进程
死锁

进程死锁

本讲内容

1. 进程死锁概念与条件
2. 进程死锁的预防机制
3. 进程死锁的避免机制
4. 进程死锁检测与解决
5. 进程死锁问题的思考

进程死锁概念与条件

1 死锁定义

什么是死锁，死锁是怎么产生的

- 在单道环境下面是不存在死锁问题的，一个进程占据了系统中所有资源，没有死锁问题

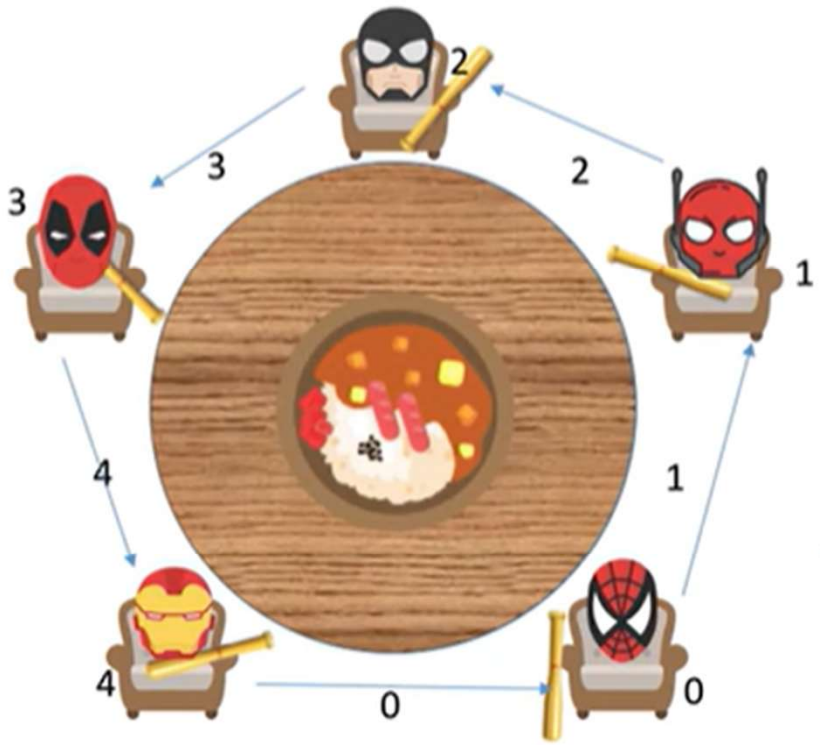
背景：多道进程的并发执行改善系统的资源利用率，但也可能进程相互等待对方释放资源才能继续运行

死锁：系统中多个进程无限期地等待永远不会满足的条件，处于停滞状态，称为进程死锁

- 死锁是在多道程序的并发执行的环境，中存在的问题
- A进程执行，需要等待B进程释放资源，但是如果B进程一直不释放相应的资源，那么A进程就会无限等待

什么是死锁

哲学家进餐问题中，如果5位哲学家进程并发执行，都拿起了左手边的筷子…




```
semaphore chopstick[5]={1,1,1,1,1};
Pi (){          //i号哲学家的进程
    while(1){
        P(chopstick[i]);              //拿左
        P(chopstick[(i+1)%5]);        //拿右
        吃饭…
        V(chopstick[i]);              //放左
        V(chopstick[(i+1)%5]);        //放右
        思考…
    }
}
```

每位哲学家都在等待自己右边的人放下筷子，这些哲学家进程都因等待筷子资源而被阻塞。即发生“死锁”

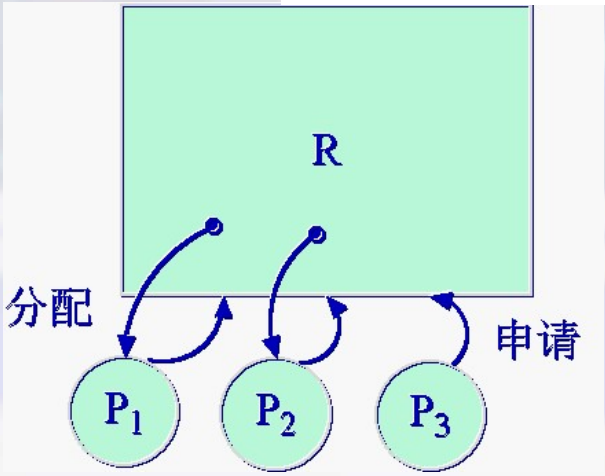
进程死锁概念与条件

2 死锁场景

并发申请同类资源

- 内存资源有*m*个分配单位
- *n*个进程共享内存资源
- 进程每次只能申请一个单位
- 满足总量才能使用
- 每个进程使用完一次

一共有*m* = 2个内存资源R;
P1、P2需要2个内存资源;
P3需要1个内存资源

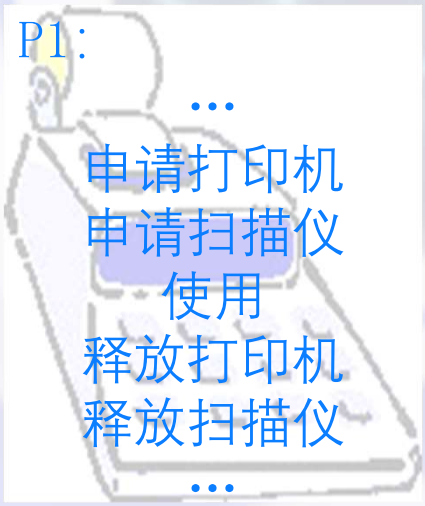


P1获得1个内存资源;
P2获得1个内存资源;
两个内存资源都分配出去了;
P1等着P2释放一个，P2等着P1，P3等着P1/P2;
彼此等待，彼此占用部分资源，都无法同时获资源，造成死锁

进程死锁概念与条件

2 死锁场景

申请不同类资源



P1:

…

申请打印机

申请扫描仪

使用

释放打印机

释放扫描仪

…

P2:

…

申请扫描仪

申请打印机

使用

释放打印机

释放扫描仪

…

并发环境，P1和P2的申请步调，不好控制

P1申请并使用打印机，还没申请扫描仪的时候，P2就进入申请了扫描仪并且获得了扫描仪的使用

这个时候P2再去申请打印机的时候，已经被P1占用，因此P1等着P2释放扫描仪，P2等着P1释放打印机

彼此都拿着部分资源等着对方释放，但又不释放手中已经拥有的资源，无限等待的情况。。。

进程死锁概念与条件

3 死锁条件

必要条件，但不是充分条件，满足条件，也不一定发生死锁

- 互斥使用（资源独占）：资源每次只能给一个进程使用
- 不可强占（不可剥夺）：资源申请者不能强行从占有者手中夺取资源，只能由占有者自愿释放
- 请求保持（部分分配，占有申请）：进程在申请新资源的同时保持对原有资源的占有
- 循环等待：存在进程等待队列 {P1，P2，…，P*n*}，P1等待P2占有的资源，P2等待P3占有的资源，…，P*n*等待P1占有的资源，形成环路

