# 操作系统复习

第七章： 死锁

第七章总纲：

* System Model
* Deadlock Characterization
* Methods for Handling Deadlocks
* Deadlock Prevention
* Deadlock Avoidance
* Deadlock Detection
* Recovery from Deadlock

**#系统模型**

死锁的定义：一组进程中，每个进程都无限等待被该组进程中另一进程所占有的资源，因而永远无法得到资源，这种现象称为**进程死锁**，这组进程称为**死锁进程**。

正常情况下，进程只允许按如下顺序使用资源：

1. 申请：如果申请不能被允许，会一直等待。

2. 使用：进程对资源进行操作。

3. 释放：进程释放资源。

**#死锁特征**

**\*死锁的必要条件：**

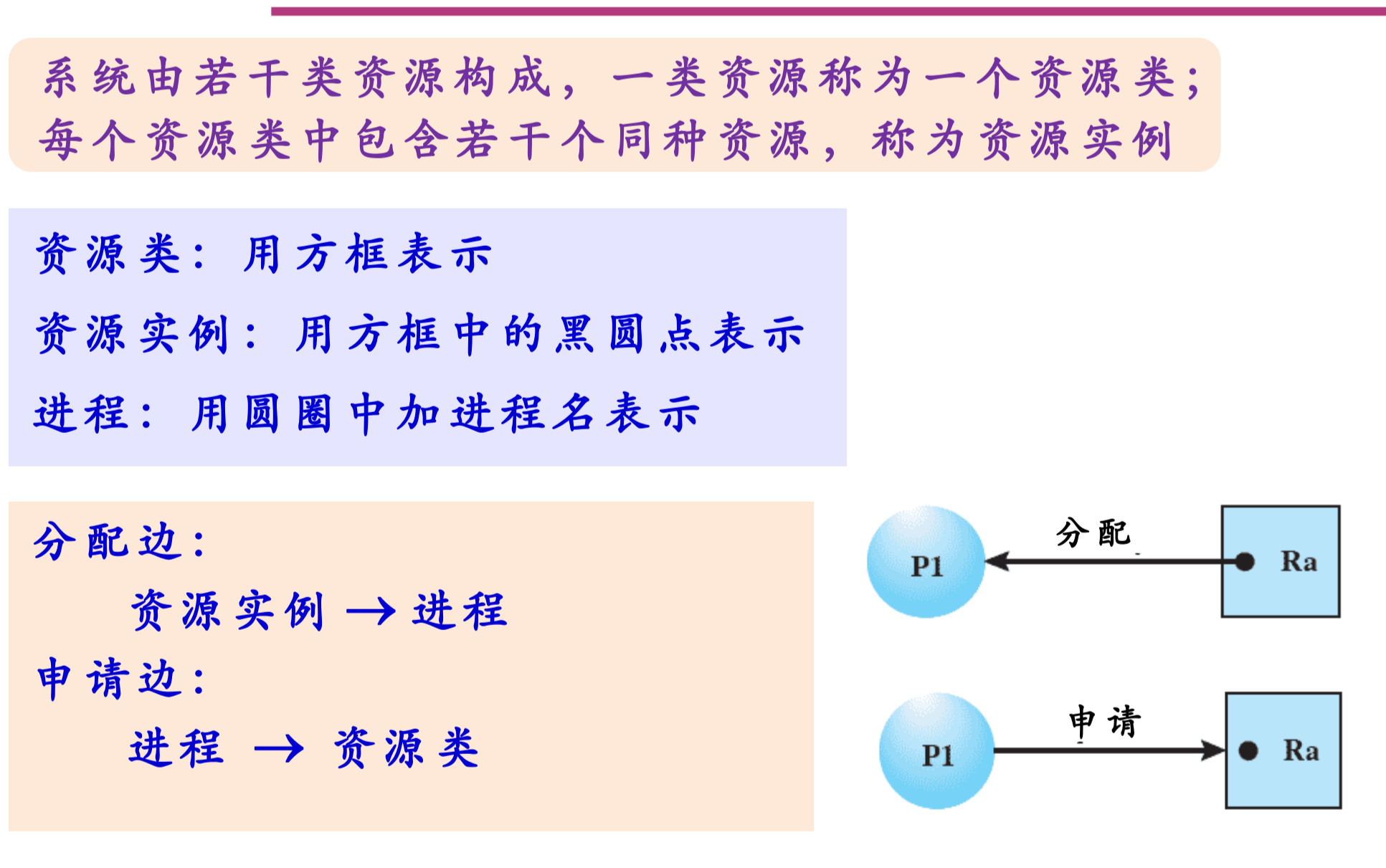
**互斥Mutual exclusion**: 在同一时间内，一个资源只能被一个资源占用。

