南京大学
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	P1
r2	P2
r3	P3
r4	P4
r5	P5
…	…

进程	等待资源
P1	r1
P2	r2
P3	r3
…	…

死锁的检测

- 死锁检测程序定时检测这两张表，若有进程Pi等待资源rk，且rk被进程Pj占用，则说Pi和Pj具有“等待占用关系”，记为W(Pi, Pj)
- 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(Pi, Pj), W(Pj, Pk), ..., W(Pm, Pn), W(Pn, Pi)时，显然，系统中存在一组循环等待资源的进程：Pi, Pj, Pk, ..., Pm, Pn，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

进程 / A[bij] 进程	P1	P2	……	Pn
P1	b11	b12	……	bn2
P2	b21	b22	……	bn2
…	…	…	……	…
Pn	bn1	bn2	……	bn2

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

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*[bij]
- A*[bij]中的每个bij是对A[bij]执行如下算法：

```
for k:=1 to n do
    for i:=1 to n do
        for j:=1 to do
            bij:= bij ∨ (bik ∧ bkj)
```

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，恢复工作应包含重启动一个或全部进程，以及从哪一点开始重启动
- 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