死锁、饥饿、死循环的区别

死锁：各进程互相等待对方手里的资源，导致各进程都阻塞，无法向前推进的现象。
饥饿：由于长期得不到想要的资源，某进程无法向前推进的现象。比如：在短进程优先（SPF）算法中，若有源源不断的短进程到来，则长进程将一直得不到处理机，从而发生长进程“饥饿”。
死循环：某进程执行过程中一直跳不出某个循环的现象。有时是因为程序逻辑 bug 导致的，有时是程序员故意设计的。

	共同点	区别
死锁	都是进程无法顺利向前推进的现象（故意设计的死循环除外）	死锁一定是“循环等待对方手里的资源”导致的，因此如果有死锁现象，那至少有两个或两个以上的进程同时发生死锁。另外，发生死锁的进程一定处于阻塞态。
饥饿		可能只有一个进程发生饥饿。发生饥饿的进程既可能是阻塞态(如长期得不到需要的I/O设备)，也可能是就绪态(长期得不到处理机)
死循环		可能只有一个进程发生死循环。死循环的进程可以上处理机运行（可以是运行态），只不过无法像期待的那样顺利推进。死锁和饥饿问题是由于操作系统分配资源的策略不合理导致的，而死循环是由代码逻辑的错误导致的。死锁和饥饿是管理者（操作系统）的问题，死循环是被管理者的问题。

进程死锁

本讲内容

1. 进程死锁概念与条件
2. 进程死锁的预防机制
3. 进程死锁的避免机制
4. 进程死锁检测与解决
5. 进程死锁问题的思考

进程死锁的预防机制

1 机制原理

- 预先确定资源分配，保证不发生死锁
- 通过破坏死锁4个必要条件之一来实现
- 破坏“互斥使用”这一必要条件不现实

就是发生死锁一定满足必要条件

并不是所有的资源都可以改造成可共享使用的资源。并且为了系统安全，很多地方还必须保护这种互斥性。因此，很多时候都无法破坏互斥条件。

进程死锁的预防机制

2 解决方案

破坏“不可剥夺”

- 允许进程动态申请资源
- 进程在申请新资源不能得到满足而变为等待状态之前，必须释放已占有的资源
- 若需要资源必须重新申请

意味着只要暂时得不到某个资源，之前获得的那些资源就都需要放弃，以后再重新申请，会导致进程饥饿。反复地申请和释放资源会增加系统开销，降低系统吞吐量。

进程死锁的预防机制

2 解决方案

破坏“请求保持”

采用静态分配方法

- 不允许进程动态申请资源
- 进程运行前须一次性申请所需的所有资源
- 进程所要资源均可满足时给予一次性分配
- 执行过程中不会出现等待资源的情况

缺点：一开始执行时，并不知道需要哪些，申请而不使用，降低资源利用率

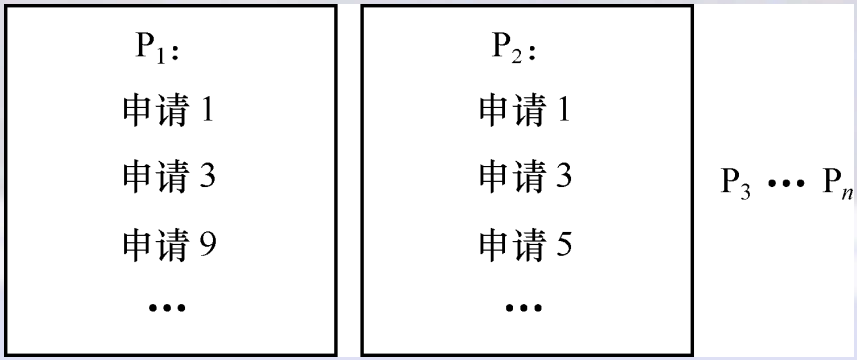
进程死锁的预防机制

2 解决方案

每个进程都是先申请小号资源，再申请大号资源，
P1申请并获得1号资源后，P2再申请1号资源就需要等待
P2就不会再继续申请大号资源，破坏循环彼此等待

破坏“循环等待”

- 采用资源有序分配法
- 系统中所有资源编号
- 进程须严格按资源编号的递增次序申请资源
- 违反上述规则操作系统不予分配



如果有一个进程，就是需要先申请使用大号资源，那它也必须先申请小号资源，使得该资源很早被持有，但是很晚才使用

进程死锁

本讲内容

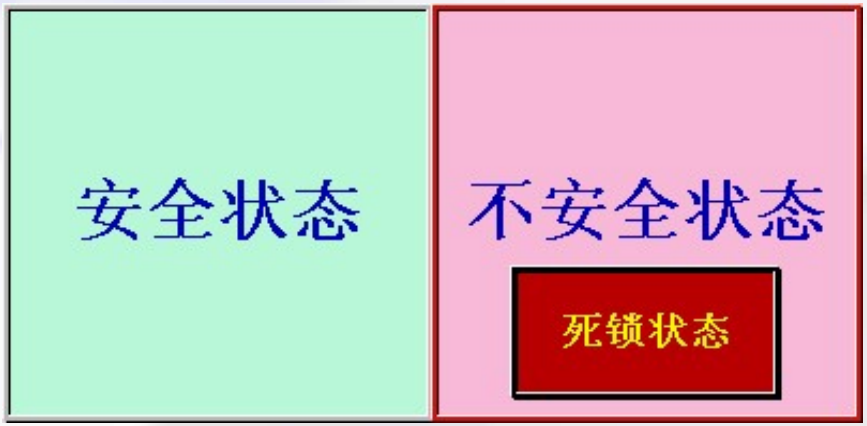
1. 进程死锁概念与条件
2. 进程死锁的预防机制
3. 进程死锁的避免机制
4. 进程死锁检测与解决
5. 进程死锁问题的思考

进程死锁的避免机制

1 机制原理

不是一开始就分配好、预先规划好死锁的避免

- 对进程发出的每一个资源申请进行动态检查
- 根据检查结果决定是否分配资源
- 若试分配后可能发生死锁，则不予分配，否则分配



进程死锁的避免机制

2 银行家算法

分期贷款1000=300+300+400，各期贷款全部到位后完成业务，后可以把钱到期归还给银行

- 银行家拥有一笔周转资金
- 客户要求分期贷款，如果能够得到各期贷款，就一定能够归还贷款，否则就一定不能归还贷款
- 银行家应谨慎地贷款，防止出现坏账
- 银行家采用的具体方法是看是否有足够的剩余资金满足某一客户，如此反复下去
- 如果所有投资最终都被收回，则请求可以批准

