进程死锁

主讲教师：赵霞

死锁

- 目的与要求
  - 了解死锁的定义
  - 掌握死锁预防
  - 了解死锁避免、死锁检测、死锁恢复的方法
- 重点与难点
  - 死锁预防法则的使用
  - 死锁避免和检测算法
- 作业
  - 4.26，　4.31
  - 制作本章思维导图

课后阅读与思考

- 教材
  - 第4章
- Operating System Concepts (6th edition)
  - Chapter 8 Deadlock
- Modern Operating System (2nd edition)
  - Chapter 3

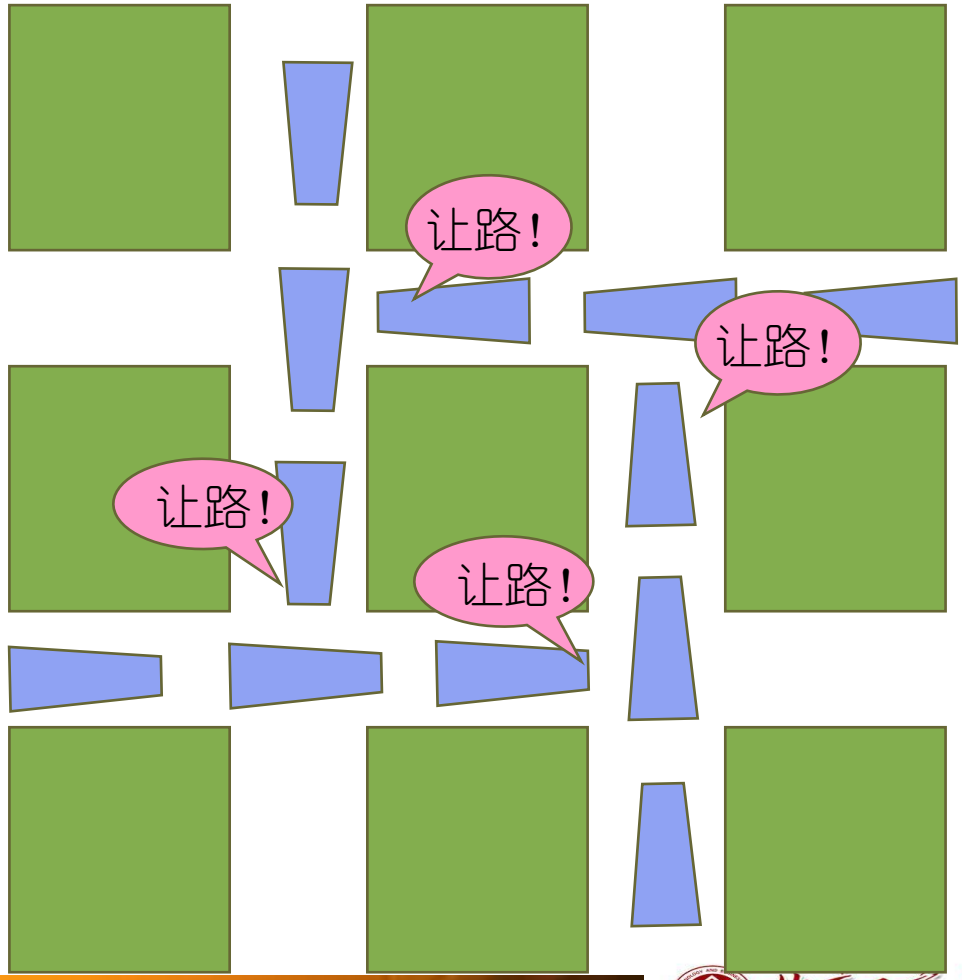
死锁

- 死锁示例
- 死锁定义
- 死锁防止
- 死锁避免
- 死锁检测
