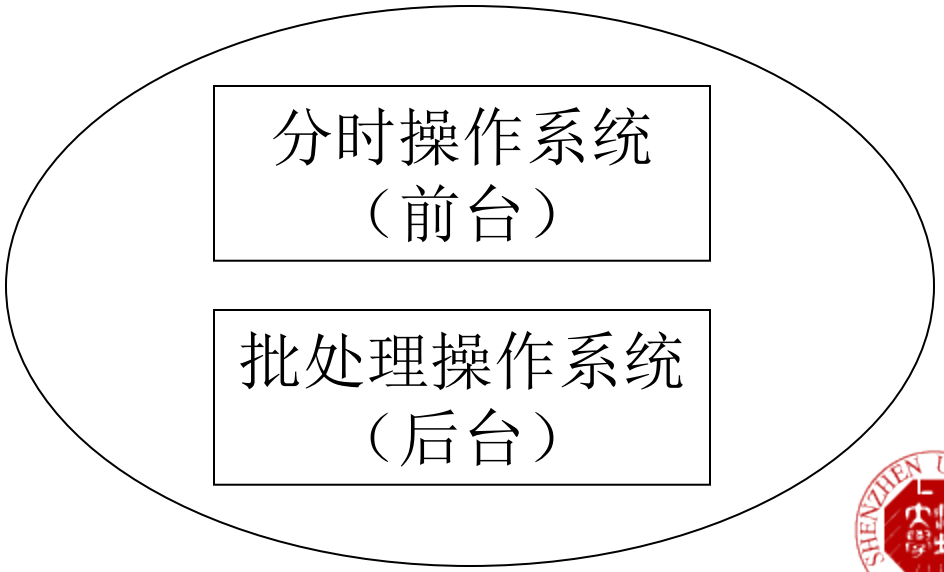
  调度开销较大，未考虑实时响应要求

第4章　处理机调度与死锁

第三节　调度算法

七、多级队列调度算法

- **为了提高计算机系统的性能，一个系统中可能同时配置几种操作系统**



第4章　处理机调度与死锁

第三节　调度算法

七、多级队列调度算法

- 将就绪队列分成多种不同队列（前台轮转就绪队列、后台FCFS就绪队列）
- 每个进程固定地分属于一个队列
- 不同队列采用不同的调度算法（前台就绪队列采用RR调度算法，后台就绪队列采用FCFS算法）
- 只有当前台就绪队列空时，才执行后台队列中的进程

第4章　处理机调度与死锁

第三节　调度算法

七、多级队列调度算法

- **特性**
  同一计算机系统存在多个OS

- **优点：**
  可以同时兼顾到分时及批量处理任务

- **缺点：**
  未考虑紧迫性作业或进程

第4章　处理机调度与死锁

第三节　调度算法

八、多级反馈队列调度算法

- 设置多个就绪队列，并从高到低赋予不同的优先级
- 每个队列采用RR算法，时间片长度从高优先级到低优先级依次增加（一般加倍）（S1<S2<…<Sn）

第4章　处理机调度与死锁

第三节　调度算法

八、多级反馈队列调度算法

- 新进程放到最高优先级就绪队列尾部
- 如果进程在当前就绪队列中执行一次未完成，则插入低一级就绪队列尾部
- 较高优先级就绪队列全部为空时，才执行当前队列进程
- 如果有更高优先级进程，则将当前进程放在当前就绪队列尾部，转而执行优先级高的进程

第4章　处理机调度与死锁

第三节　调度算法

七、多级反馈队列调度算法

- **举例**

有一系统，采用三级反馈队列调度算法，时间片大小分别为：4，8，16，现有三个进程，到达时刻分别为0，2，9，执行时间分别为6，8，10，求每个进程的周转时间。

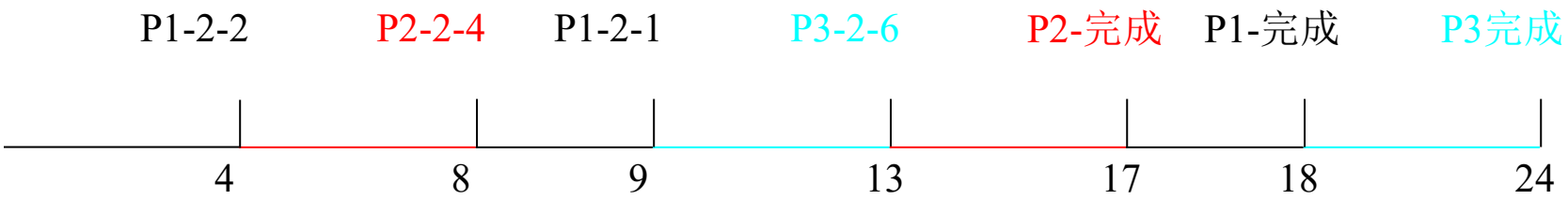
进程	到达时刻	运行时间
P1	0	6
P2	2	8
P3	9	10

第4章　处理机调度与死锁

第三节　调度算法

七、多级反馈队列调度算法

- 举例：有一系统，采用三级反馈队列调度算法，时间片大小分别为：4，8，16，现有三个进程，到达时刻分别为0，2，9，执行时间分别为6，8，10，求每个进程的周转时间。
- Pi-n-m　　i:进程号　　n:就绪队列号　　m:剩余时间