进程死锁的避免机制

2 银行家算法

- 操作系统（银行家）
- 操作系统管理的资源（周转资金）
- 进程（要求贷款的客户）

系统拥有某类资源10个		
进程	已有资源数	还要申请资源数
P	4	4
Q	2	2
R	2	7

总共有10个资源，已经分配出去了8个资源，还剩2个资源

进程死锁的避免机制

3 单种资源情况

还剩资源数：2=10-(1+1+2+4)

Marvin -> 4

Barbara-> 5

Andy-> 6

Suzanne

进程死锁的避免机制

3 单种资源情况

问题：一个共有150个存储单元的系统，分配给3个进程，
P1最大需求70，已占有25；
P2最大需求60，已占有40；
P3最大需求60，已占有45

- 情况1：P4进程到达，最大需求60，最初请求25个
- 情况2：P4进程到达，最大需求60，最初请求35个

问能不能分配给P4？如果分配，会不会导致系统，从安全状态变成死锁状态？

进程死锁的避免机制

4 多种资源情况

总的资源E、已分配资源P、剩余资源A

进程	磁带机	绘图仪	打印机	CD-ROM
A	3	0	1	1
B	0	1	0	0
C	1	1	1	0
D	1	1	0	1
E	0	0	0	0

已分配的资源

进程	磁带机	绘图仪	打印机	CD-ROM
A	1	1	0	0
B	0	1	1	2
C	3	1	0	0
D	0	0	1	0
E	2	1	1	0

仍需要的资源

E = (6342)
P = (5322)
A = (1020)

= 6个磁带机、3个绘图仪、4个打印机、2个CD-ROM

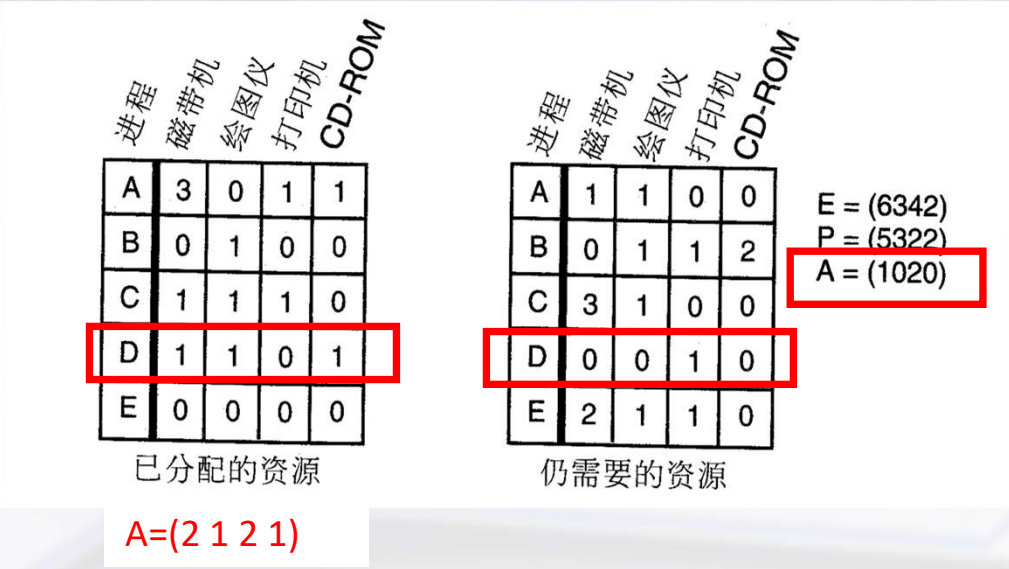
多个进程、多种资源的情况下，
如何避免进程死锁的情况发生？
思路：尝试性先分配，看看系统是否会由安全状态变成死锁状态？
如果会就阻塞进程，不会就分配给它

进程死锁的避免机制

4 多种资源情况

总的资源E、已分配资源P、剩余资源A

- 查找右边矩阵是否有一行，其未被满足的资源数均小于或等于向量A。如果找不到，死锁发生
- 若找到这样一行，假设它获得所需的资源并运行结束，将该进程标记为结束，并将资源加到向量A上
- 重复以上两步，直到所有进程都标记为结束，则状态是安全的，否则将发生死锁



整数运算转变成向量运算

更新A=已分配的释放 + 上一轮的A

进程死锁的避免机制

4 多种资源情况

- 问题：试处理进程P1发出的资源请求(1, 0, 0, 0)

进程	已分配资源数				进程总需资源数				当前系统剩余资源数			
	A	B	C	D	A	B	C	D	A	B	C	D
P_0	0	0	3	2	0	0	4	4	1	6	2	2
P_1	1	0	0	0	2	7	5	0				
P_2	1	3	5	4	3	6	10	10				
P_3	0	3	3	2	0	9	8	4				
P_4	0	0	1	4	0	6	6	10				

进程死锁

本讲内容

1. 进程死锁概念与条件
2. 进程死锁的预防机制
3. 进程死锁的避免机制
4. 进程死锁检测与解决
5. 进程死锁问题的思考

进程死锁检测与解决

1 机制原理

- 允许死锁发生：系统资源并不紧张，发生死锁的概率不是很高
- 系统不断监视进展情况，判断死锁是否发生
- 一旦死锁发生则采取专门的措施，解除死锁并以最小的代价恢复运行

采取先检测，后解除死锁的方法

进程死锁检测与解决

1 机制原理

检测时机

定时检测、

进程等待时 (系统中有大量进程处于等待状态)

资源利用率下降时 (明明很多进程，但是利用率一直下降)

