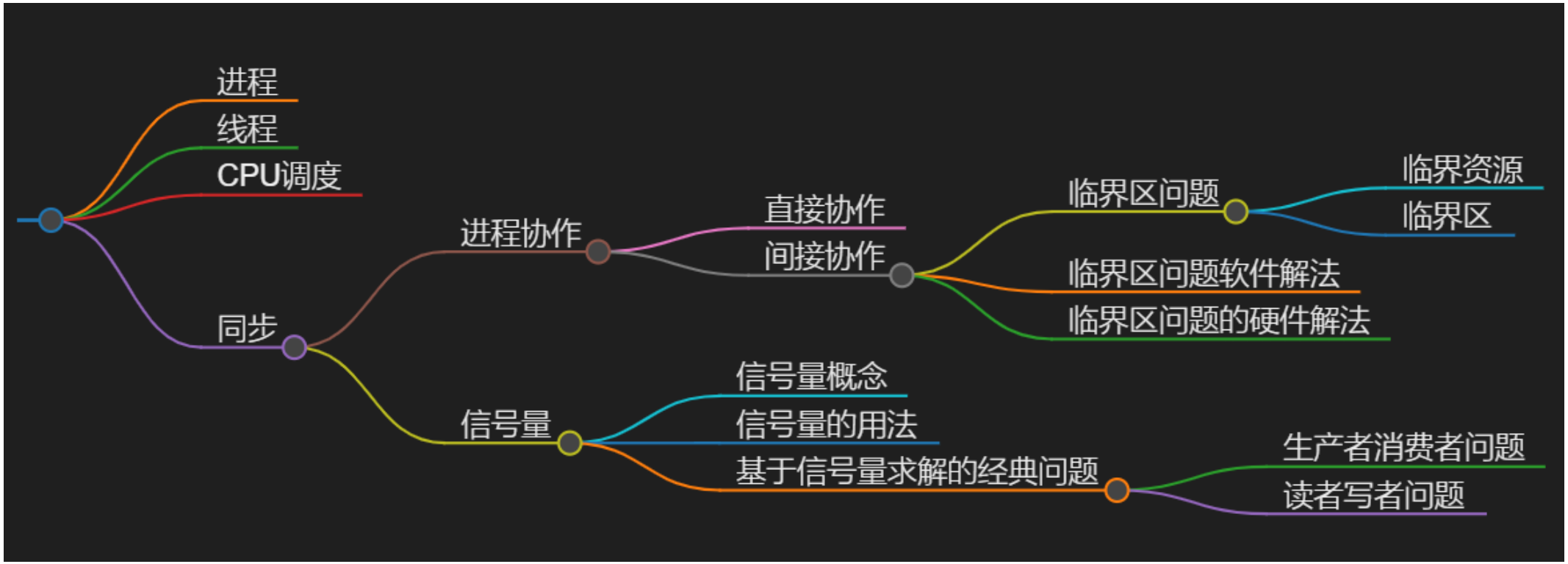
操作系统

L11 死锁1

胡燕
大连理工大学 软件学院

前情回顾

进程同步机制


进程
线程
CPU调度
同步
进程协作
直接协作
间接协作
临界区问题
临界资源
临界区
临界区问题软件解法
临界区问题的硬件解法
信号量
信号量概念
信号量的用法
基于信号量求解的经典问题
生产者消费者问题
读者写者问题

n 个并发进程，信号量初始值为 1，当 n 个进程都执行 P 操作后，信号量的值为(　)。

信号量初值为 4，多次 PV 操作后变为 -2，那么当前时刻已经顺利获得资源的进程数目=（ ）。

5 个并发进程，信号量初始值为 3，那么信号量取值范围是整数区间为[(　), (　)]。

请用信号量解决以下的“过独木桥”问题：
同一方向的行人可连续过桥，当某一方向有人过桥时，另一方向的行人必须等待；当某一方向无人过桥时，另一方向的行人可以过桥。

主观题 10分

设置

请用信号量解决以下的“过独木桥”问题：
同一方向的行人可连续过桥，当某一方向有人过桥时，另一方向的行人必须等待；当某一方向无人过桥时，另一方向的行人可以过桥。

作答

Pthread中使用semaphore:示例

```
int cnt = 0;

void * Count(void * a)
{
    int i, tmp;
    for(i = 0; i < NITER; i++)
    {
        cnt += 1;
    }
}
```

```
if(pthread_create(&tid1, NULL, Count, NULL))
{
    printf("\n ERROR creating thread 1");
    exit(1);
}
```

```
if(pthread_create(&tid2, NULL, Count, NULL))
{
    printf("\n ERROR creating thread 2");
    exit(1);
}
```

参考：http://www.csc.villanova.edu/~mdamian/threads/posixsem.html

Pthread中使用semaphore

信号量API	信号量API原型
sem_init	int sem_init(sem_t *sem, int pshared, unsigned int value);
sem_wait	int sem_wait(sem_t *sem);
sem_post	int sem_post(sem_t *sem);
sem_getvalue	int sem_getvalue(sem_t *sem, int *valp);
sem_destroy	int sem_destroy(sem_t *sem);

参考：http://www.csc.villanova.edu/~mdamian/threads/posixsem.html

Pthread中使用semaphore:示例

```
int cnt = 0;

void * Count(void * a)
{
    int i, tmp;
    for(i = 0; i < NITER; i++)
    {
        tmp = cnt;      /* copy the global cnt locally */
        tmp = tmp+1;    /* increment the local copy */
        cnt = tmp;      /* store the local value into the global cnt */
    }
}
```

```
sem_t mutex;

sem_init(&mutex, 0, 1);

sem_wait (&mutex);

sem_post (&mutex);
```

参考：http://www.csc.villanova.edu/~mdamian/threads/posixsem.html

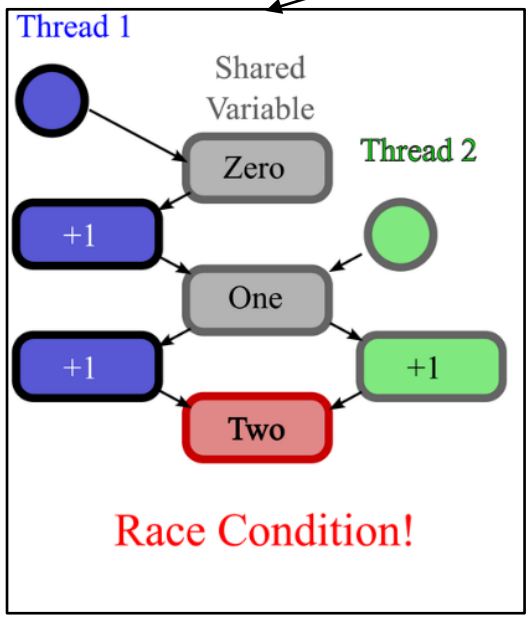
并发控制中的问题

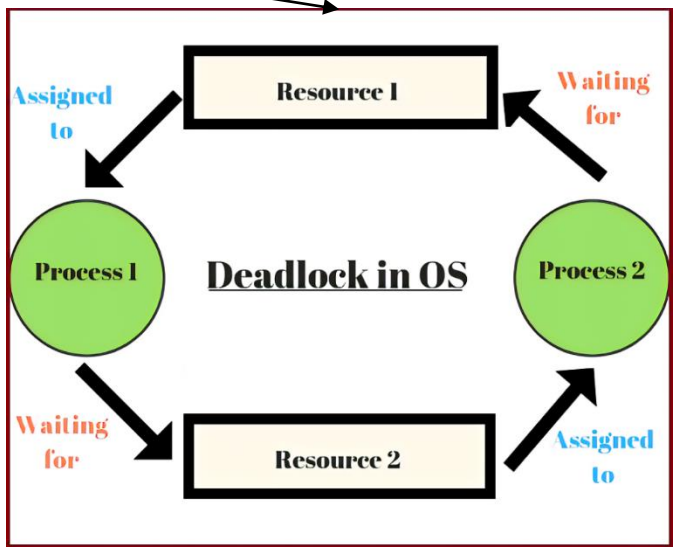
并发操作中的问题

并发控制容易吗?