进程	到达时刻	运行时间	周转时间
P1	0	6	18
P2	2	8	17
P3	9	10	15

第4章　处理机调度与死锁

第三节　调度算法

七、多级反馈队列调度算法

- 特性

  是一种比较好的进程调度算法，能满足各种类型用户的需要

(1)、终端型用户：交互性作业小

(2)、短时作业用户：执行时间短

(3)、长时作业用户：在第n个队列中按RR算法轮转执行

第4章　处理机调度与死锁

第三节　调度算法

七、多级反馈队列调度算法

- **优点：**
  可以同时兼顾到实时、分时及批量处理作业（进程）

- **缺点：**
  调度算法比较复杂，调度开销较大

第4章　处理机调度与死锁

第四节　实时调度

一、实时任务特点

- **紧迫性**

  必须在截止时间之前开始或结束

- **截止时间：最迟时间**

  开始截止时间，完成截止时间

第4章　处理机调度与死锁

第四节　实时调度

二、实现实时系统调度的基本条件

1、提供必要的调度信息（p97）

- **就绪时间**
- **开始截止时间/完成截止时间**
- **处理时间**
- **资源要求**
- **优先级**

第4章　处理机调度与死锁

第四节　实时调度

二、实现实时系统调度的基本条件

2、系统处理能力强

- **设系统中有m个周期性的硬实时任务，它们的处理时间为Ci，周期时间为Pi，则：**

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

- **提高系统处理能力的途径（p98）**

第4章　处理机调度与死锁

第四节　实时调度

二、实现实时系统调度的基本条件

3、广泛采用抢占式调度机制

- **优先级高的任务必须优先投入运行**

4、具有快速切换机制

- **对外部中断的快速响应能力**
  - **禁止中断的时间间隔尽量短**
- **快速的任务分派能力**

第4章　处理机调度与死锁

第四节　实时调度

三、实时系统调度方式

- 一般采用抢占方式