- 死锁的恢复
- 死锁综合处理


11455

死锁示例

- 交通死锁



竞争外设死锁示例

进程A　　进程B

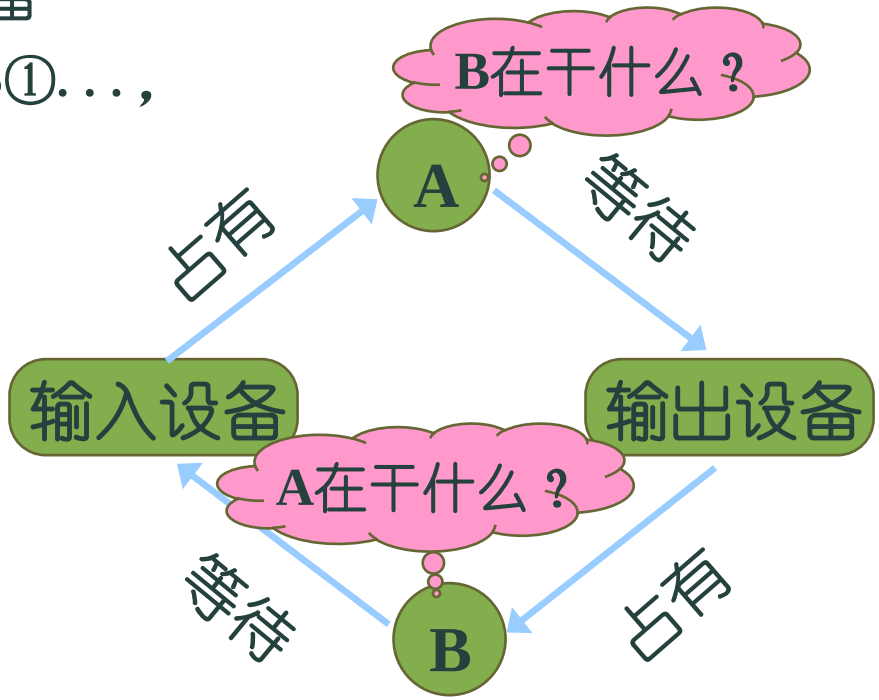
①申请输入设备　①申请输出设备

②申请输出设备　②申请输入设备

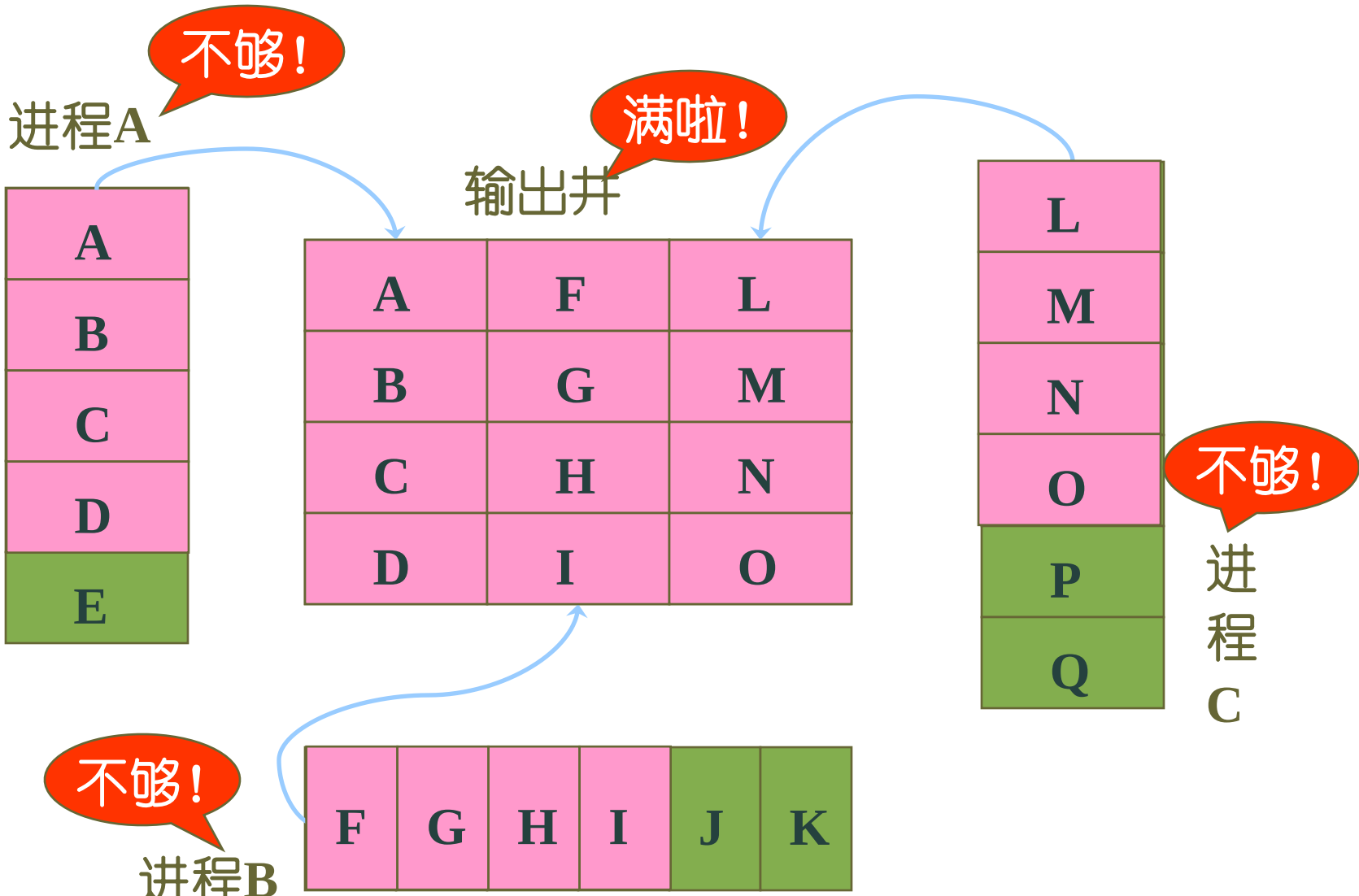
③释放输入设备　③释放输出设备

④释放输出设备　④释放输入设备

如果执行次序为：进程A①→进程B①...，则发生死锁。



SPOOLing系统死锁示例



死锁定义

- 定义1
  - 在一个进程集合中，若每个进程都在等待某些事件（指：释放资源）的发生，而这些事件又必须由这个进程集合中的某些进程来产生，就称该进程集合处于死锁状态
- 定义2