不容易，稍有不慎就会出问题

典型的并发控制问题





Race Condition: 对共享变量并发操作的竞态冲突问题(不一致问题)

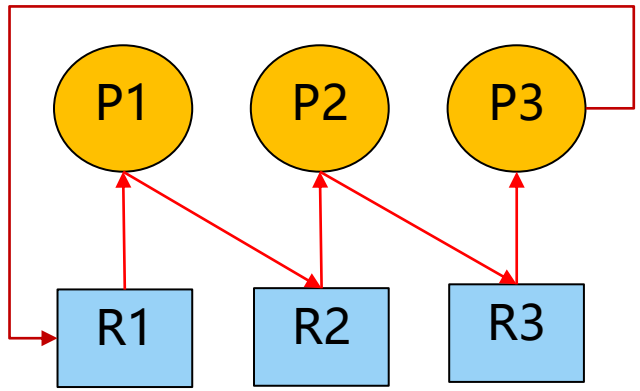
Deadlock: 即死锁，典型成因是加锁操作不当导致若干进程永远无法进一步推进

什么是死锁

死锁概念

Deadlock represents a condition where there exist a set of blocked processes, each holding at least one resource and waiting to acquire a resource held by another process in the set.

死锁是指一种系统状态，在该状态下，存在一组进程，其中每一个进程都持有至少一种资源，并且在申请由该组进程中的另外一个进程所持有的资源。



现实世界中的死锁

俄罗斯-乌克兰局势：

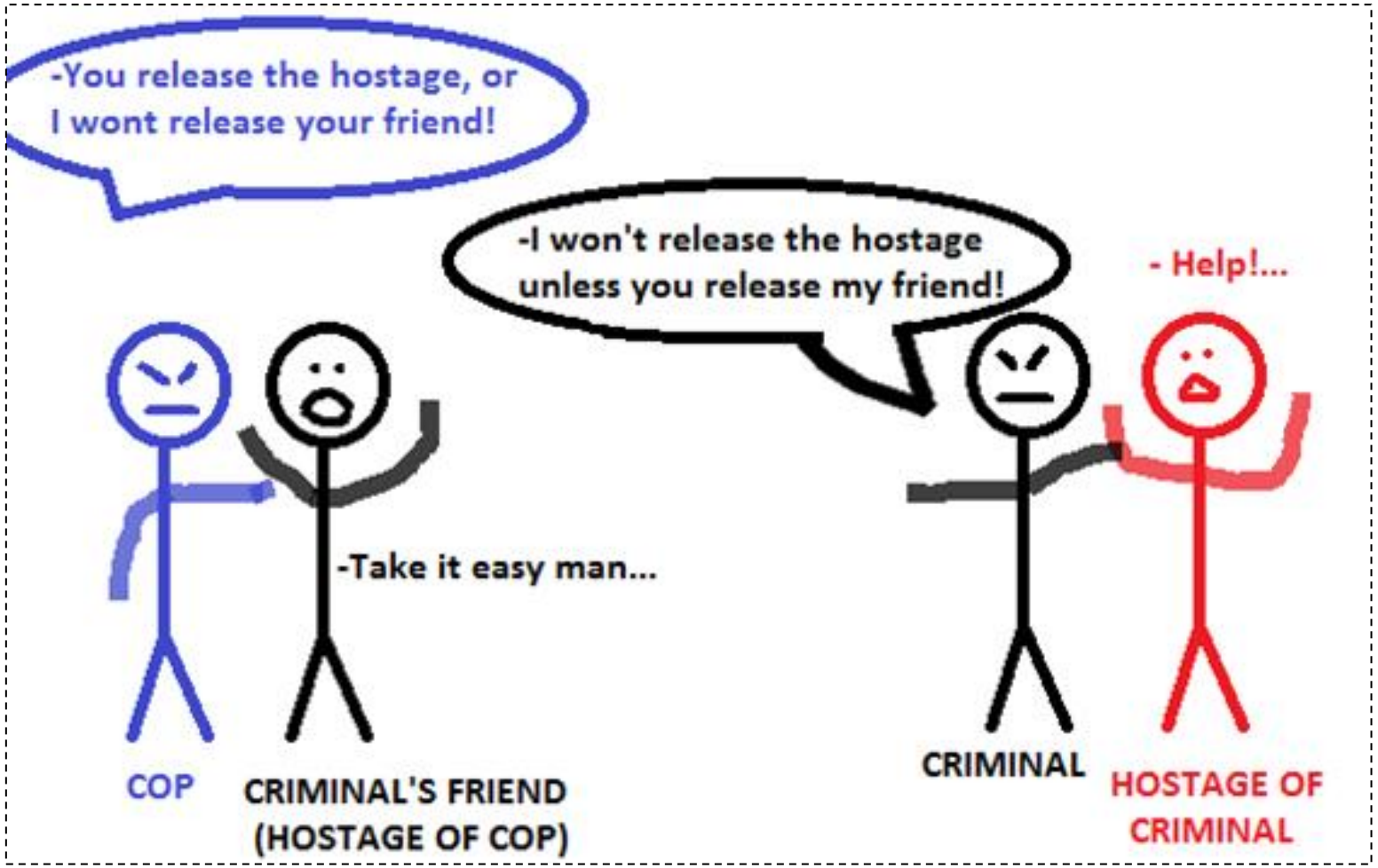
俄罗斯：以打促谈，军事介入，要求谈判保证乌克兰不加入北约，不再攻打乌东俄族区
（我停止军事行动，乌克兰就不可能谈，所以边军事行动，边倡导谈判）

乌克兰：要求俄罗斯先退出乌东，再谈

要维持住这个僵局，必须存在军事拉锯（北约在持续提供武器）

典型死锁场景

场景1：解救人质行动



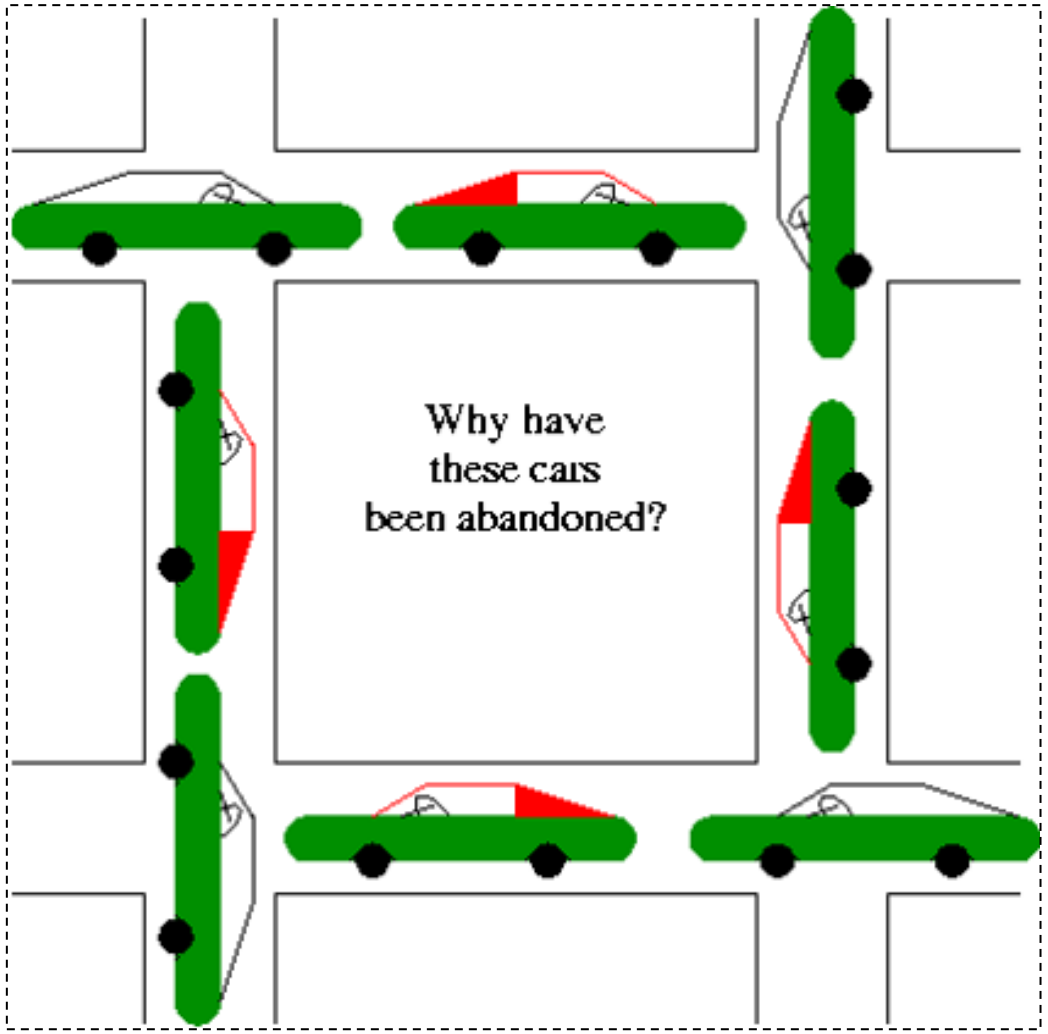
Who will act first?
- No one because each of them waits for the other to act.