- 如果采用非抢占方式，则所有的实时任务（进程）执行时间都必须很短，并能在执行完最关键的区域和临界区后，自动阻塞

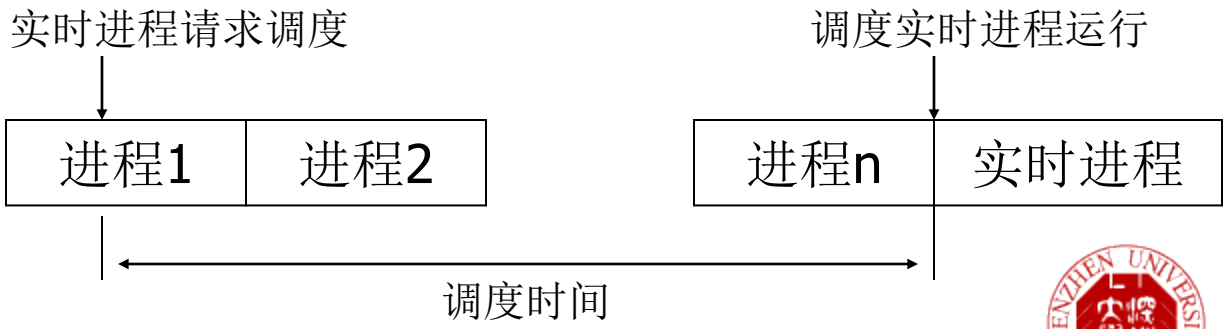
第4章　处理机调度与死锁

第四节　实时调度

四、实时调度算法

1、时间片轮转调度算法

- 可以满足响应时间在数秒的实时控制
- 可应用于一般实时系统



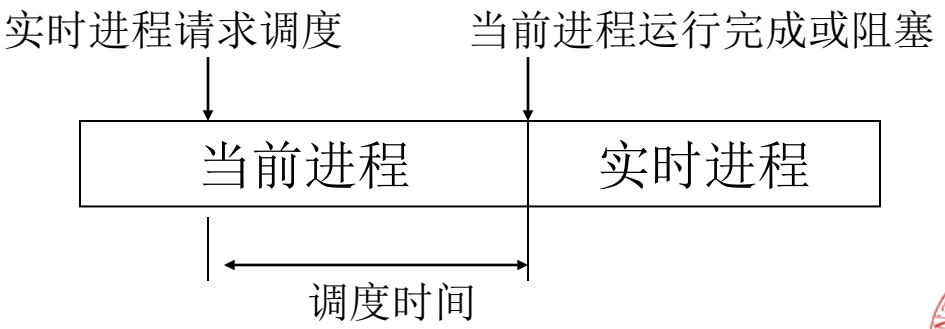
第4章　处理机调度与死锁

第四节　实时调度

四、实时调度算法

2、非抢占优先权调度算法

- 可以满足响应时间在数百毫秒的实时控制
- 可用于要求不太严格的实时控制系统



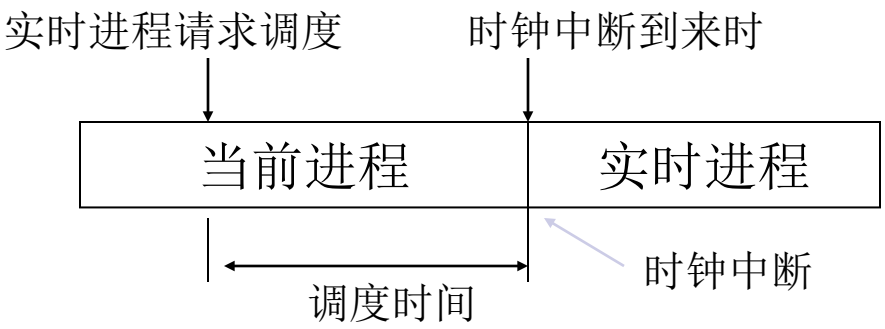
第4章　处理机调度与死锁

第四节　实时调度

四、实时调度算法

3、基于时间中断的抢占优先权调度算法

- 可以满足响应时间在几毫秒到几十毫秒的实时控制
- 可用于大多数实时控制系统



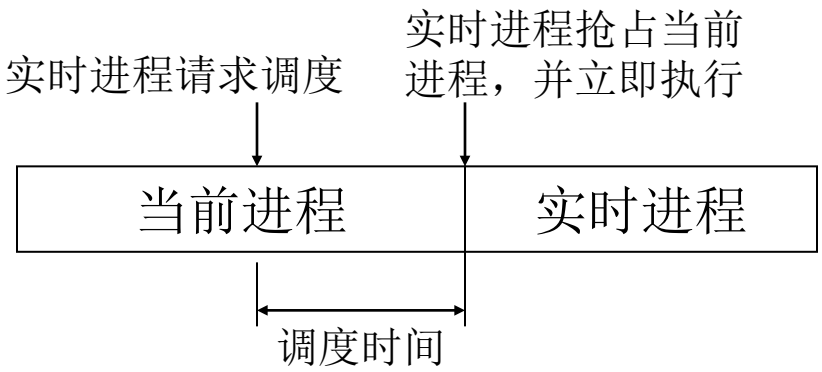
第4章　处理机调度与死锁

第四节　实时调度

四、实时调度算法

4、立即抢占的优先权调度算法

- 可以满足响应时间在**毫微秒到几毫秒**的实时控制
- 基本可满足**所有实时控制系统**要求



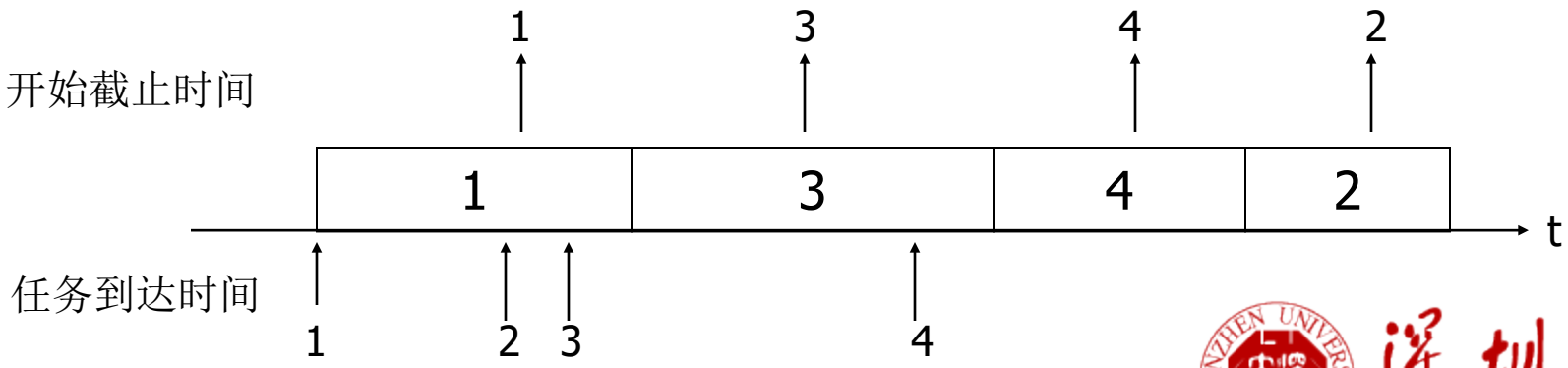
第4章　处理机调度与死锁

第四节　实时调度

四、实时调度算法

5、最早开始截止时间优先(EDF)算法（p99）

- 根据任务的开始截止时间确定任务的优先级
- 开始截止时间越早，优先级越高



第4章　处理机调度与死锁

第四节　实时调度

四、实时调度算法

6、最低松驰度优先(LLF)算法

- 根据任务的紧急(松驰)程度确定任务的优先级

松驰度＝完成时间－剩余处理时间－当前时间

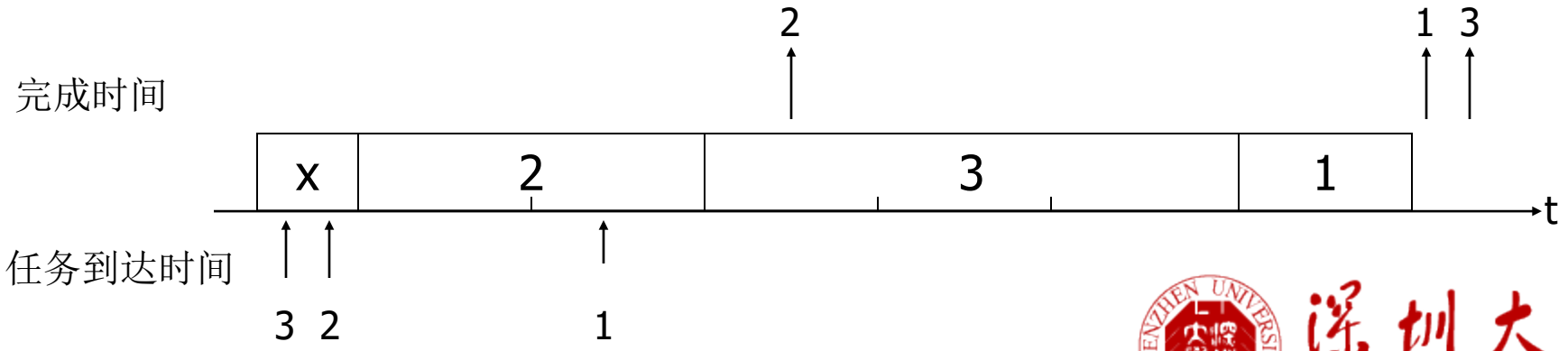
第4章　处理机调度与死锁

第四节　实时调度

四、实时调度算法