检测手段

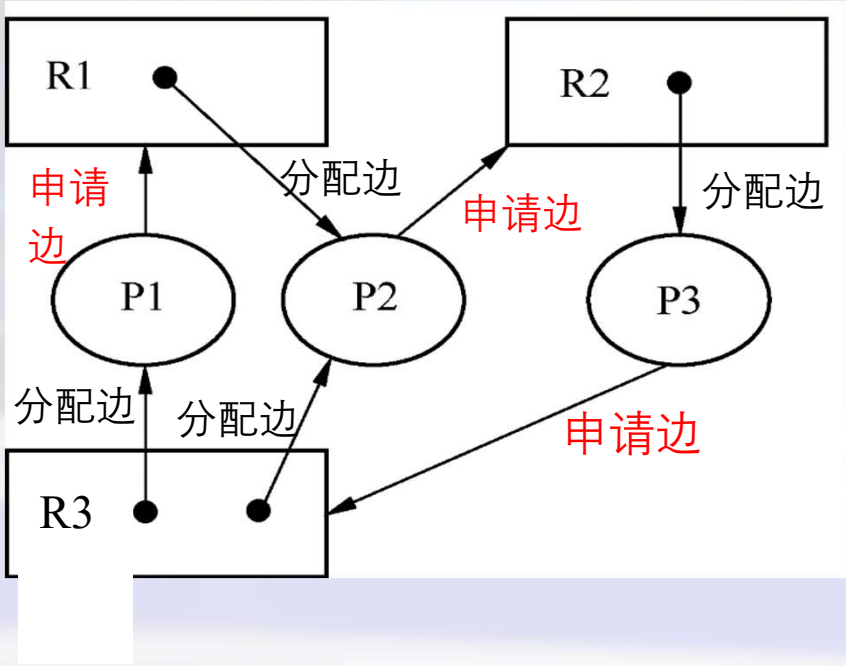
进程-资源分配图

进程死锁检测与解决

2 死锁检测

检测模型

- 方框表示资源类
- 黑圆点表示资源实例
- 圆圈中加进程名表示进程



- 资源实例指向进程的一条有向边来表示分配边
- 进程指向资源类的一条有向边来表示申请边
- 检测“进程-资源分配图”是否可完全简化

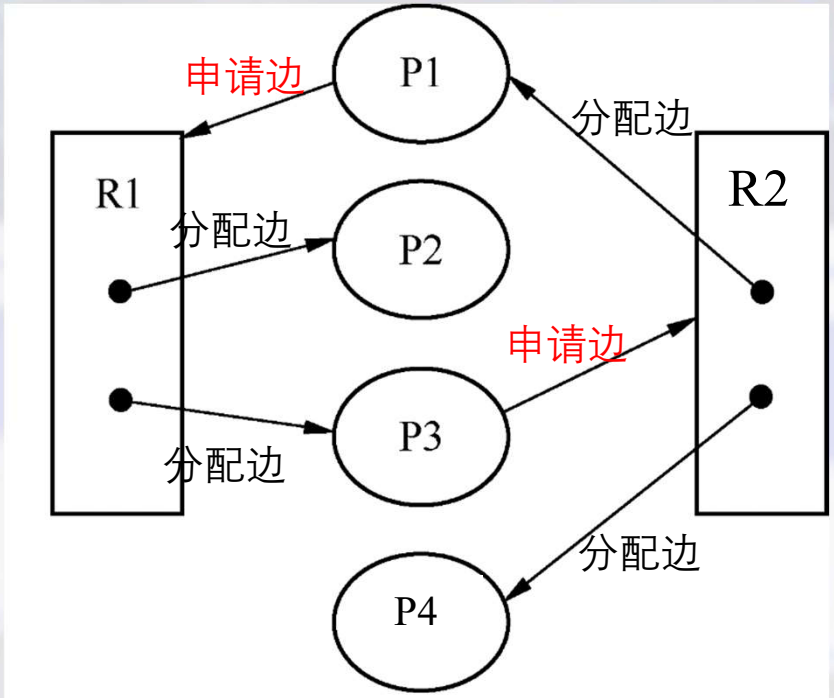
进程死锁检测与解决

只有分配边的进程，已经获得了他们所需要的所有资源

2 死锁检测

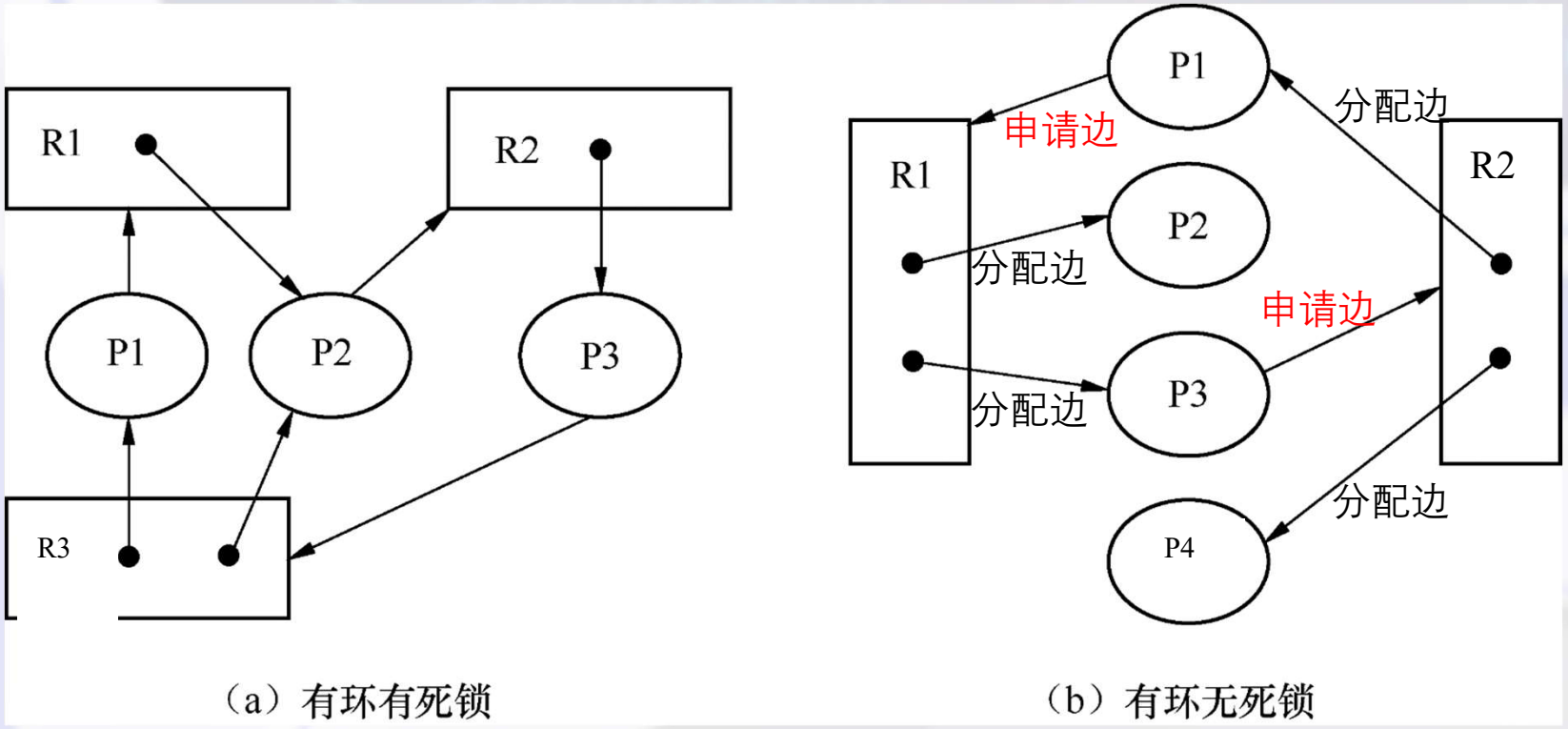
检测步骤

- ①找一个只有分配边的非孤立进程结点，去掉分配边将其变为孤立结点；若找不到则转③
- ②将资源分配给一个等待资源的进程，将某进程的申请边变为分配边，转①
- ③图中有进程不是孤立结点，则此图不可完全简化，满足死锁的充分条件，系统为死锁状态



