



南京大學  
NANJING UNIVERSITY

# 计算机操作系统

6 并发程序设计 – 6.6 死锁

6.6.4 死锁的检测

掌握死锁检测

掌握死锁检测的方法

掌握死锁检测后的解决方法

# 死锁的检测

- 解决死锁问题的另一条途径是死锁检测方法
- 这种方法对资源的分配不加限制，但系统定时运行一个“死锁检测”程序，判断系统内是否已出现死锁，若检测到死锁则设法加以解除

# 死锁的检测

- 检测的一种方法：可设置两张表格来记录进程使用资源的情况
- 等待资源表记录每个被阻塞进程等待的资源
- 占用资源表记录每个进程占有的资源
- 进程申请资源时，先查该资源是否为其它进程所占用；若资源空闲，则把该资源分配给申请者且登入占用资源表；否则，则登入进程等待资源表

# 死锁的检测

资源	占用进程
r1	P <sub>1</sub>
r2	P <sub>2</sub>
r3	P <sub>3</sub>
r4	P <sub>4</sub>
r5	P <sub>5</sub>
...	...

进程	等待资源
P <sub>1</sub>	r1
P <sub>2</sub>	r2
P <sub>3</sub>	r3
...	...

# 死锁的检测

- 死锁检测程序定时检测这两张表，若有进程  $P_i$  等待资源  $r_k$ ，且  $r_k$  被进程  $P_j$  占用，则说  $P_i$  和  $P_j$  具有“等待占用关系”，记为  $W(P_i, P_j)$
- 死锁检测程序反复检测这两张表，可以列出所有的“等待占用关系”
- 如果出现  $W(P_i, P_j), W(P_j, P_k), \dots, W(P_m, P_n), W(P_n, P_i)$  时，显然，系统中存在一组循环等待资源的进程： $P_i, P_j, P_k, \dots, P_m, P_n$ ，也就是说出现了死锁

# 死锁检测的数据结构

- 把两张表格中记录的进程使用和等待资源的情况用一个矩阵A来表示

<div>进程 A[b<sub>ij</sub>] 进程</div>	P1	P2	.....	Pn
P1	b11	b12	.....	bn2
P2	b21	b22	.....	bn2
...	...	...	.....	...
Pn	bn1	bn2	.....	bn2

其中  $b_{ij} = \begin{cases} 1 & \text{当 } P_i \text{ 等待被 } P_j \text{ 占用的资源时} \\ 0 & \text{当 } P_i \text{ 与 } P_j \text{ 不存在等待占用关系时} \end{cases}$

# 死锁检测的算法

- 死锁检测程序可用Warshall的传递闭包算法检测是否有死锁发生，即对矩阵A构造传递闭包 $A^*[b_{ij}]$
- $A^*[b_{ij}]$ 中的每个 $b_{ij}$ 是对 $A[b_{ij}]$ 执行如下算法：  
    for  $k:=1$  to  $n$  do  
        for  $i:=1$  to  $n$  do  
            for  $j:=1$  to  $n$  do  
                 $b_{ij}:= b_{ij} \vee (b_{ik} \wedge b_{kj})$

# 死锁检测后的解决办法

- 可以采用重新启动进程执行的办法，恢复工作应包含重新启动一个或全部进程，以及从哪一点开始重新启动
- 全部卷入死锁从头开始启动，但这样的代价是相当大的
- 在进程执行过程中定时设置**校验点**，从校验点开始重执行
- 中止一个卷入死锁的进程，以后重执行