6、最低松驰度优先(LLF)算法　（p101　例子）

- 举例：任务1处理时间为10，任务2处理时间20，任务3处理时间30



第4章　处理机调度与死锁

第五节　死锁

一、产生死锁的原因

- **死锁**

1、指多个进程因竞争资源而造成的一种僵局

2、若无外力作用，这些进程都将永远不能再向前进

第4章　处理机调度与死锁

第五节　死锁

一、产生死锁的原因

- **竞争资源**
  当两个或以上进程需要两个或以上资源

- **进程推进顺序非法**
  请求和释放资源的顺序不当

第4章　处理机调度与死锁

第五节　死锁

一、产生死锁的原因

- **竞争资源**

1、**可剥夺性资源**：某进程获得该类资源后，可被其他进程或系统录夺（如CPU，RAM等）

2、**非剥夺性资源**：某进程获得该类资源后，其他进程或系统不可剥夺，只能在进程用完后自行释放（如打印机等）

第4章　处理机调度与死锁

第五节　死锁

一、产生死锁的原因

- **竞争资源**

3、临时性资源：由某进程产生，由另一进程临时使用的资源（如信号量）

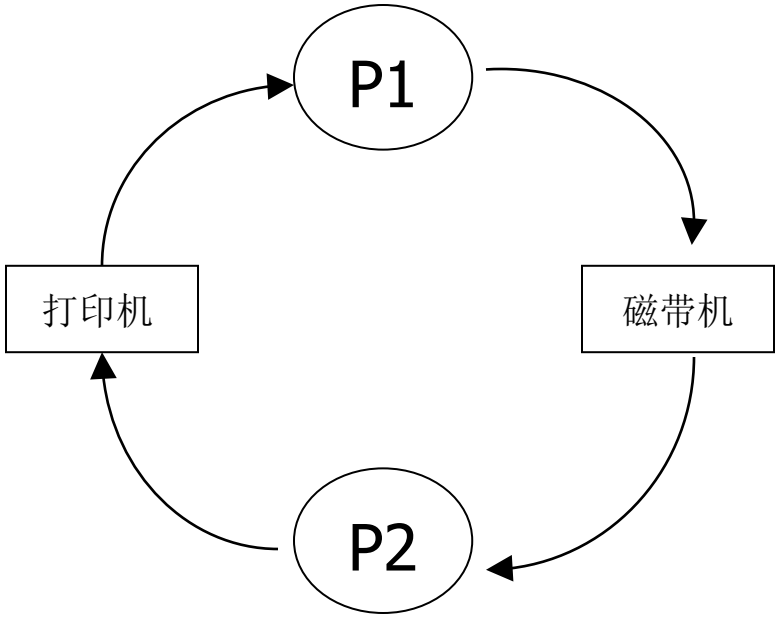
当两个或以上进程需要两个或以上非剥夺资源

第4章　处理机调度与死锁

第五节　死锁

一、产生死锁的原因

- **竞争非剥夺性资源**



第4章　处理机调度与死锁

第五节　死锁

一、产生死锁的原因

- **竞争临时性资源**

举例(哲学学就餐问题)：

```
var chopstick: array[0,…,4]: semaphore := 1, 1, 1, 1, 1;        //定义变量

 procedure OddProcess;                //哲学家就餐进程
 begin
     wait(chopstick[i]);
     wait(chopstick[i+1]);
     吃饭……
     signal(chopstick[i]);
     signal(chopstick[i+1]);
 end;
```