R1: hostage of the criminal

R2: friend of the criminal
(hostage of the cop)

典型死锁场景

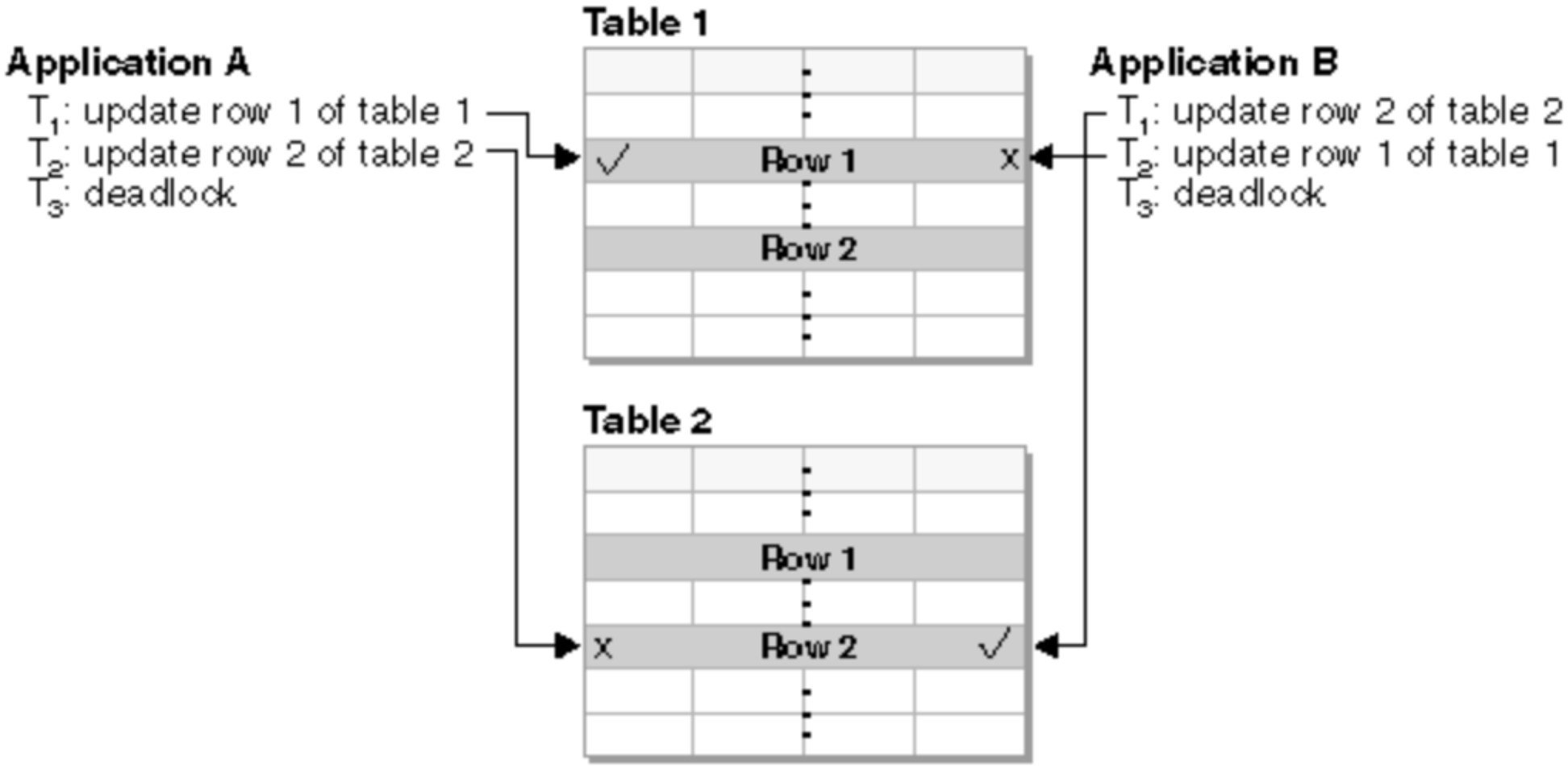
场景2：Traffic Jam



此场景中竞争的资源是什么？

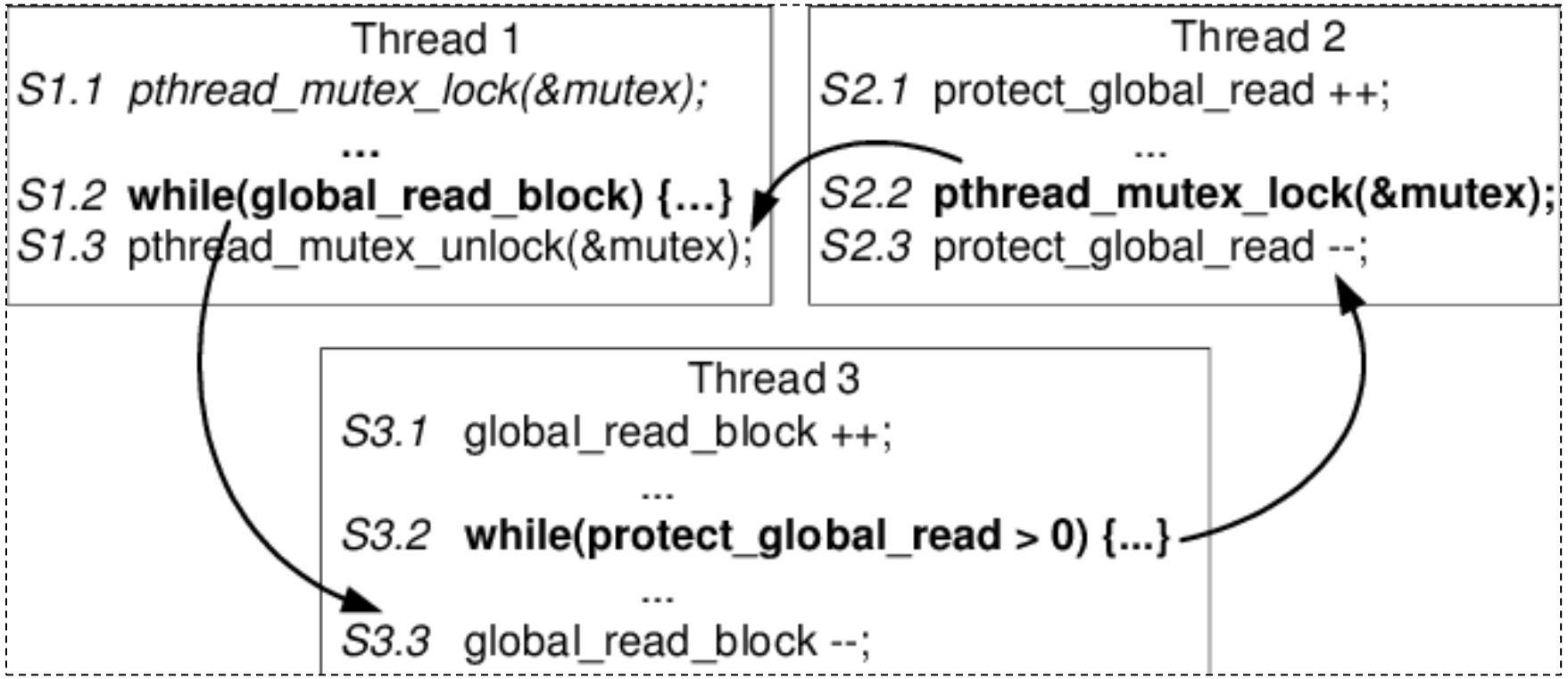
软件系统的死锁案例

案例1：MySQL deadlock scenario



软件系统的死锁案例

案例2：Apache Server deadlock scenario



执行顺序

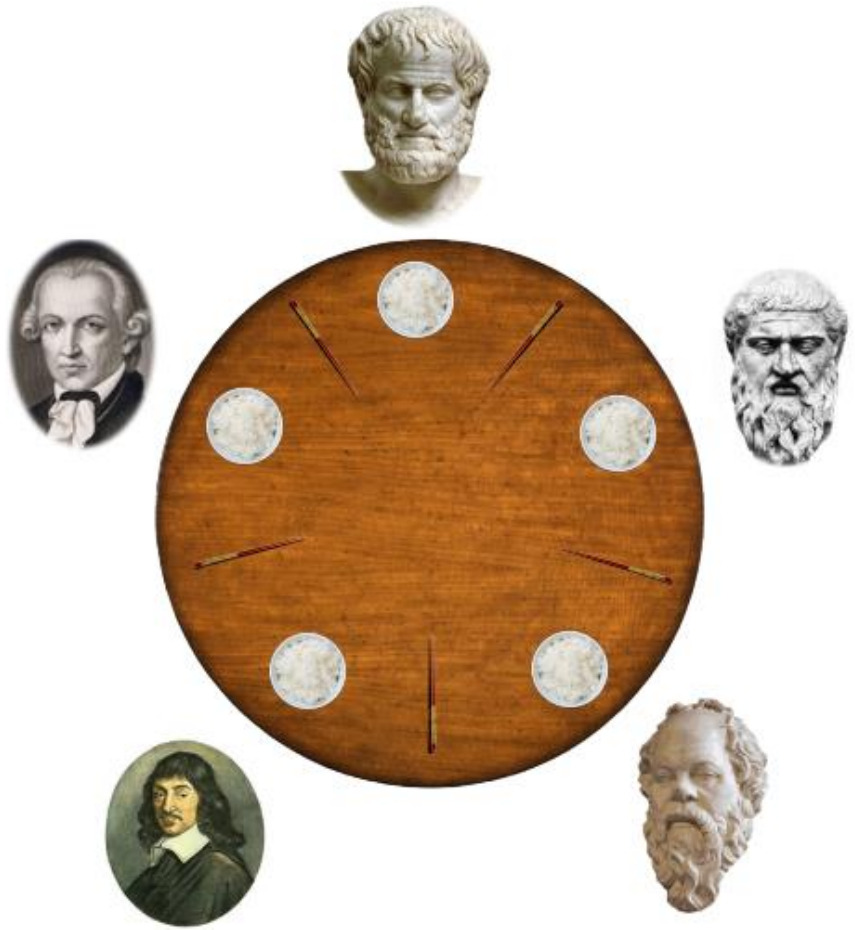
S1.1

S2.1

S3.1

经典死锁问题

Dining Philosophers



五位哲学家

就餐，思考，就餐，思考，...