**占有并等待Hold and wait**: 一个进程必须占有一个资源并等待另一个被其他进程占有的资源。

**非抢占No preemption**: 资源不能被抢占，智能被释放。

**循环等待Circular wait**: 如字面意思。

**\*资源分配图**



**#死锁处理方法**

总的来说三种方法：

* 1. 可是有协议以预防或避免死锁。
  2. 可允许系统进入死锁状态，然后检测它，并加以恢复。
  3. 可忽视这个问题，认为死锁不可能在系统发生。

**#死锁预防**

1. 互斥：这个条件无法修改，必须保证互斥。
2. 占有并等待：可以修改为，当一个进程申请一个资源时，它不能占有其他资源。但这样会导致资源利用率过低，或者发生饥饿。
3. 非抢占：可改为，当一个进程占有资源并申请另一个不能立即分配的资源，它占有的资源被视为可抢占。
4. 循环等待：可改为，对所有资源类型进行排序，且要求每个进程按递增顺序来申请资源。

**#死锁避免**

以不让死锁发生为目标，跟踪并评估资源分配过程，根据评估结果决定是否分配。

如果系统能按某个顺序为每个进程分配资源，并能避免死锁，那么系统就是安全的。更为准确地说，如果存在一个安全序列，那么系统处于安全状态。

1. 资源分配图算法

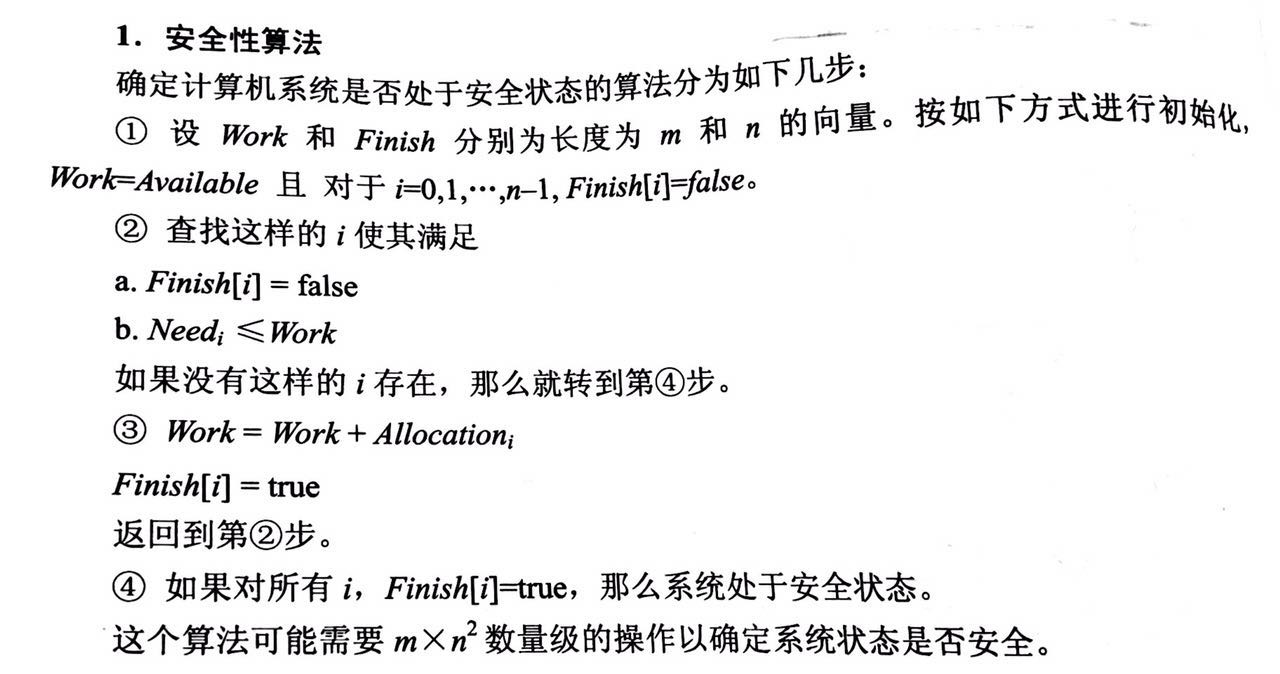
环检测

**2. 银行家算法**

* **Available**: 长度为m的向量，表示每种资源的现有实例的数量。
* **Max**：n\*m的矩阵，定义每个进程的最大需求。Max[i][j]=k，表示进程Pi最多可申请到k个资源类型的Rj实例。
* **Allocation**：n\*m的矩阵， 定义每个进程现在所分配的各种资源类型的实例数量。
* **Need**：n\*m的矩阵，表示每个进程还需要的神域的资源。