第4章　处理机调度与死锁

第五节　死锁

一、产生死锁的原因

- **进程推进顺序非法**

  请求和释放资源的顺序不当

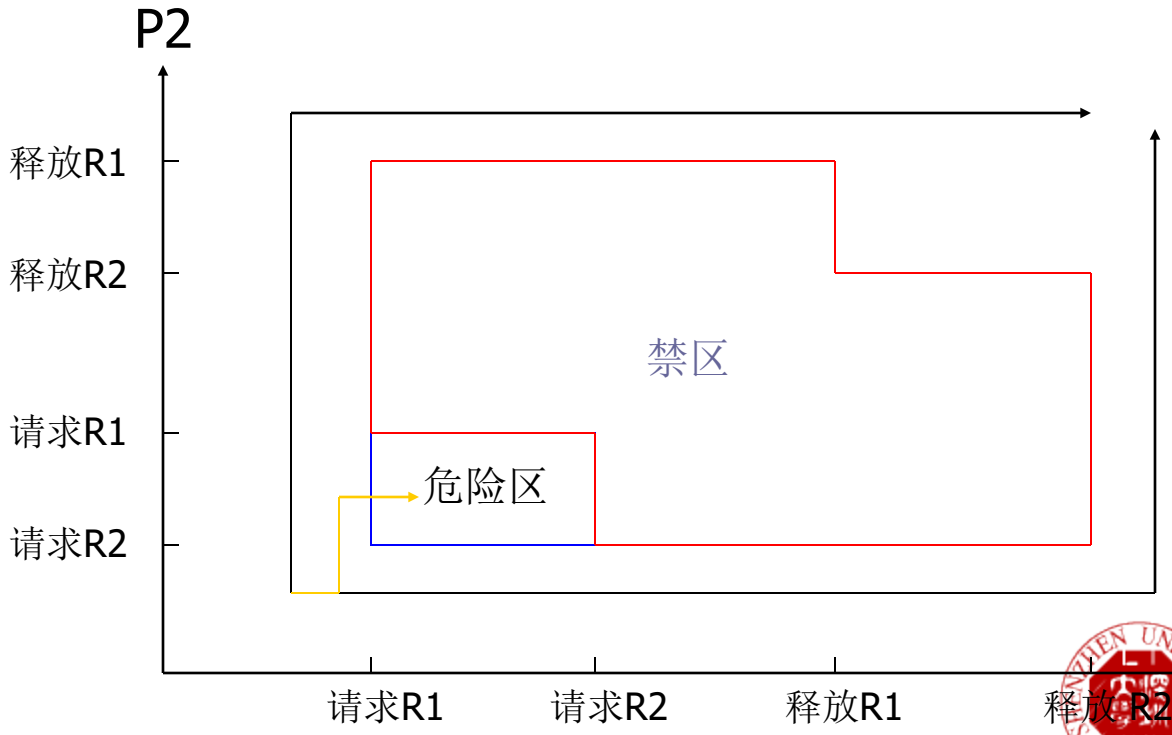
第4章　处理机调度与死锁

第五节　死锁

一、产生死锁的原因

- **进程推进顺序举例**

只能从危险区进入禁区
其它方向不会进入禁区



第4章　处理机调度与死锁

第五节　死锁

二、产生死锁的必要条件

1、**互斥条件**：请求的资源为临界资源

2、**请求和保持条件**：申请新资源，保持旧资源

3、**不剥夺条件**：已获得的资源，在使用完之前，不被外力剥夺

4、**环路等待条件**：互相等待资源

第4章　处理机调度与死锁

第五节　死锁

三、处理死锁的基本方法

1、预防死锁

- 设置某些限制条件，破坏产生死锁的必要条件中的一个或几个条件

(1). 一次将资源全部分配（摒弃“请求和保持”条件）

(2). 当请求的资源得不到满足时，释放已分配的资源（摒弃“不剥夺”条件）

(3). 对资源的申请必须按一定顺序进行（摒弃“环路等待”条件）

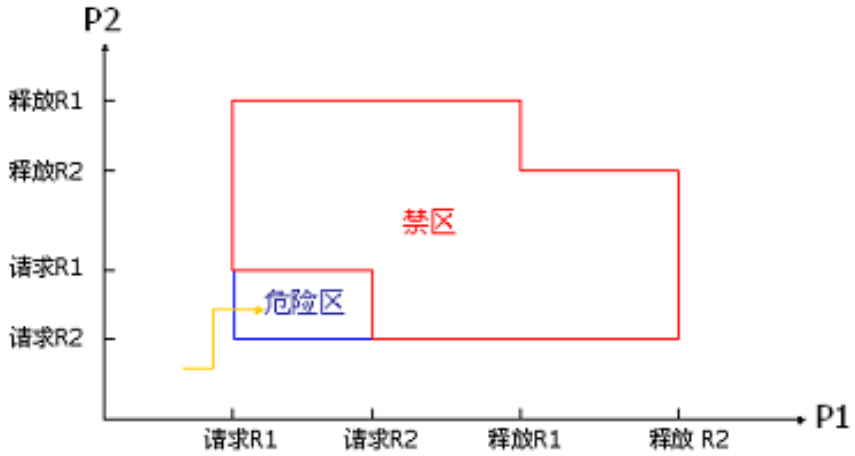
第4章　处理机调度与死锁

第五节　死锁

三、处理死锁的基本方法

2、避免死锁

- 在资源的动态分配过程中，用某种方法，去防止系统进入不安全状态



第4章　处理机调度与死锁

第五节　死锁

三、处理死锁的基本方法

3、检测死锁

- 通过检测机构，及时地检测出死锁的发生及原因，并确定有关的进程和资源
- 采取措施，解除死锁

第4章　处理机调度与死锁

第五节　死锁

三、处理死锁的基本方法

4、解除死锁

- 剥夺资源或通过撤消进程，收回一些资源。
- 解除死锁与检测死锁是相互合作的关系。

第4章　处理机调度与死锁

第五节　死锁

四、死锁的预防

1、摒弃“请求和保持”条件

一次将资源全部分配

优点：简单，易于实现且安全

缺点：资源浪费严重

进程延迟运行

第4章　处理机调度与死锁

第五节　死锁

四、死锁的预防

2、摒弃“不剥夺”条件

当请求的资源得不到满足时，释放已分配的资源