  - 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，因而永远无法得到的资源，这种现象称为死锁，这一组进程就称为死锁进程

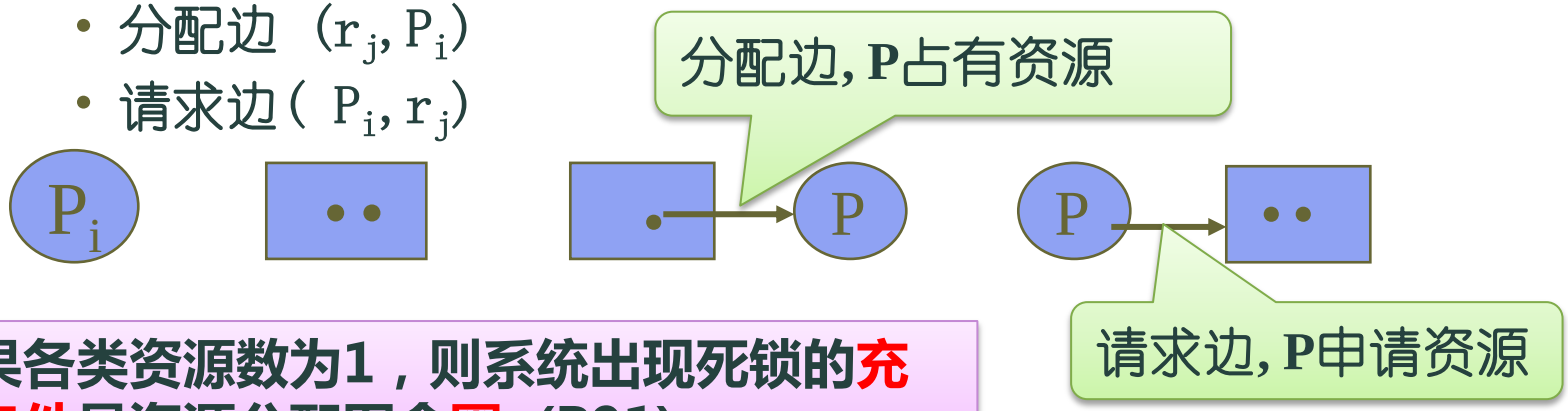
死锁的四个必要条件

- 互斥（资源独占）
  - 一个资源每次只能给一个进程使用
- 占有等待
  - 一个进程在申请新的资源的同时保持对原有资源的占有
- 非剥夺
  - 资源申请者不能强行的从资源占有者手中夺取资源，资源只能由占有者自愿释放
- 循环等待
  - 存在一个进程等待队列 $\{P_1, P_2, \cdots, P_n\}$，其中 P_1等待P_2占有的资源，P_2等待P_3占有的资源，…，P_n等待P_1占有的资源，形成一个进程等待环路

