西安交通大学
软件学院

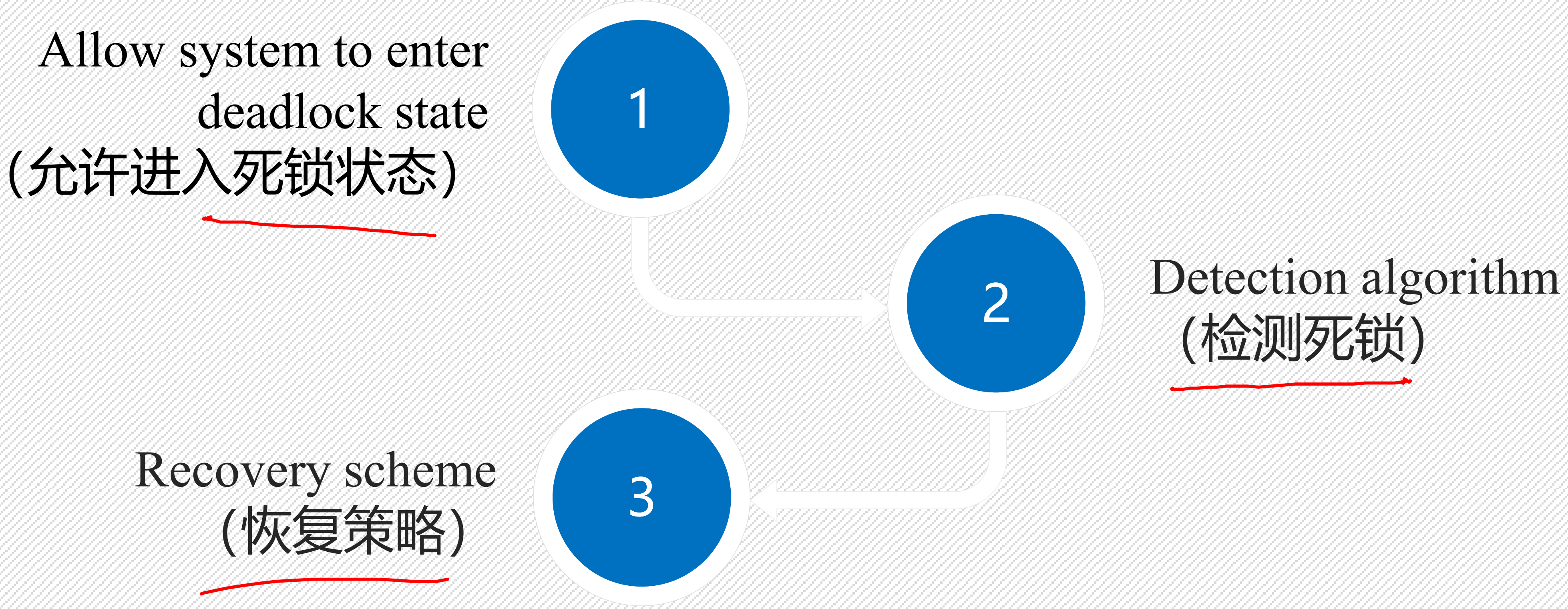
操作系统原理

Operating System Principle

田丽华

7-6　死锁检测和恢复

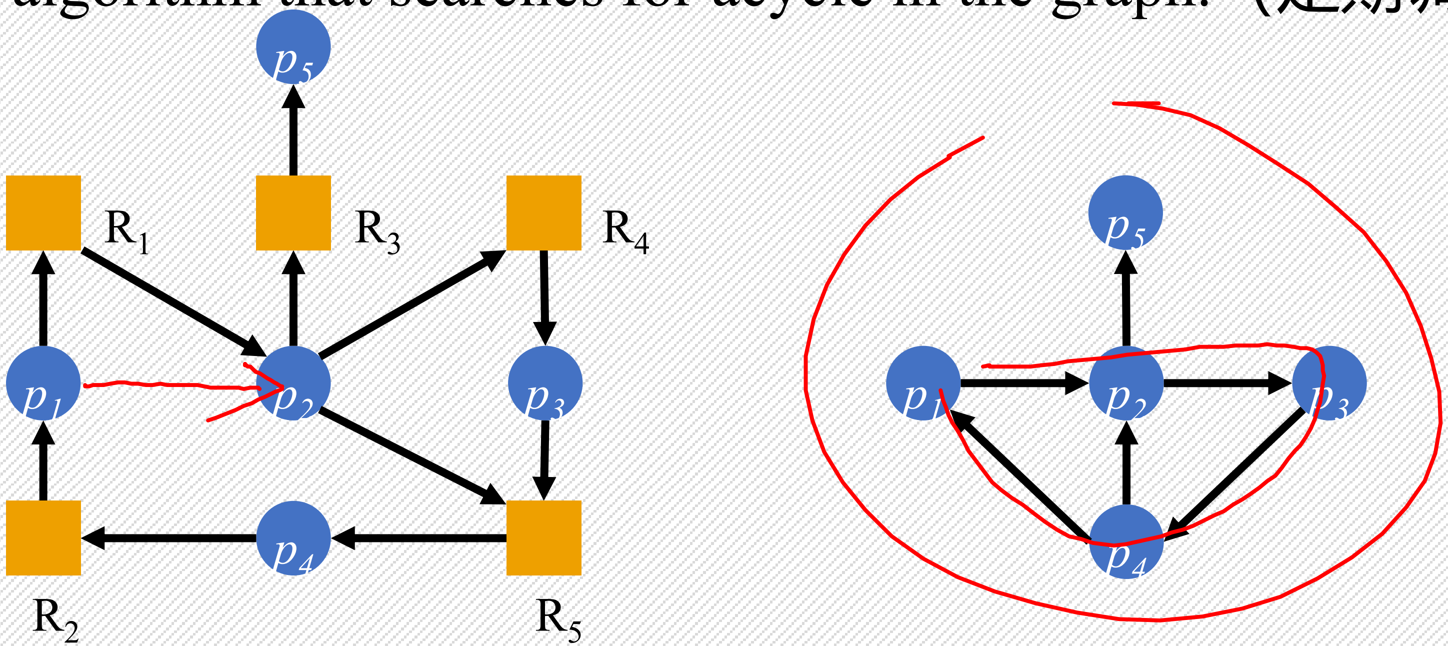
Deadlock Detection
死锁检测



Single Instance of Each Resource Type

每一种资源类型只有一个实例

- Maintain wait-for graph（维护等待图）
  - Nodes are processes.（节点是进程）
  - $P_i \rightarrow P_j$ if Pi is waiting for P_j.（$P_i \rightarrow P_j$表明P_i在等待P_j.）
- Periodically invoke an algorithm that searches for acycle in the graph.（定期调用算法来检查是否有环）



(a)
Resource-Allocation Graph

(b)
Corresponding wait-for graph

Several Instances of a Resource Type

一个资源类型的多个实例

01

Available: A vector of length *m* indicates the number of available resources of each type.

（可用：一个长度m的向量代表每种资源类型的有效数目）

02

Allocation: An *n x m* matrix defines the number of resources of each type currently allocated to each process.

（分配：一个*n x m* 的矩阵定义了当前分配的每一种资源类型的实例数目）

03

Request: An *n x m* matrix indicates the current request of each process. If *Request [i,j]* = *k*, then process P_i is requesting *k* more instances of resource type. R_j.

（请求：一个*n x m* 的矩阵申明了当前的进程请求。如果Request[i,j]=k, 那么进程Pi请求k个Rj资源的实例）

Detection Algorithm

检测算法

01 Let *Work and Finish* be vectors of length m and n, respectively Initialize(让Work和Finish作为长度为m和n的向量)

(a) $Work := Available$

(b) For i = 1,2, …, n, if $Allocation_i \neq 0$, then $Finish[i] :=$ false;otherwise, $Finish[i] := true$.

02 Find an index i such that both（找到下标i）

(a) $Finish[i] = false$

(b) $Request_i \leq Work$

If no such i exists, go to step 4. （如果没有这样的i存在，转4）

Detection Algorithm (Cont.)

03

$Work := Work + Allocation_i$

$Finish$[i] := $true$

go to step 2.

04

If $Finish[i]$ = false, for some i, $1 \leq i \leq n$, then the system is in deadlock state. Moreover, if $Finish[i] = false$, then P_i is deadlocked.