问题1： 系统资源不足

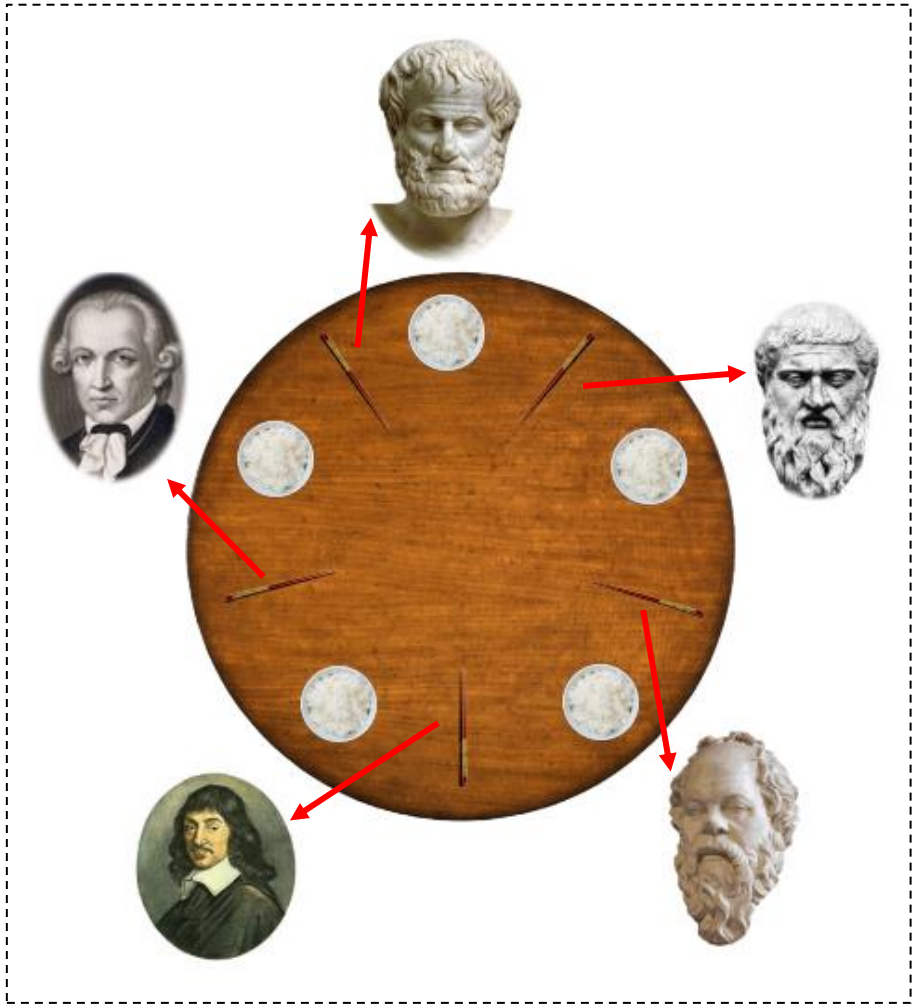
可能导致非常激烈的竞争

问题2： 对用餐资源（筷子）的竞争

无序竞争可能导致严重后果

经典死锁问题

Dining Philosophers



资源不足情况下，对于资源分配过程安排不当的后果：

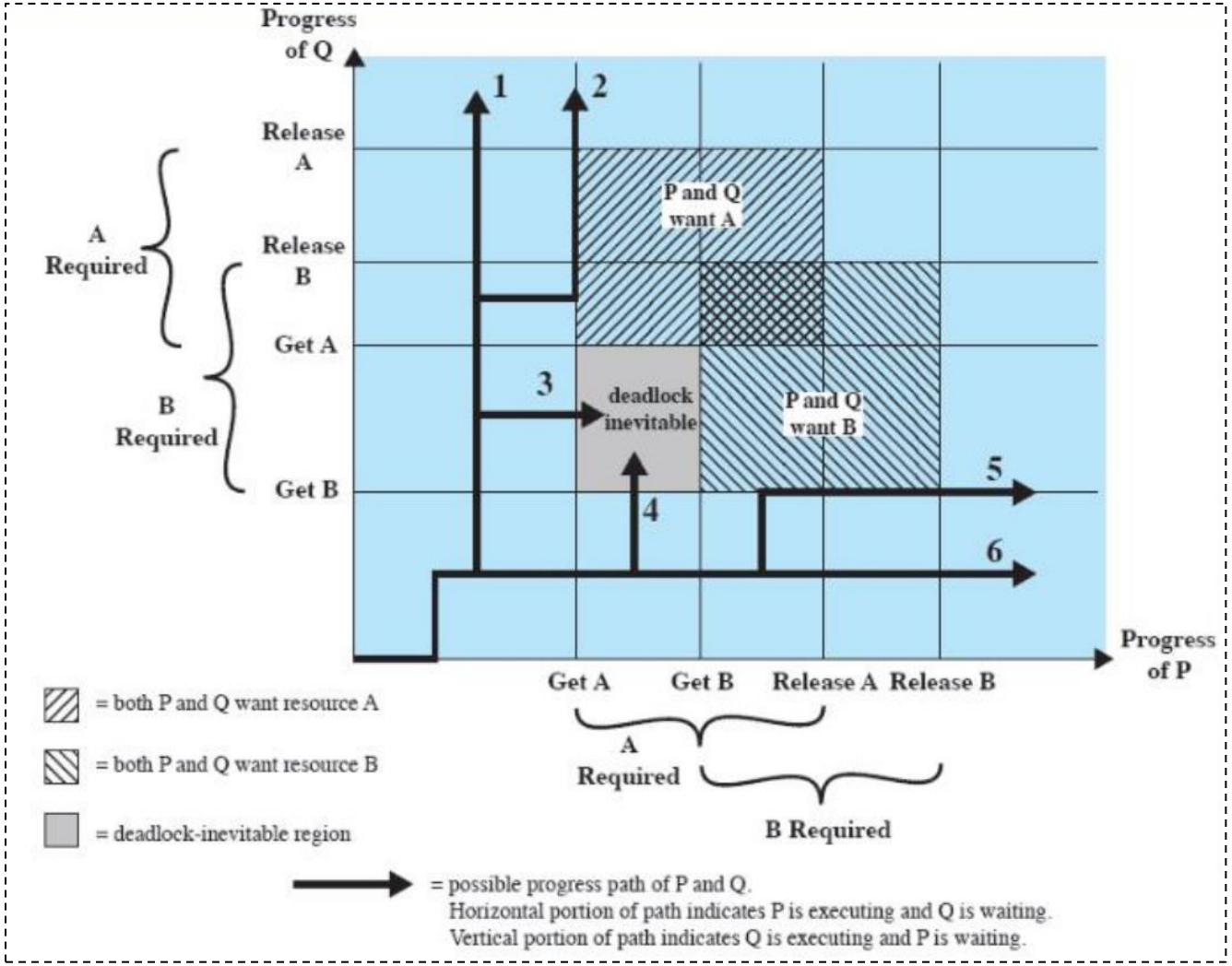
Deadlock

死锁根本原因

死锁的两大根本原因

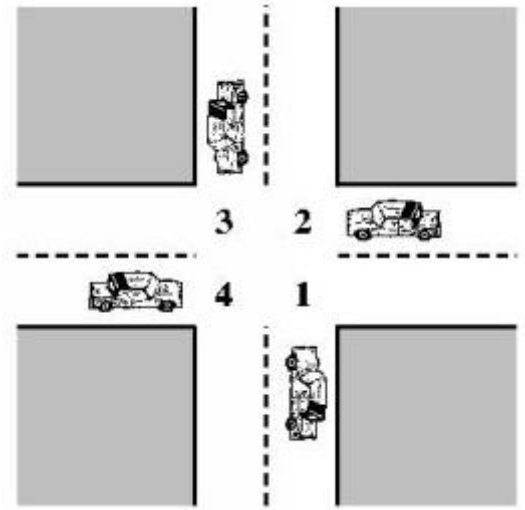
1.系统资源不足

2.进程推进顺序不当



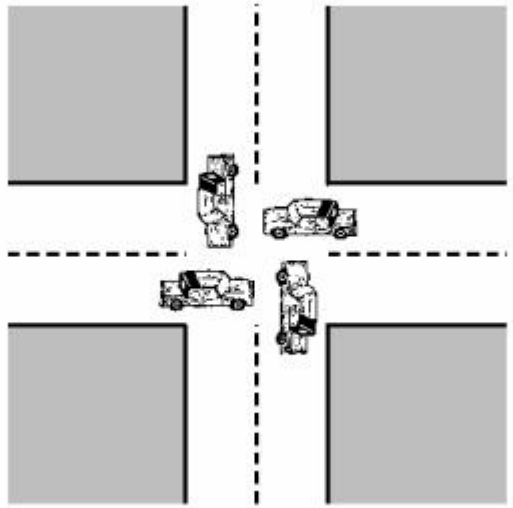
死锁根本原因

死锁成因示例



(a)Deadlock possible

十字路口，空位资源稀缺



(b)Deadlocked

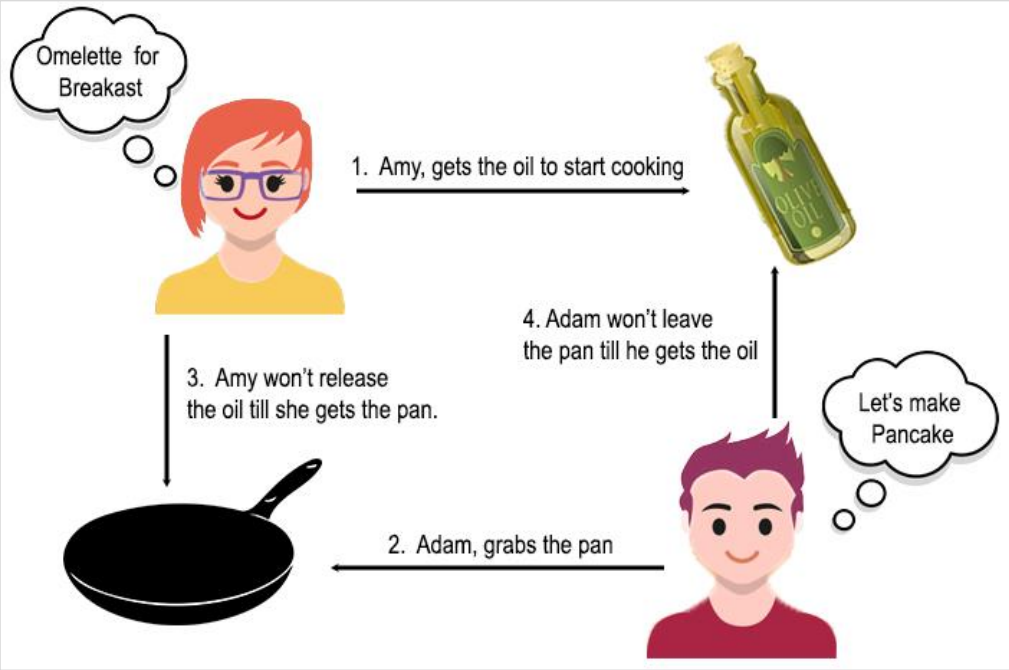
4车前进方式不当，导致死锁

死锁的四大必要条件