缺点：前期工作全部作废，代价高

反复申请，性能下降

第4章　处理机调度与死锁

第五节　死锁

四、死锁的预防

3、摒弃“环路等待”条件

对资源的申请必须按一定顺序进行

缺点：资源序号需要相对稳定

先获得的资源有可能长期闲置

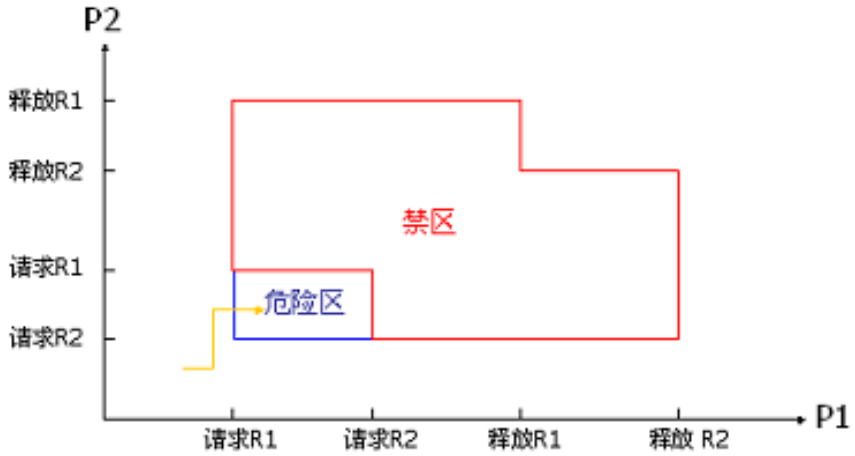
对用户编程(资源申请时)有限制

第4章　处理机调度与死锁

第五节　死锁

五、死锁的避免

- 分配资源之前，先判断是否会进入不安全区（危险区和禁区）
- 如果会进入危险区，则不分配资源



第4章　处理机调度与死锁

第五节　死锁

五、死锁的避免

- Dijkstra银行家算法：

(1). 进程需求资源数如果大于现有资源数，不分配

(2). 预分配，检查是否存在一个进程序列，可以安全执行完成。存在则分配，否则不分配。

第4章　处理机调度与死锁

第五节　死锁

五、死锁的避免

- 银行家算法数据结构：

1. Available[m]：可利用资源向量，表示系统中可被利用的各类(共m类)资源的数目

2. Max[n,m]：最大需求矩阵，表示每个进程(共n个进程)需要各类资源的最大数目

第4章　处理机调度与死锁

第五节　死锁

五、死锁的避免

- **银行家算法数据结构：**

3. Allocation[n,m]：分配矩阵，表示系统为每个进程已分配各类资源的数目

4. Need[n,m]：需求矩阵，表示每一个进程尚需要的各类资源数目

Need[i,j] = Max[i,j] – Allocation[i,j]

第4章　处理机调度与死锁

第五节　死锁

五、死锁的避免

- **银行家算法中资源分配表　（p113）：**

资源情况 / 进程	Max			Allocation			Need			Available		
	A	B	C	A	B	C	A	B	C	A	B	C
P0	7	5	3	0	1	0	7	4	3	3	3	2
P1	3	2	2	2	0	0	1	2	2			
P2	9	0	2	3	0	2	6	0	0			
P3	2	2	2	2	1	1	0	1	1			
P4	4	3	3	0	0	2	4	3	1			