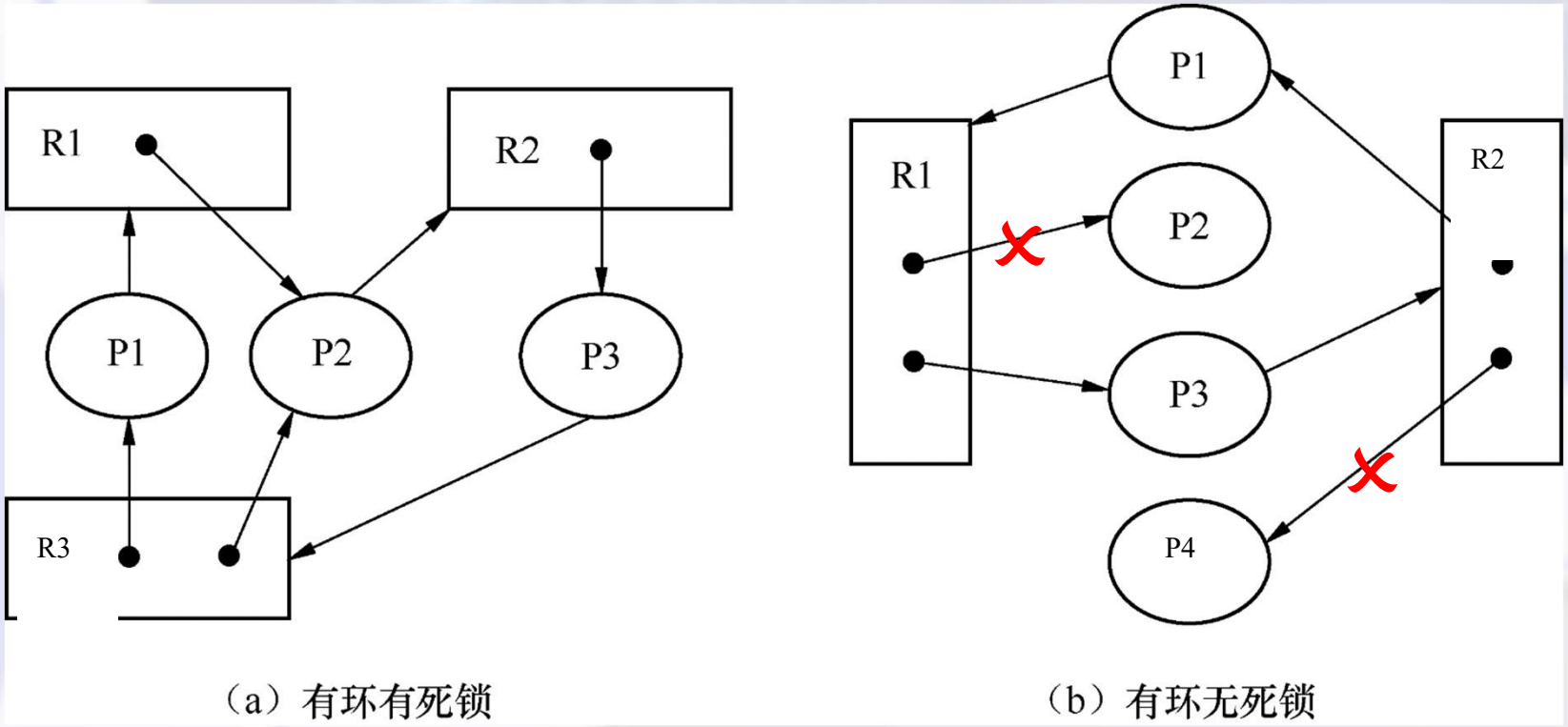
进程死锁检测与解决

2 死锁检测



进程死锁检测与解决

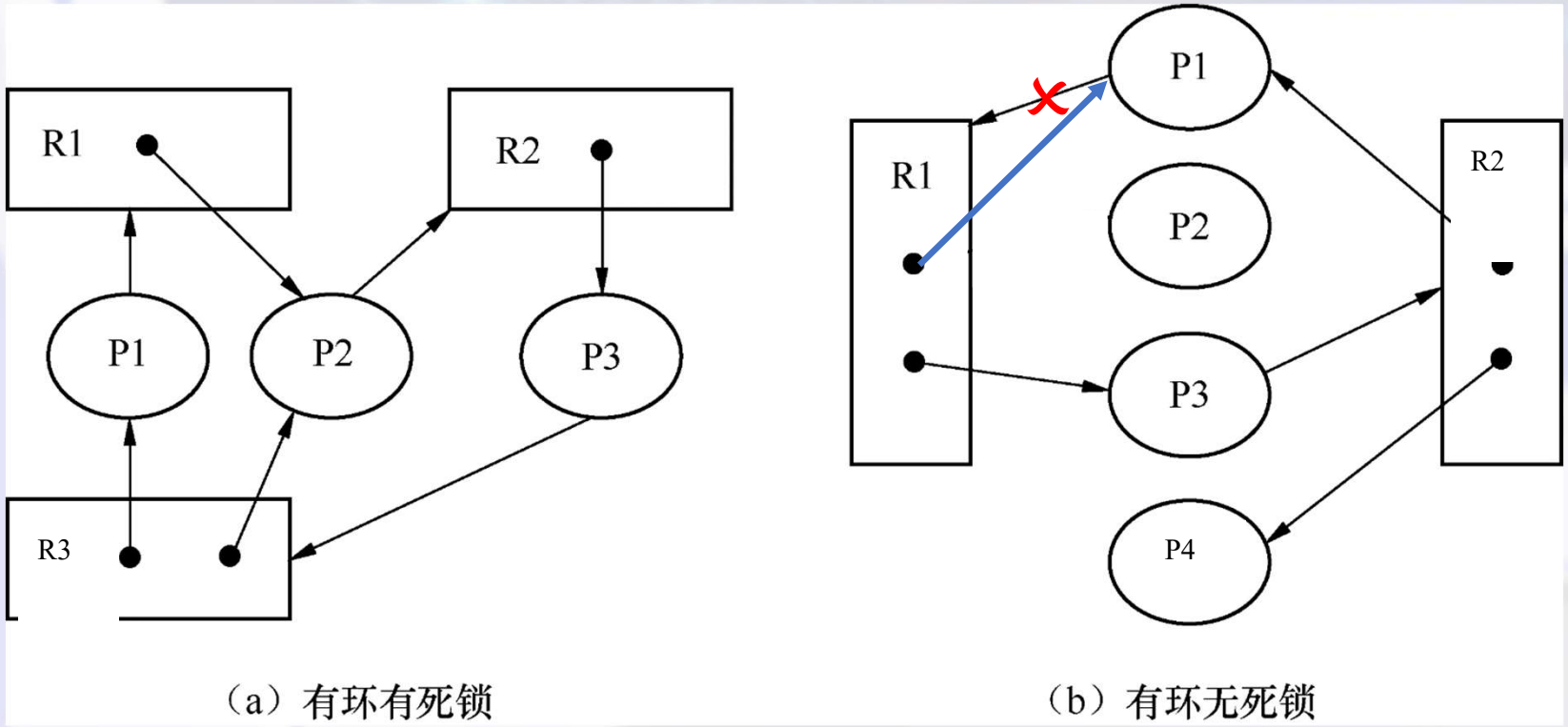
2 死锁检测



（a）有环有死锁　　（b）有环无死锁

进程死锁检测与解决

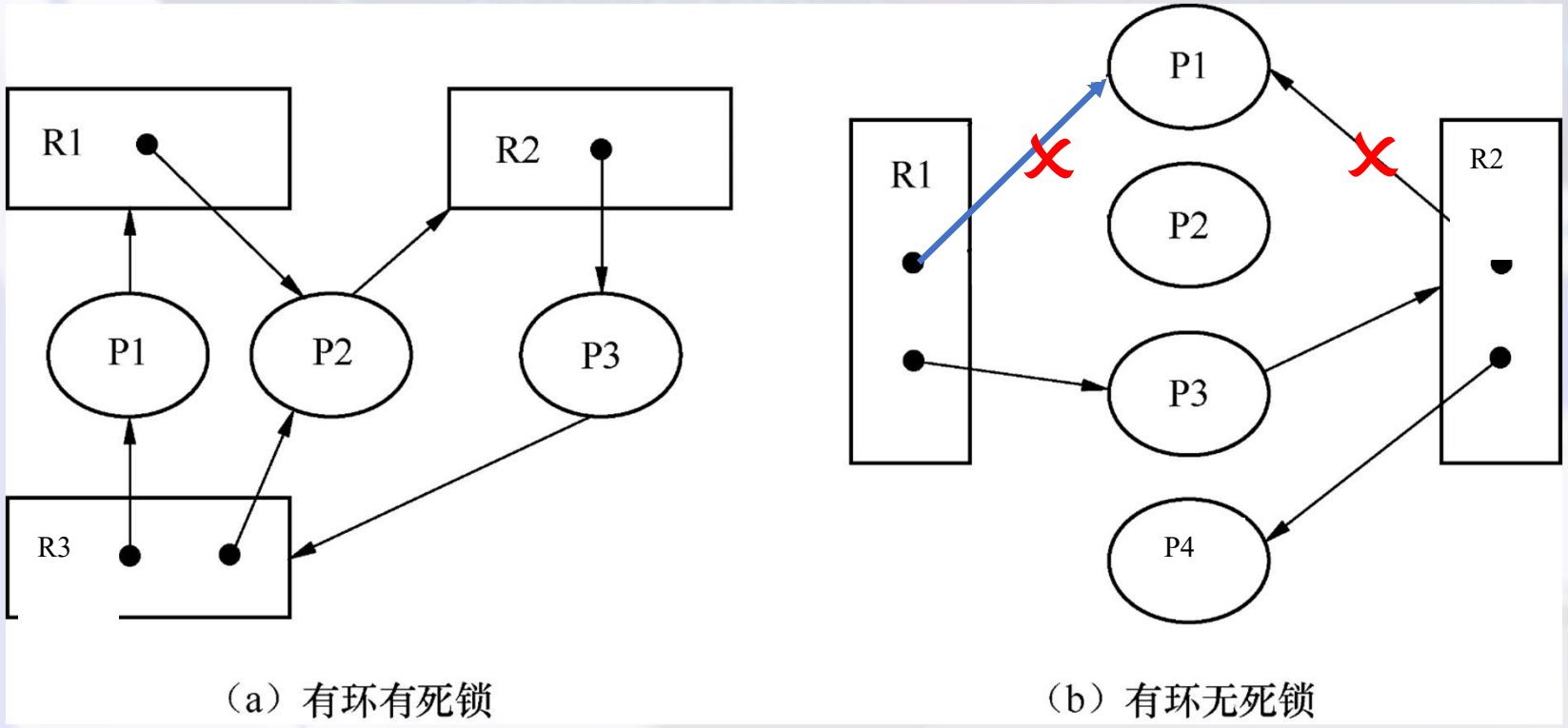
2 死锁检测



（a）有环有死锁　　（b）有环无死锁

进程死锁检测与解决

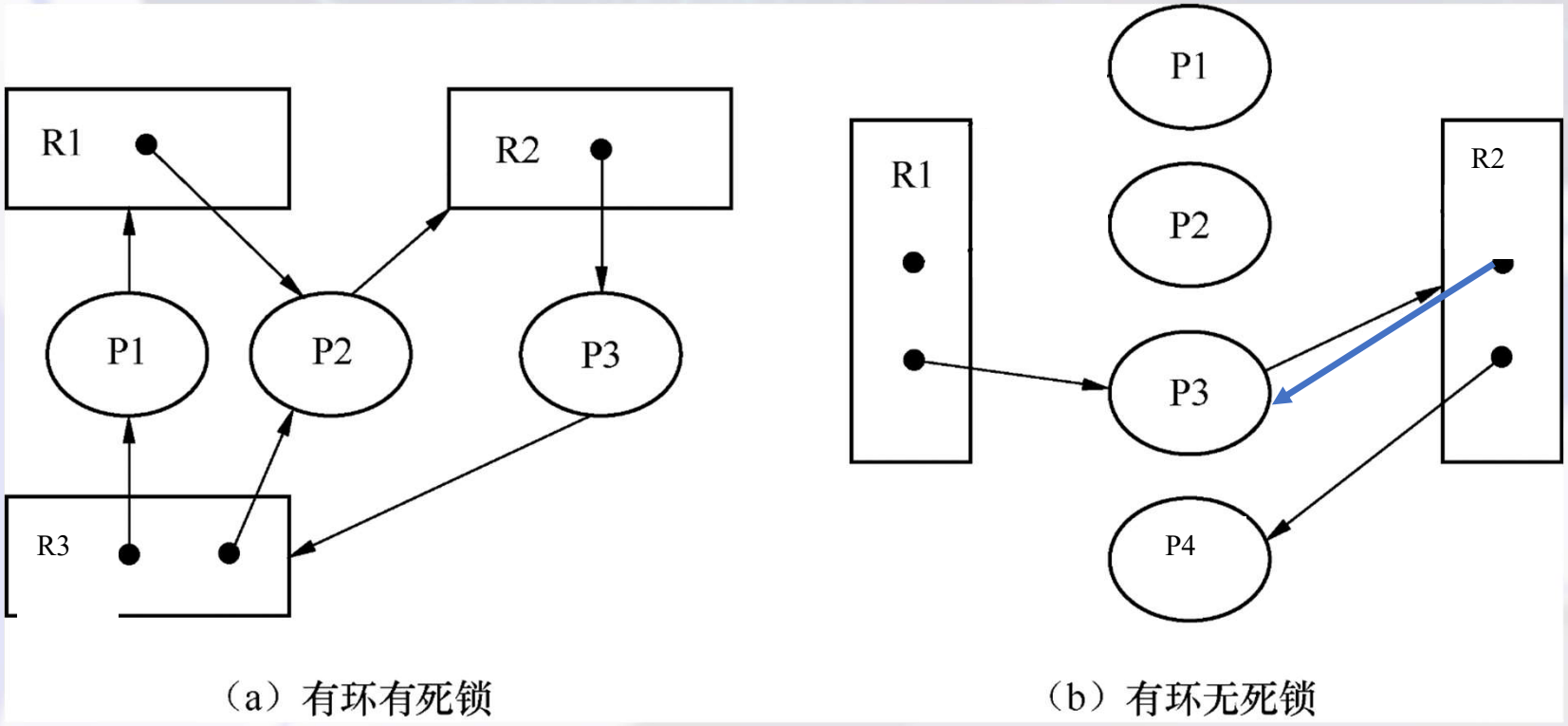
2 死锁检测



（a）有环有死锁　　（b）有环无死锁

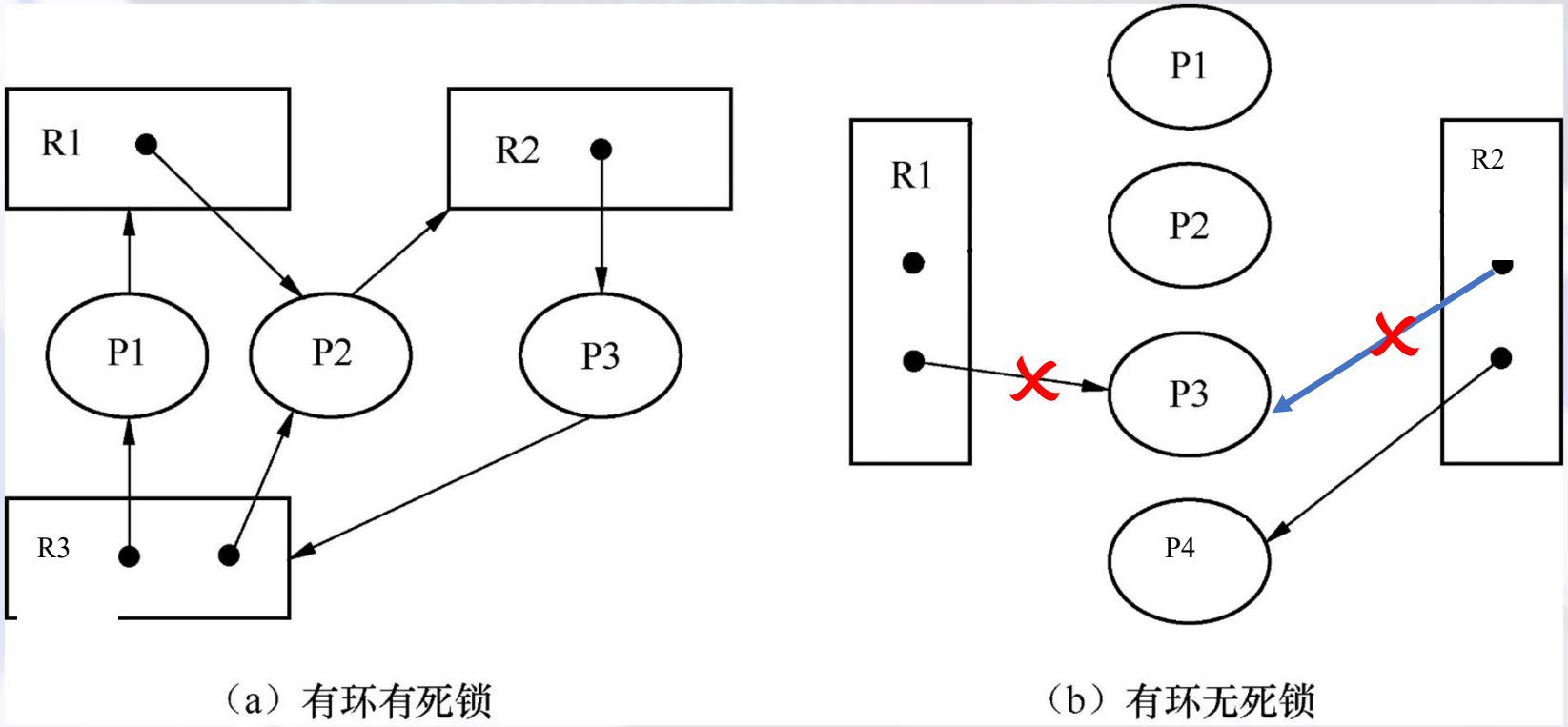
进程死锁检测与解决

2 死锁检测



进程死锁检测与解决

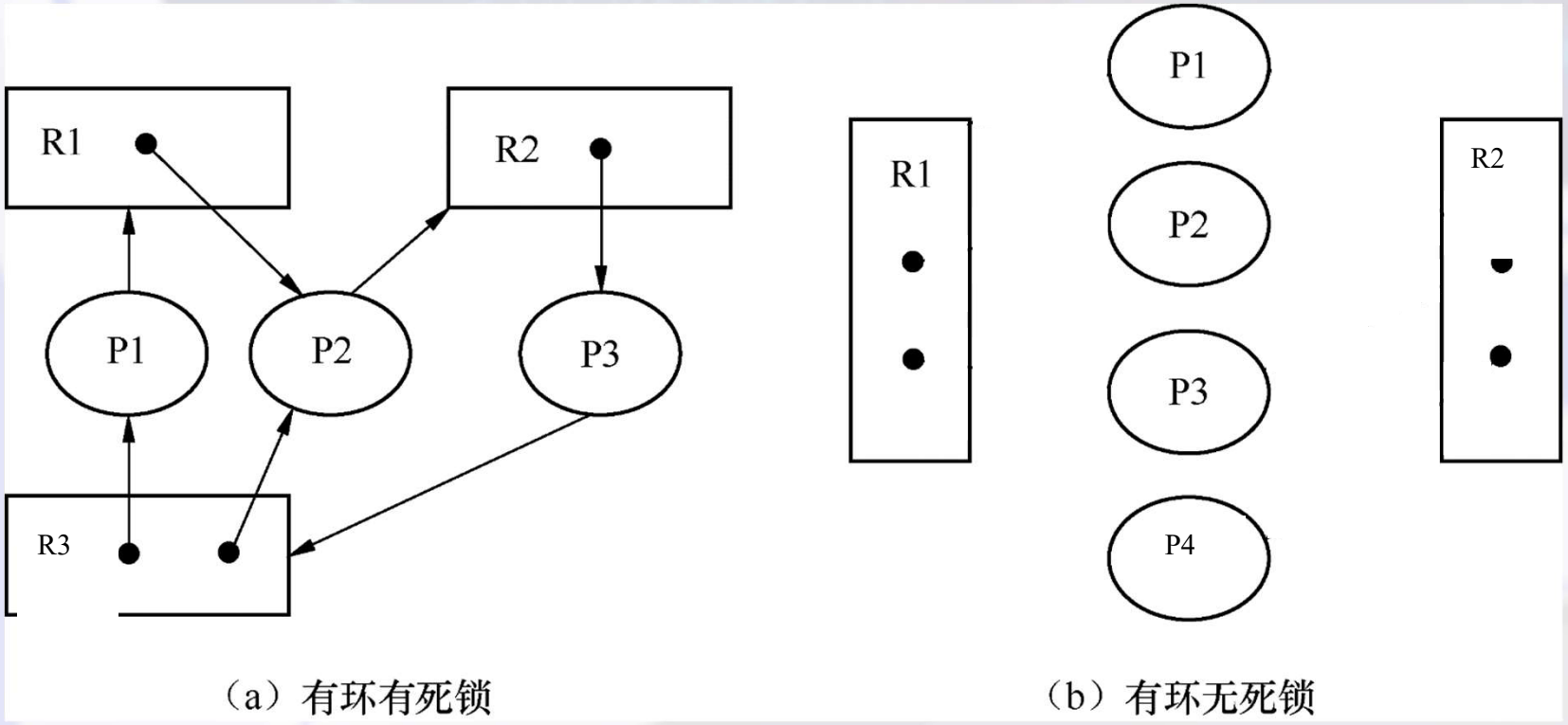
2 死锁检测



（a）有环有死锁　　（b）有环无死锁

进程死锁检测与解决