- **形成死锁的四大必要条件**
  - 资源以互斥方式使用（Mutual exclusion）
  - 持有并等待（Hold and wait）
  - 已持有资源不可被剥夺（No preemption）
  - 循环等待（Circular wait）

死锁的四大必要条件

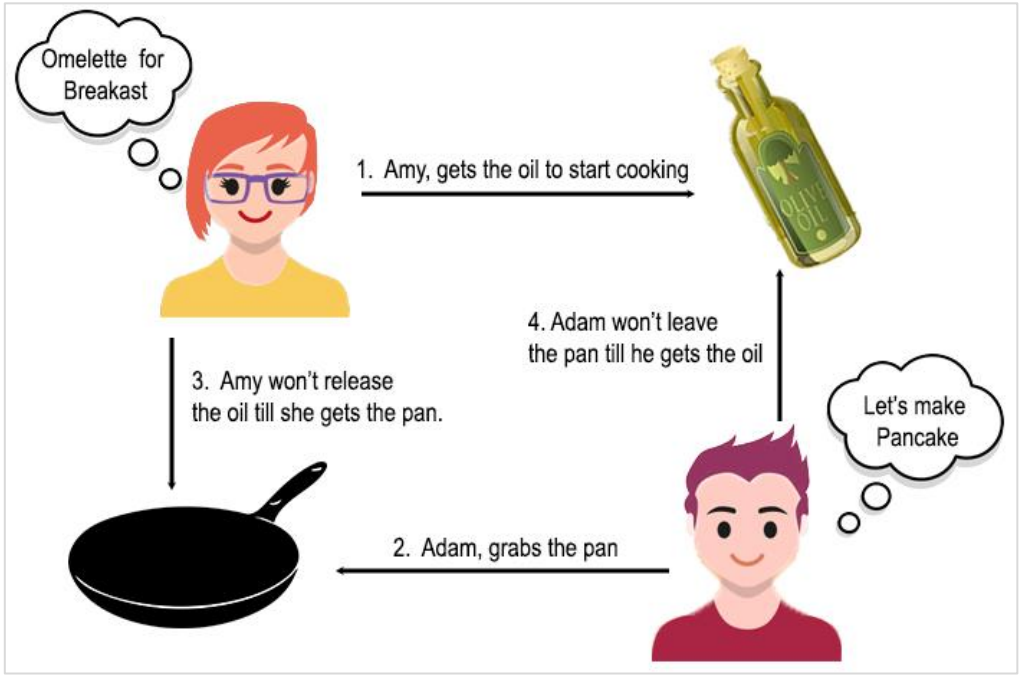
Mutual Exclusion | Hold and Wait | No Preemption | Circular Wait



如果当前状态下，Amy说：
“All right, Adam, please start making your pancake, I will serve you some olive oil”

死锁的四大必要条件

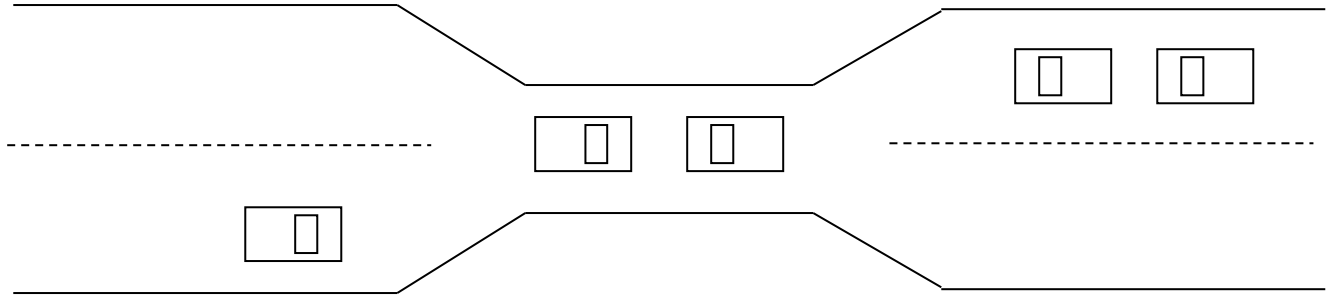
Mutual Exclusion | Hold and Wait | No Preemption | Circular Wait