资源分配图定义

形式化描述

- 资源分配图 G=(V, E)，其中
- 顶点集V=P∪R,
  - P是进程集合： $P=\{P_1, P_2, \cdots, P_n\}$
  - 资源集$R=\{r_1, r_2, \cdots, r_m\}$，$r_i$表示系统中的$r_i$类资源
- 资源的数目用方框中的圆点表示
- E是边的集合，E中有两类边
  - 分配边 (r_j, P_i)
  - 请求边 (P_i, r_j)



- **如果各类资源数为1，则系统出现死锁的充要条件是资源分配图含圈 (P91)**

解决死锁问题的方法

- 设计无死锁的系统
  - 死锁防止
    - 在应用编程时或资源分配管理设计时破坏死锁的必要条件
  - 死锁避免
    - 进行资源分配时，判断是否存在安全序列，存在情况下进行分配，否则拒绝分配
- 允许死锁、出现排除
  - 死锁检测
  - 死锁恢复

死锁的防止

死锁的逻辑公式：

$$D \rightarrow C1 \wedge C2 \wedge C3 \wedge C4$$

$$\neg C1 \vee \neg C2 \vee \neg C3 \vee \neg C4 \rightarrow \neg D$$

在应用编程时或资源分配管理设计时破坏死锁存在的必要条件

死锁的防止

- 破坏互斥占用条件
  - 让资源共享使用
- 破坏占有等待条件
  - 将进程所要资源一次性分给进程：资源预分配
    - 要么不分，要么全部满足
    - 适合廉价资源的分配，如外存空间分配
  - 进程在下一轮申请资源时，释放所占的所有资源
    - 用完一个再用下一个

死锁的防止

- 破坏非剥夺条件
  - 用于内存管理、CPU管理等
  - 方法1
    - 当进程P_i申请r_j类资源时，若有则分配，若没有则回收P_i占有的所有资源；
  - 方法2
    - 当进程P_i申请r_j类资源时，若P_k占用r_j类资源
    - 若P_k处于等资源状态，则剥夺其资源，分配给P_i；
    - 若P_k 不处于等资源状态，则回收P_i的资源，设置其为等待资源状态

死锁的防止

- 破坏循环等待条件
  - 采用资源顺序分配方法
  - 给每类资源编号，进程只能按序号由小到大的顺序申请资源
  - 若不满足则拒绝分配

死锁避免

- 银行家算法
  - 在系统处理资源申请时，判断在满足申请时，系统是否还处于安全状态？是则满足本次资源申请 ，否则拒绝
- 单类资源：系统安全状态定义
  - 设系统中有n个进程，若存在一个序列〈P_1， P_2…P_n〉使得P_i(i＝1，2…n)以后还需要的资源可以通过系统现有资源加上所有P_j(j＜i)占有的资源来满足 ，则称这个系统处于安全状态
  - 序列〈P_1， P_2，…，P_n〉称为安全序列

举例

- 设银行家有10万贷款，P,Q,R分别需要8,3,9万元搞项目（假设任何人满足资金总额后都会归还所有贷款）
  - 如果P已申请到了4万，Q申请2万，R申请2万，则银行家还剩2万
  - 此时，如果Q又申请1万
  - 显然，如果满足Q的申请，有安全序列〈QPR〉
  - 但如果R要申请4万，显然，如果满足R的申请，则不存在安全序列

扩展的银行家算法描述

- n：系统中的进程个数
- m：系统中的资源类型数
- Available(1:m):现有资源向量
  - Available(j)=k表示有k个未分配的j类资源
- Max(1:n,1:m):资源最大申请量矩阵
  - Max(i,j)=k表示第i个进程对第j类资源的最大申请量为k
- Allocation(1:n,1:m):资源分配矩阵
  - Allocation(i,j)=k表示进程i已占有k个j类资源
- Need(1:n,1:m):进程以后还需要的资源矩阵
  - Need(i,j)=k表示第i个进程以后还需要k个第j类资源
  - 显然Need=Max-Allocation
- Request(1:n,1:m):进程申请资源矩阵
  - Request(i,j)=k表示进程i申请k个第j类资源