第4章　处理机调度与死锁

第五节　死锁

五、死锁的避免

- **银行家算法数据结构：**

5. Request[m]：请求向量，表示进程请求分配各类资源的数目

第4章　处理机调度与死锁

第五节　死锁

五、死锁的避免

- 银行家算法：

1. 如果Request[j] > Need[i, j], 出错
2. 如果Request[j] > Available[j]，资源数目不够分配，进入等待状态
3. 执行安全性检查算法，如果系统处理安全状态（有一个安全进程执行序列），则分配资源；否则，不分配，进程进入等待状态

第4章　处理机调度与死锁

第五节　死锁

五、死锁的避免

- 银行家算法中安全性算法数据结构：

1、Work[m]：工作向量，表示系统可提供进程继续运行所需要的各类资源数目，其初值为Available[m]

2、Finish[n]：完成向量，表示系统是否有足够的资源分配给进程（true/false）

第4章　处理机调度与死锁

第五节　死锁

五、死锁的避免

- 银行家算法中安全性算法：

1、从进程集合中找到一个进程Pi，满足：
Finish[i] = false，Need[i,j] <= Work[j]

2、Work[j] = Work[j]+Allocation[i,j]
Finish[i] = true

3、重复1,2步骤

4、如果所有进程的Finish[i]=true，则存在一个安全执行序列；否则，系统处于不安全状态

第4章　处理机调度与死锁

第五节　死锁

五、死锁的避免

- **银行家算法中查找安全进程序列举例：**

资源 / 进程	Work			Need			Allocation			Work+Allocation			完成
	A	B	C	A	B	C	A	B	C	A	B	C	
P1	3	3	2	1	2	2	2	0	0	5	3	2	true
P3	5	3	2	0	1	1	2	1	1	7	4	3	true
P4	7	4	3	4	3	1	0	0	2	7	4	5	true
P2	7	4	5	6	0	0	3	0	2	10	4	7	true
P0	10	4	7	7	4	3	0	1	0	10	5	7	true

安全进程执行序列P1→ P3→ P4→ P2→ P0

第4章　处理机调度与死锁

第五节　死锁

五、死锁的避免

- **银行家算法中查找安全进程序列举例：**
  进程P1请求分配资源Request(1, 0, 2)

资源情况 / 进程	Max			Allocation			Need			Available		
	A	B	C	A	B	C	A	B	C	A	B	C
P0	7	5	3	0	1	0	7	4	3	2	3	0
P1	3	2	2	3	0	2	0	2	0			
P2	9	0	2	3	0	2	6	0	0			
P3	2	2	2	2	1	1	0	1	1			
P4	4	3	3	0	0	2	4	3	1			

第4章　处理机调度与死锁

第五节　死锁

五、死锁的避免

- **银行家算法中查找安全进程序列举例：**
  进程P1请求分配资源Request(1, 0, 2)后：

资源 / 进程	Work			Need			Allocation			Work+Allocation			完成
	A	B	C	A	B	C	A	B	C	A	B	C	
P1	2	3	0	0	2	0	3	0	2	5	3	2	true
P3	5	3	2	0	1	1	2	1	1	7	4	3	true
P4	7	4	3	4	3	1	0	0	2	7	4	5	true
P0	7	4	5	7	4	3	0	1	0	7	5	5	true
P2	7	5	5	6	0	0	3	0	2	10	5	7	true

安全进程执行序列P1→ P3→ P4→ P0→ P2

第4章　处理机调度与死锁

第五节　死锁

五、死锁的避免

- **银行家算法中查找安全进程序列举例：**
  进程P1分配到资源后，P0请求分配资源Request(0, 2, 0)

资源情况 / 进程	Max			Allocation			Need			Available		
	A	B	C	A	B	C	A	B	C	A	B	C
P0	7	5	3	0	1	0	7	4	3	2	3	0
P1	3	2	2	3	0	2	0	2	0			
P2	9	0	2	3	0	2	6	0	0			
P3	2	2	2	2	1	1	0	1	1			
P4	4	3	3	0	0	2	4	3	1			

第4章　处理机调度与死锁

第五节　死锁

五、死锁的避免

- **银行家算法中查找安全进程序列举例：**
  P0请求分配资源Request(0, 2, 0)后，预分配

资源情况 / 进程	Max			Allocation			Need			Available		
	A	B	C	A	B	C	A	B	C	A	B	C
P0	7	5	3	0	3	0	7	2	3	2	1	0
P1	3	2	2	3	0	2	0	2	0			
P2	9	0	2	3	0	2	6	0	0			
P3	2	2	2	2	1	1	0	1	1			
P4	4	3	3	0	0	2	4	3	1			

第4章　处理机调度与死锁

第五节　死锁

五、死锁的避免

- **银行家算法中查找安全进程序列举例：**
  进程P0请求分配资源Request(0, 2, 0)后，执行安全性检查：

资源 / 进程	Work			Need			Allocation			Work+Allocation			完成
	A	B	C	A	B	C	A	B	C	A	B	C	
P0	2	1	0	7	2	3	0	3	0				false
P1				0	2	0	3	0	2				false
P2				6	0	0	3	0	2				false
P3				0	1	1	2	1	1				false
P4				4	3	1	0	0	2				false