Amy holds oil, and waits for pan
Adam holds pan, and waits for oil

死锁的四大必要条件

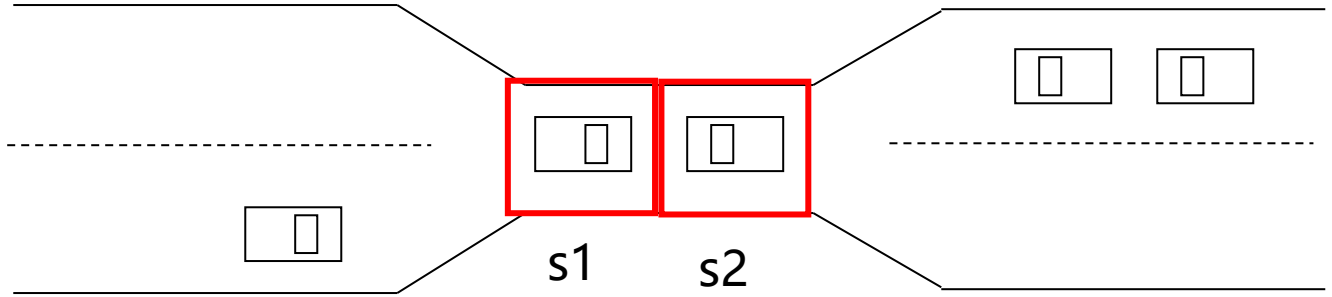
- **Hold-and-Wait (部分持有资源)**
  - 示例：过窄桥



请尝试作下分析

死锁的四大必要条件

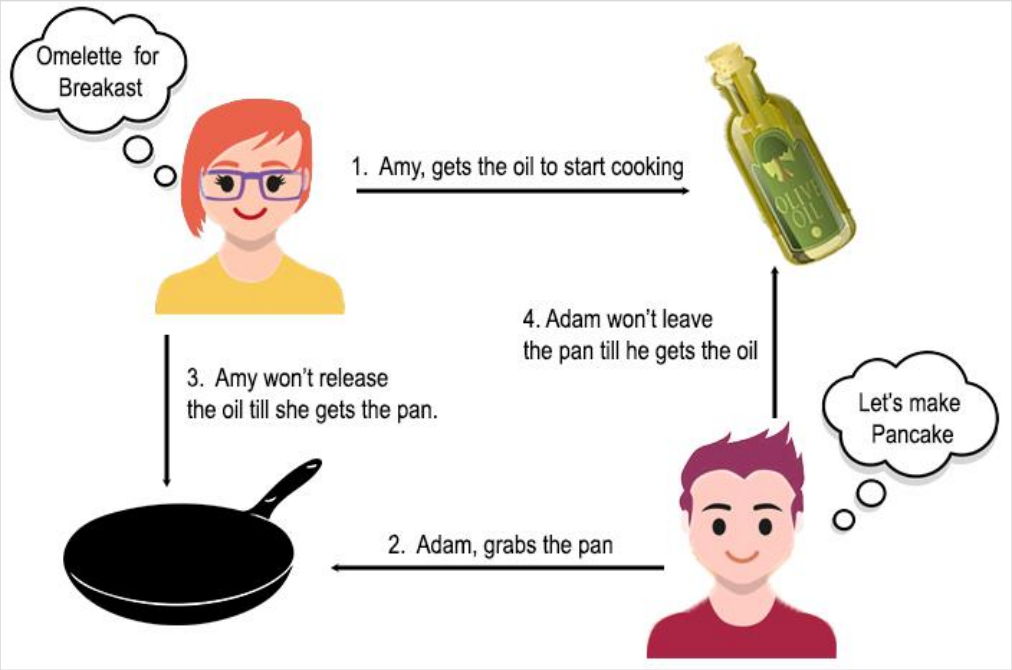
- **Hold-and-Wait (部分持有资源)**
  - 示例：过窄桥



左侧车占据S1，但还需要占据S2才能通过
右侧车占据S2，但还需要占据S1才能通过

死锁的四大必要条件

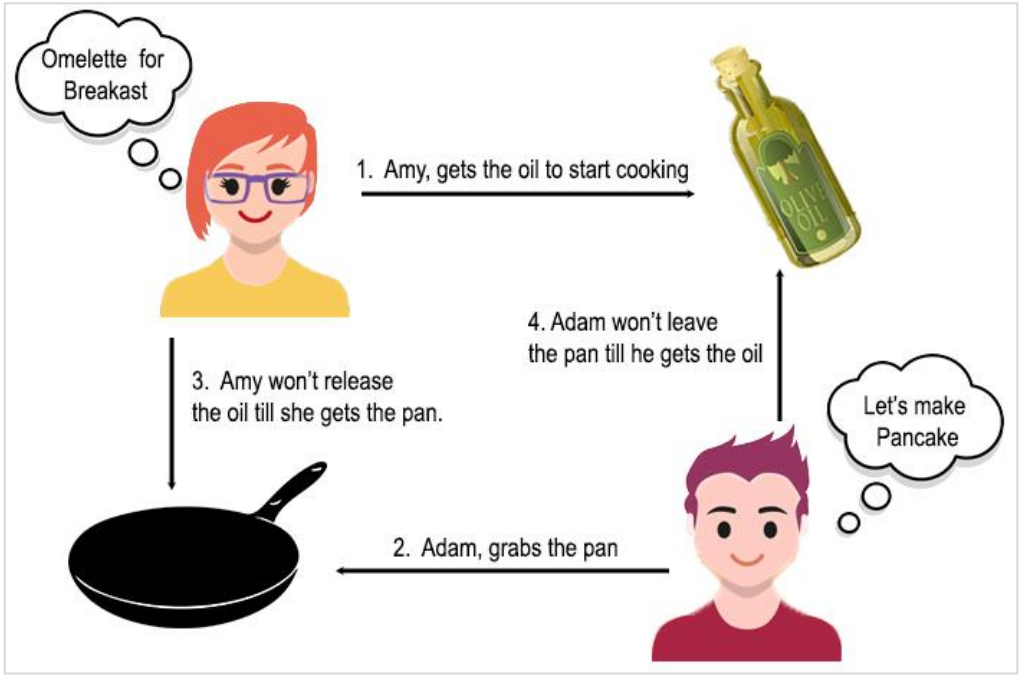
Mutual Exclusion | Hold and Wait | **No Preemption** | Circular Wait



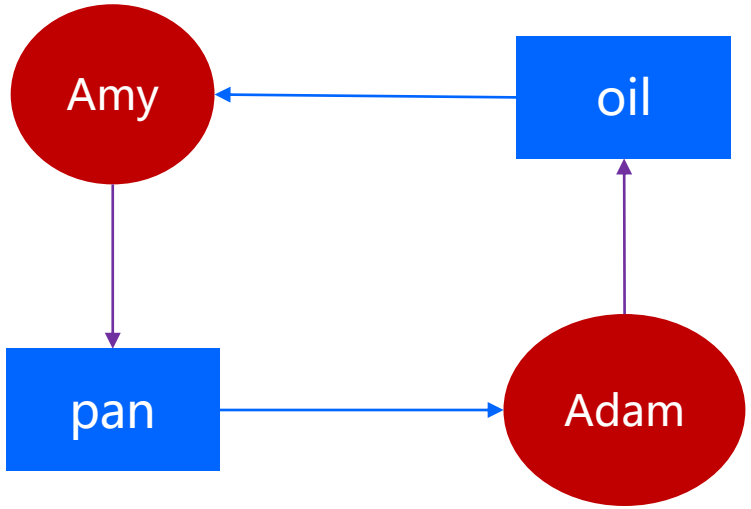
What if Amy just snatch the pan, and take it away from Adam