2 死锁检测



（a）有环有死锁　　（b）有环无死锁

进程死锁检测与解决

3 死锁解除

资源剥夺法

- 从其他进程那里剥夺足够数量的资源给死锁进程，以解除死锁状态

撤消进程法

- 撤消全部死锁进程，恢复到正常状态，简单但代价太大
- 按照某种顺序逐个撤消死锁进程，直到有足够的资源供其他未被撤消的进程使用，以消除死锁状态

进程死锁

本讲内容

1. 进程死锁概念与条件
2. 进程死锁的预防机制
3. 进程死锁的避免机制
4. 进程死锁检测与解决
5. 进程死锁问题的思考

进程死锁问题的思考

1 死锁原因

- 系统资源不足
- 进程运行推进的顺序不合适
- 资源分配不当

进程死锁问题的思考

2 解决原则

- 单独使用死锁预防、避免、检测与解除并不能全面解决操作系统中遇到的所有死锁问题

- 可将系统中的进程、资源分为若干类，对每一类进程、资源使用最适合它的办法解决死锁

1. 产生死锁的4个必要条件是互斥使用、不可剥夺、请求和保持、循环等待。

2、系统中两个或者多个进程无限期地等待永远不会发生的条件，系统处于停滞状态，这种现象称为进程死锁。