不存在安全进程执行序列，预分配无效，P0等待

第4章　处理机调度与死锁

第五节　死锁

五、死锁的避免

- Dijkstra银行家算法的缺点：

1. 很少有进程能够在运行前就知道其所需资源的最大值
2. 而且进程数也不是固定的，往往在不断地变化（如新用户登录或退出）
3. 原本可用的资源也可能突然间变成不可用（如磁带机可能坏掉）
4. 银行家算法的开销较大，实时性不是很好

第4章　处理机调度与死锁

第五节　死锁

六、死锁的检测

- **死锁检测要求：**

1、保存有关资源的请求和分配信息

2、提供一种算法，以利用这些信息来检测系统是否已进入死锁状态

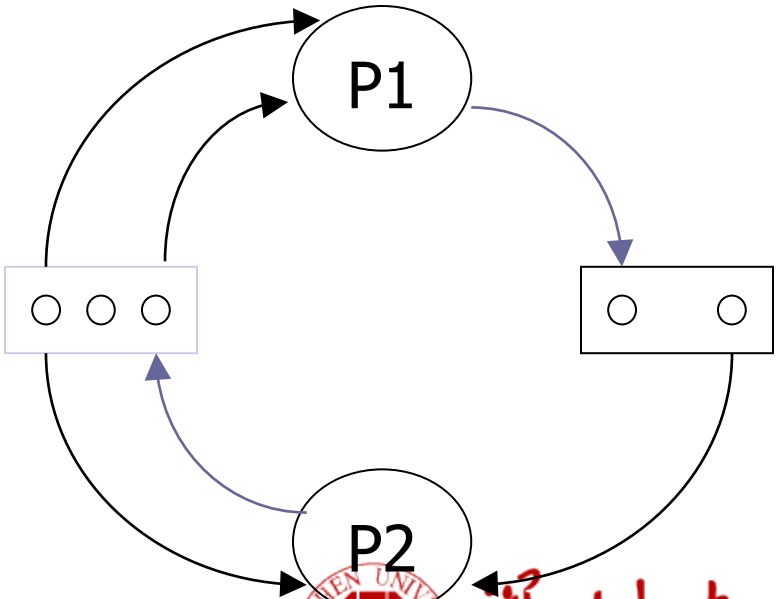
第4章　处理机调度与死锁

第五节　死锁

六、死锁的检测

- **资源分配图**

1. **进程：圆圈**
2. **资源：方框**
3. **请求边：由进程指向资源**
4. **分配边：由资源指向进程**



第4章　处理机调度与死锁

第五节　死锁

六、死锁的检测

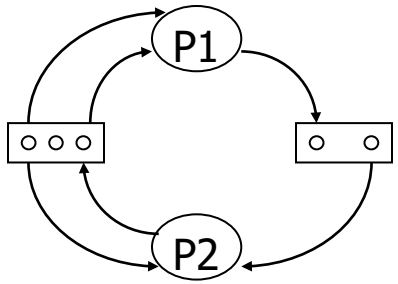
- 死锁定理

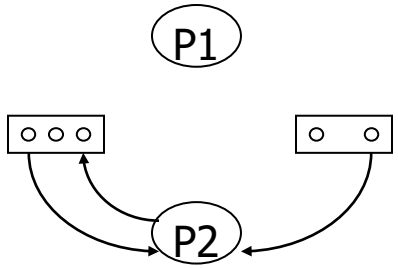
(1). 在资源分配图中，找到一个可获得所有资源的进程结点Pi，使之变成孤立点（释放资源）

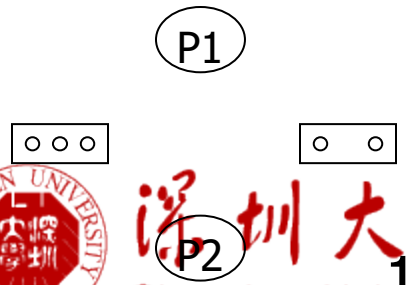
(2). 再找另一个可获得所有资源的进程结点Pj，使之变成孤立点

(3). 重复(2)

(4). 如果所有进程都变成孤立点，则没有死锁；否则，存在死锁







第4章　处理机调度与死锁

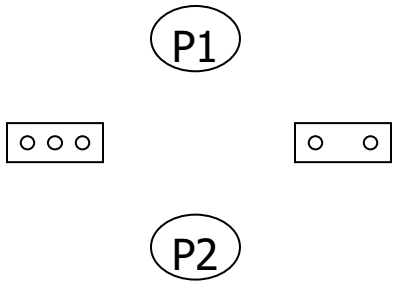
第五节　死锁

六、死锁的检测

- **死锁定理**

系统或任务处于死锁状态的充分必要条件是：

当且仅当系统或任务的资源分配图是不可完全简化的



第4章　处理机调度与死锁

第五节　死锁

七、死锁的解除

- 剥夺资源
- 撤消进程

  全部撤消法

  最小代价撤消法（p117）