死锁的四大必要条件

Mutual Exclusion | Hold and Wait | No Preemption | Circular Wait

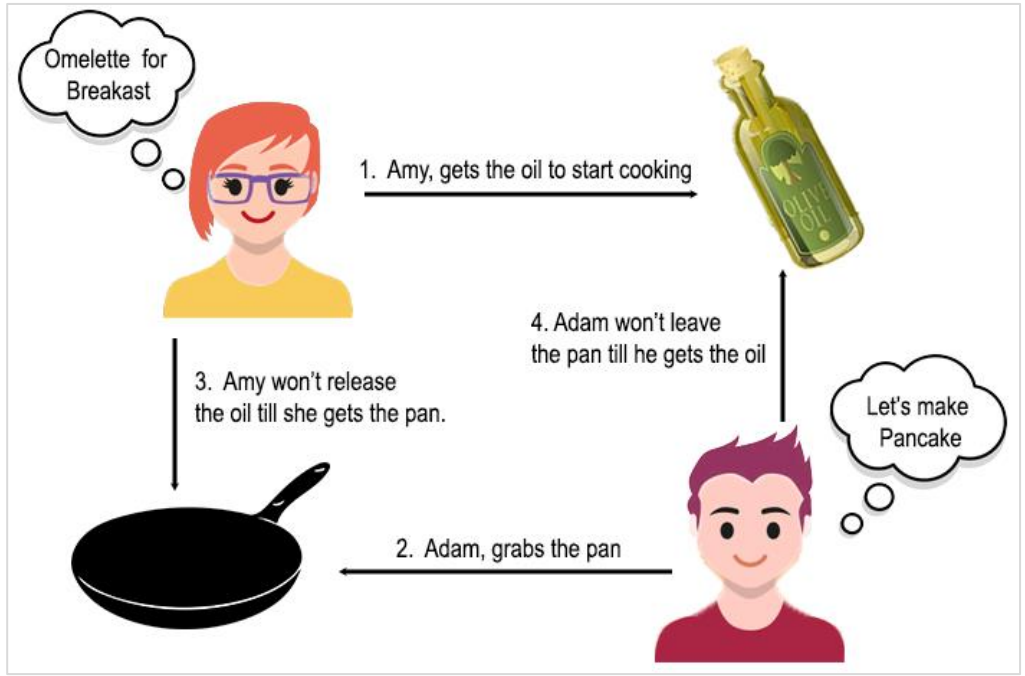


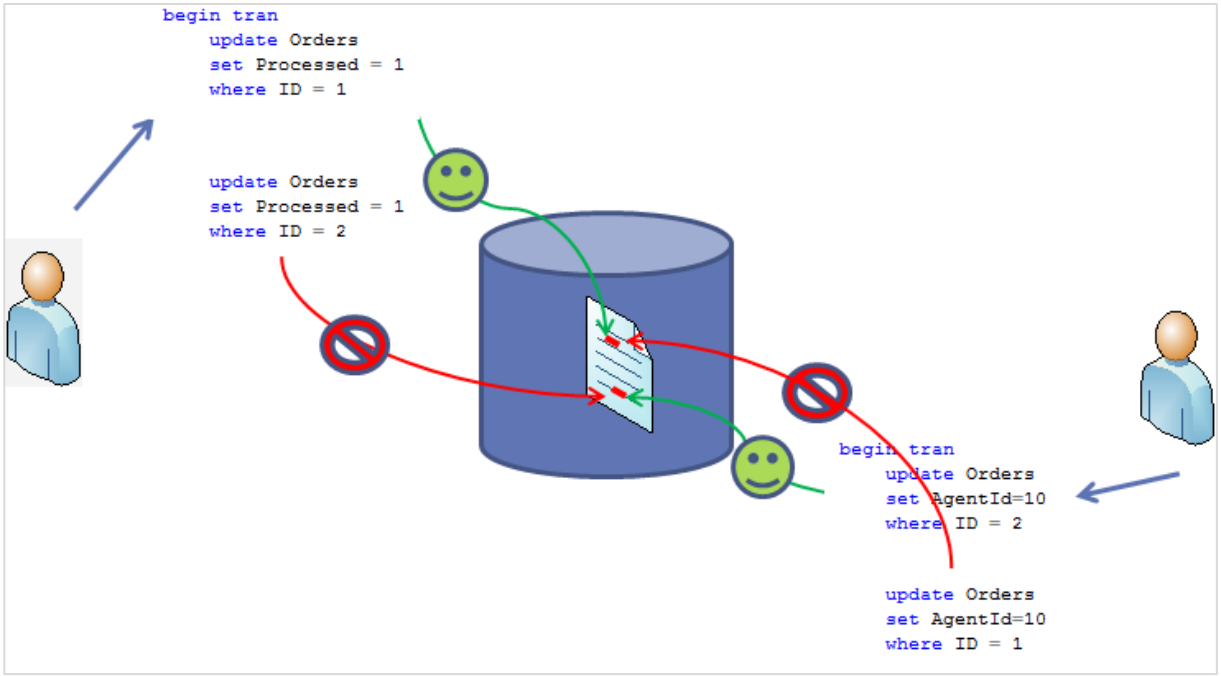
See below for the waiting circle



死锁的四大必要条件

- **Hold-and-Wait**
  - 进程已占有一部分资源，并请求更多资源





资源分配图

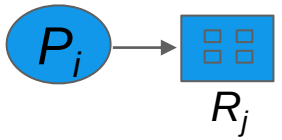
资源分配图中的重要元素

- 节点（2类）：
  - 进程节点 & 资源节点

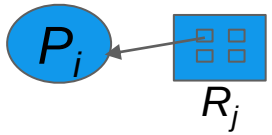
Process

Resource Type
(with 4 instances)

- 边：
  - 资源请求边
  - 资源分配边



Request edge



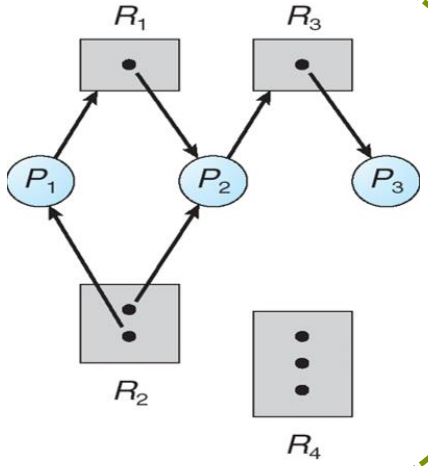
Assignment edge

用资源分配图来刻画任意时刻进程与资源的关系

资源分配图

- 死锁的现象，本质上都可以归结到资源分配不当问题
- 资源分配图：可以用来为死锁进行建模

资源分配图示例：



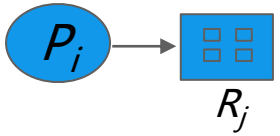
资源分配图构成要素

- 节点：
  - 分为2类：进程节点与资源节点

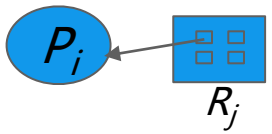
Process

Resource Type (with 4 instances)