Example of Detection Algorithm

检测算法的例子

- Five processes P_0 through P_4; three resource types

  A (7 instances), B (2 instances), and C (6 instances).

（五个进程p0到p4,三个资源类型A（7个实例），B（2个实例）,C（6个实例）

- Snapshot at time T_0（时刻Tn的状态）

	Allocation			***Request***			***Available***		
	A	B	C	A	B	C	A	B	C
P_0	0	1	0	0	0	0	0	0	0
P_1	2	0	0	2	0	2			
P_2	3	0	3	0	0	0			
P_3	2	1	1	1	0	0			
P_3	0	0	2	0	0	2			

- Sequence $<P_0, P_2, P_3, P_1, P_4>$ *will result in Finish[i] = true for all i.*

Example (Cont.)
例子续

P_2 **requests an additional instance of type C.（P2请求C的实例）**

0 1 0

	Request		
	A	B	C
P_0	0	0	0
P_1	2	0	1
P_2	0	0	1
P_3	1	0	0
P_4	0	0	2

State of system?（系统的状态）

- Can reclaim resources held by process P_0, but insufficient resources to fulfill other processes; requests.
  （可以归还P_0所有的资源，但是资源不够完成其他进程的请求）
- Deadlock exists, consisting of processes P_1, P_2, P_3, and P_4.
  （死锁存在，包括进程P_1,P_2,P_3和P_4）

Recovery from Deadlock

死锁发生后，如何处理死锁?

- 操作员人工处理
- 进程终止
- 资源抢占

Recovery from Deadlock: Process Termination
从死锁中恢复：进程终止

01 Abort all deadlocked processes.（终止所有的死锁进程）

02 Abort one process at a time until the deadlock cycle is eliminated.（一次终止一个进程直到死锁环消失）

03 In which order should we choose to abort?（选择终止顺序）

- Priority of the process.（进程的优先级）
- How long process has computed, and how much longer to completion.（进程计算了多少时间，还需要多长时间）

Recovery from Deadlock: Resource Preemption

从死锁中恢复：资源抢占

逐步从进程中抢占资源，直到打破死锁

- Selecting a victim – minimize cost.
  (选择一个牺牲品：最小化代价)
- Rollback – return to some safe state, restart process fro that state.
  (回退：返回到安全的状态，然后重新开始进程)
- Starvation – same process may always be picked as victim, include number of rollback in cost factor.
  (饥饿：同一个进程可能总是被选中，包括在回退时)