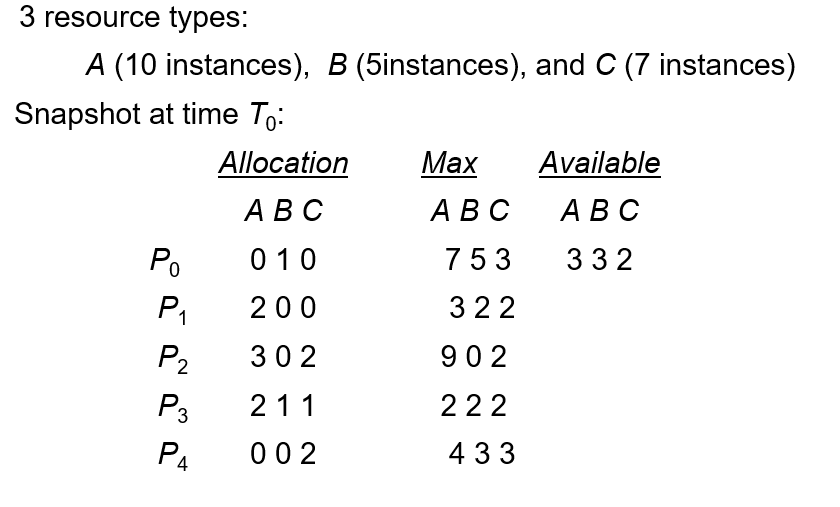
Need=Max-Allocation

**安全性算法：**

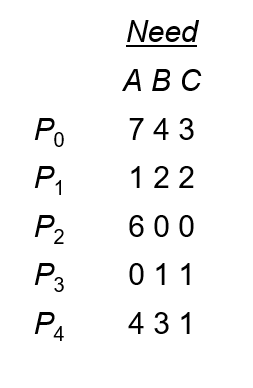


**资源请求算法：**按照请求的资源数目对应分配给矩阵，然后使用安全性算法判断是否安全。

例题：



第一步找到Need矩阵，Need=Max-Allocation



第二步，另Work初始化为Available，比较Work和Need各行的值找到Work>=Need，如这里应该第一个找到P1。

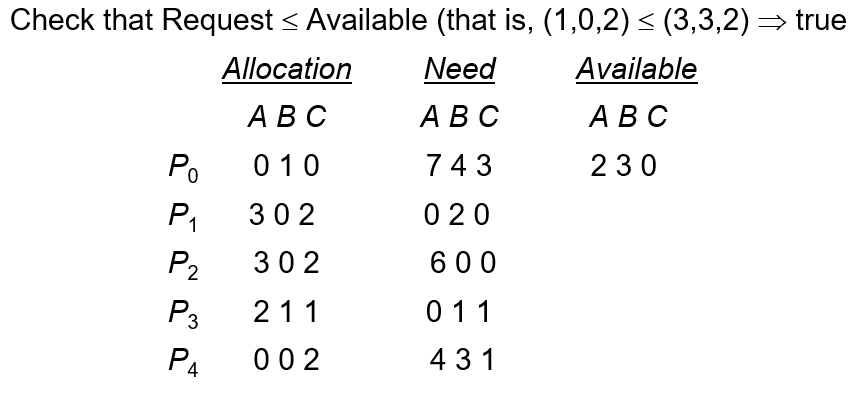
第三步，更新Work，Work=上面那步找到的i行的Allocation+work。

第四步，返回第二步直到把所有i都找到，如果找到则为安全状态，如果找不到则为不安全状态。

》》》》资源分配

假设此时P1请求1个A资源，2个C资源，即Request1=（1，0，2），请问是否可以满足？

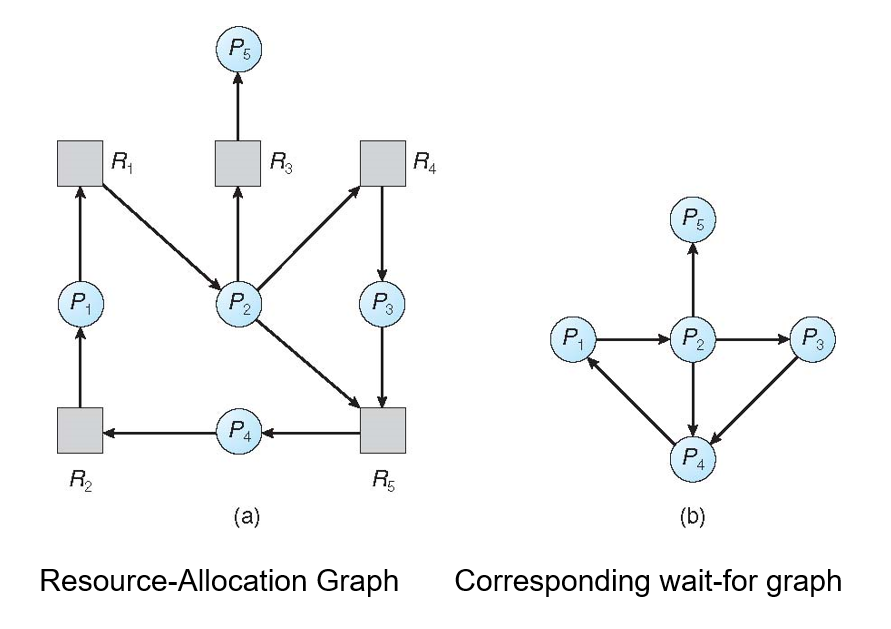
第一步，预分配，Allocation+Request，Available-Request，Need-Request，



再使用安全性算法，计算出序列《1，3，4，0，2》满足。

**#死锁检测**

资源分配图到等待图的转化



安全性算法检测

**#死锁恢复**