- 边：
  - 资源请求边
  - 资源分配边

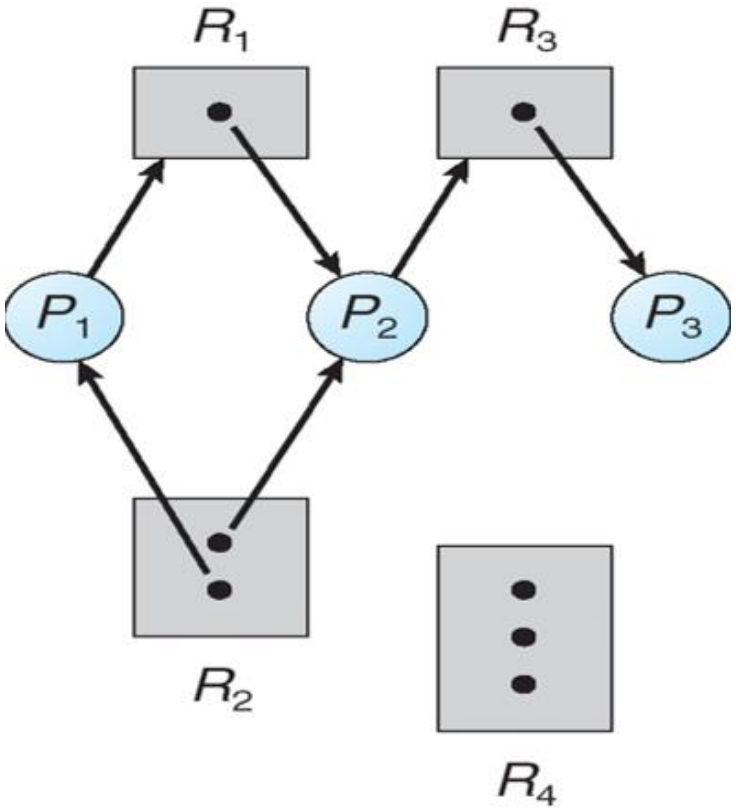


Request edge



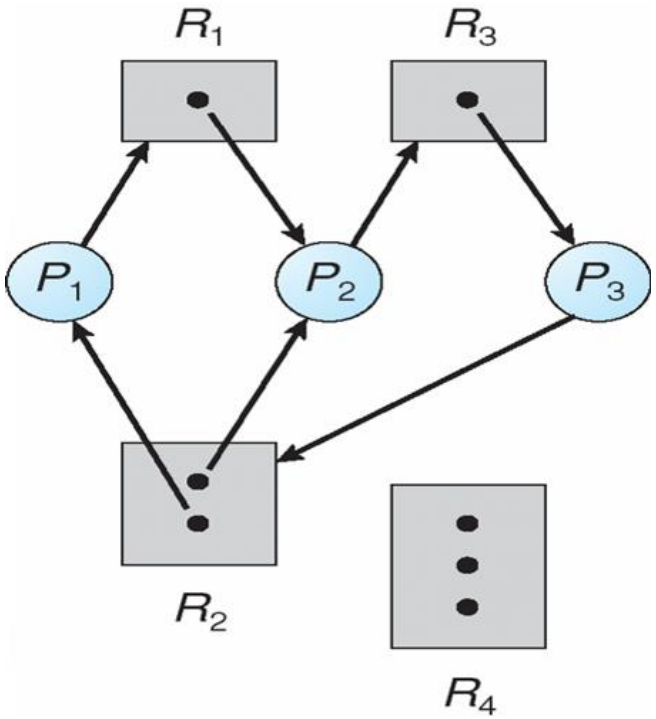
Assignment edge

- 资源分配图示意图1：无环

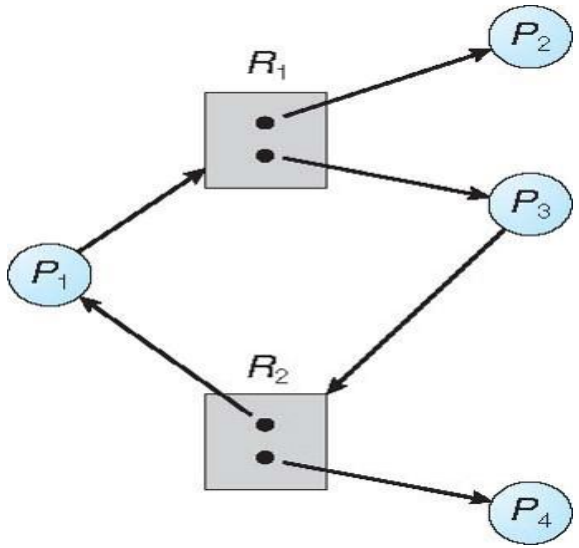


基于资源分配图的死锁分析

- 资源分配图示意图2：有环

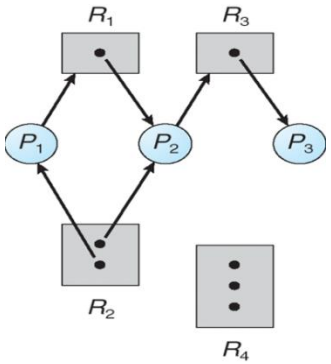


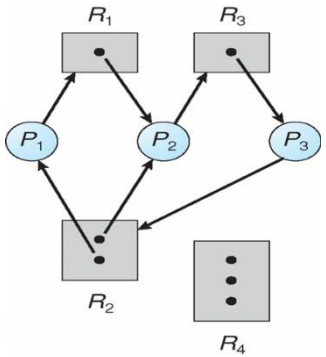
- 资源分配图示意图3：有环

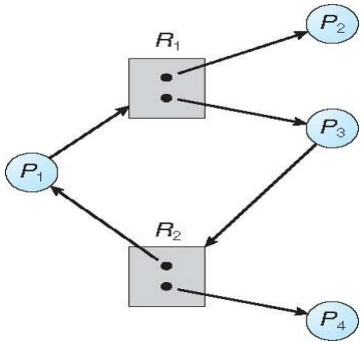


基于资源分配图的死锁分析

- 基于对资源分配图的示例分析，可以得出怎样的结论？







基于资源分配图的死锁分析

- 基于对资源分配图的示例分析，可以得出怎样的结论?

If graph contains no cycles => no deadlock

If graph contains a cycle =>

- If only one instance per resource type, then deadlock

  如果每一类资源都仅有一个，那么成环必定产生死锁

- If several instances per resource type, possibility of deadlock

  如果每类资源可能有多个，那么成环只是有可能产生死锁

Summary

小结：

- 死锁基本概念
- 死锁必要条件
- 资源建模

哲学家i

Chopstick[5] = {1,1,1,1,1}

```
do {
        P(chopstick[i])
        P(chopstick[(i+1) % 5])
                ...
              eat
                ...
        V(chopstick[i]);
        V(chopstick[(i+1) % 5]);
                ...
             think
                ...
} while (1);
```

从进程并发调度的角度，分析形成死锁的情景

哲学家i

Chopstick[5] = {1,1,1,1,1}

```
do {
        P(chopstick[i])
        P(chopstick[(i+1) % 5])
                ...
              eat
                ...
        V(chopstick[i]);
        V(chopstick[(i+1) % 5]);
                ...
             think
                ...
} while (1);
```

如何处理该问题中的死锁问题?

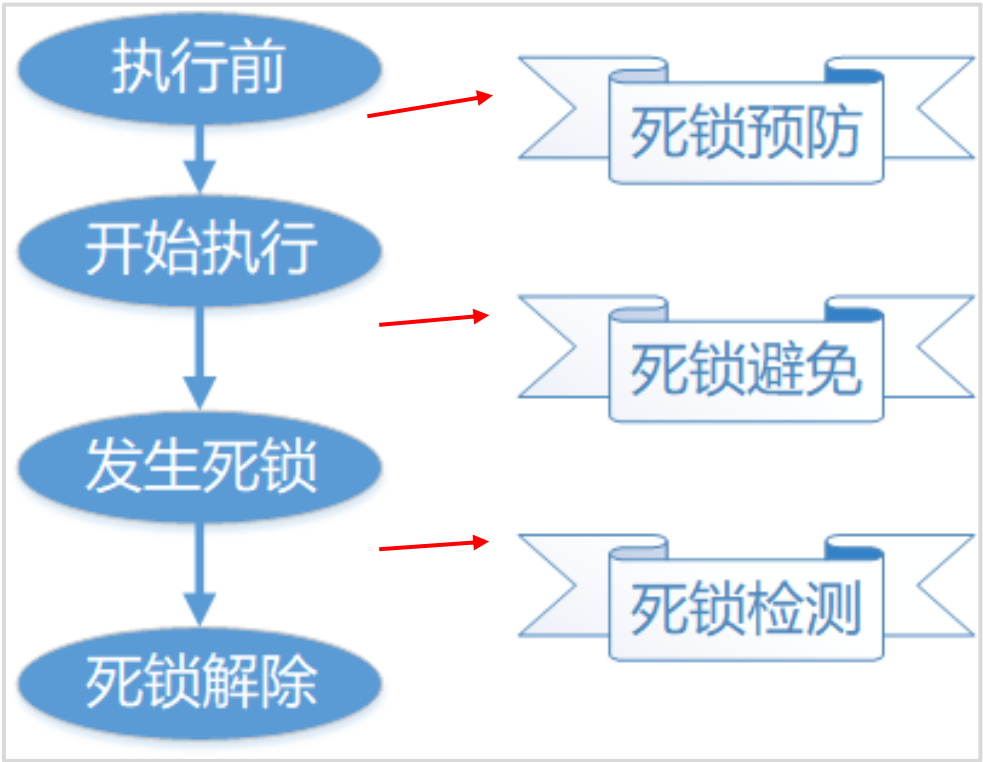
哲学家就餐问题

哲学家i

Chopstick[5] = {1,1,1,1,1}

```
do {
        P(chopstick[i])
        P(chopstick[(i+1) % 5])
            …
          eat
            …
        V(chopstick[i]);
        V(chopstick[(i+1) % 5]);
            …
         think
            …
} while (1);
```

如何处理该问题中的死锁问题?



死锁练习题 1

某系统中只有11台打印机，N个进程共享打印机，每个进程要求3台，当N取值不超过（ ）时，系统不会发生死锁?

死锁练习题 2

在（ ）的情况下，系统出现死锁。

A. 计算机系统发生了重大故障
B. 有多个竞争资源的进程同时存在
C. 若干个进程因竞争资源而无休止地相互等待他方释放已占有的资源
D. 资源数大大小于进程数或进程同时申请的资源数大大超过资源总数

死锁练习题 3

当进程个数大于资源数时，进程竞争资源（　）会发生死锁。

A. 一定
B. 不一定

死锁练习题 4

某系统中有3个并发进程，都需要同类资源4个，问该系统不会发生死锁的最少资源数是（ ）。

A. 8
B. 9
C. 10
D. 11

死锁练习题 5

以下有关资源分配图的描述中正确的是（ ）。

A. 有向边包括进程指向资源类的分配边和资源类指向进程申请边两类
B. 矩形框表示进程，其中圆点表示申请同一类资源的各个进程
C. 圆圈节点表示资源类
D. 资源分配图是一个有向图，用于表示某时刻系统资源与进程之间的状态

死锁练习题 6

系统的资源分配图在下列情况中，无法判断是否处于死锁的情况有（ ）。
I. 出现了环路
II．没有环路
III．每种资源只有一个，并出现环路
IV．每个进程节点至少有一条请求边

A. I、II、III、IV
B. I、III、IV
C. I、IV
D. 以上答案都不正确

现实代码中的死锁问题

现实软件代码中的死锁问题

谢 谢!
Thank you!