7、 银行家算法是一种（ B ）算法。

A）死锁解除　　B）死锁避免　C）死锁预防　D）死锁检测

2、用文字描述银行家算法的基本思想。

银行家算法的基本思想是：将系统中的所有资源比做银行家的资金，每进行一次资源的分配，银行家都要从当前的资源分配情况出发，计算这种分配方案的安全性（2分），如果是安全的，则进行分配，否则选择其它可能的分配方案（2分）。这样，每次分配都计算安全性，从而可以避免死锁的发生（1分）。

1. 请简要说明死锁的概念及其四个必要条件。

死锁定义：系统中两个或者多个进程无限期地等待永远不会发生的条件，系统处于停滞状态，这种现象称为进程死锁，这一组进程就称为死锁进程。【2分】

四个必要条件：

（1）互斥使用（资源独占）：一个资源每次只能给一个进程使用；

（2）不可强占（不可剥夺）：资源申请者不能强行地从资源占有者手中夺取资源，资源只能由占有者自愿释放；

（3）请求和保持（部分分配，占有申请）：一个进程在申请新的资源的同时保持对原有资源的占有（只有这样才是动态申请，动态分配）；

（4）循环等待：存在一个进程等待队列 {P1，P2，…，P*n*}，其中P_i等待$P_{(i+1) \bmod n}$占有的资源，形成一个进程等待环路。【每点1分】

1、 有三个进程P1，P2和P3并发工作。进程P1需用资源S3和S1；进程P2需用资源S1和S2；进程P3需用资源S2和S3，会发生什么？回答：

(1) 若对资源分配不加限制，会出现什么情况?为什么?

(2) 为保证程正确工作，应采用怎样的资源分配策略?为什么?

答：(1)可能会发生死锁（3分）

例如：进程P1，P2和P3分别获得资源S3，S1和S2后再继续申请资源时都要等待(2分)，这是循环等待。(或进程在等待新源时均不释放已占资源)

(2)可有几种答案：（3分）

A.采用静态分配　由于执行前已获得所需的全部资源，故不会出现占有资源又等待别的资源的现象(或不会出现循环等待资源现象)。

或B.采用按序分配　　不会出现循环等待资源现象。

或C.采用银行家算法　　因为在分配时，保证了系统处于安全状态。