资源分配程序的工作过程

- 记A_i为A(i,1),A(i,2)…A(i,m)
  - 其中A为n×m矩阵
- 定义长度为m的向量 X、Y间的关系为
  - X≤Y当且仅当X(i)≤Y(i)(i=1,2…m)

- 当进程提出资源申请时
  - 系统首先检查
    - 该进程对资源的申请量是否超过其最大需求量
    - 系统现有资源能否满足进程需要
  - 若能则进一步检查
    - 若把资源分给该进程系统能否处于安全状态
    - 若安全则分配，否则置该进程为等待资源状态

资源分配程序的工作过程

- 设进程i申请资源j，申请资源向量为$Request_{i,j}$，则有如下的资源分配过程
  - 如果$Request_{i,j}$>$Need_{i,j}$则报错返回
  - 如果$Request_{i,j}$ >$Available_j$,则进程i进入等待资源状态，返回
  - 假设进程i的申请已获准，于是修改系统状态
    - $Available_j$ =$Available_j$- $Request_{i,j}$
    - $Allocation_{i,j}$= $Allocation_{i,j}$+ $Request_{i,j}$
    - $Need_{i,j}$= $Need_{i,j}$ - $Request_{i,j}$
  - 调用安全状态检查算法
    - 若系统处于安全状态，则将进程i申请的资源分配给进程i，返回
    - 若系统处于不安全状态，则进程i进入等待资源状态，并恢复系统状态后返回

安全状态检查算法

- 设Work(1:m)为临时工作向量
  - 初始时 Work = Available，所有可用资源的集合
  - 进程号 集合 ={1，2…n}
- (1) 寻找进程 i∈N 使其满足$Need_i \leqslant Work$
  - 若不存在这样的 i 则转(3)
  - 若找到，则表明能把$Need_i$ 分配给i进程，i进程能执行完；
- (2) Work = Work + Allocation ，N =N-{i}， 转(1)
  - 把分给i进程的所有资源收回，继续判断下一进程的需求是否能满足
- (3)
  - 如果 N 为空则返回(系统安全)
  - 如果 N 不为空则返回(系统不安全)

举例

	Allocation	Max	Request	Available
P1	1 2 4	2 5 8	1 2 1	1 3 3
P2	0 3 3	4 4 4	3 0 0	
P3	4 1 1	5 4 4	1 2 2	

- 如果p1先申请，无安全序列
- 如果p3申请，可有安全序列<p3, p2, p1>或<p3, p1, p2>

死锁检测

- 不做死锁避免，定期检测死锁是否发生
- 采用化简资源分配图的方法可以检测系统中有无进程处于死锁状态
- 资源分配图的简化过程
  - (1) 在图中找一个进程顶点 P_i，P_i的请求边均能立即满足
  - (2)
    - 若找到这样的P_i则将与 P_i相连的边全部删去，转(1)；
    - 否则化简过程结束
  - 如果化简后所有的进程顶点都成了孤立点，则称该图可完全化简；否则称该图是不可完全化简的
- 系统中有死锁的充分必要条件是资源分配图不可完全化简
  - 参见 P96

死锁恢复

- 检测出死锁后的处理
  - 破坏循环等待
  - 杀掉有关进程
  - 或使某个进程退回 ， 使系统摆脱死锁的状态

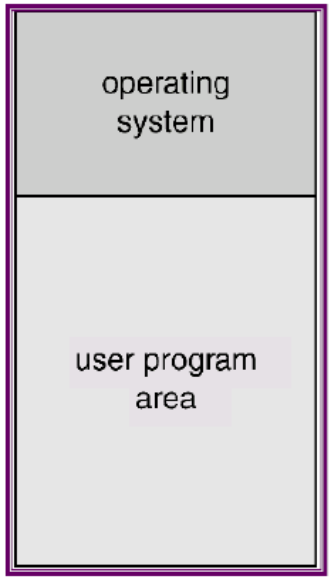
死锁的综合处理

- 把系统中的资源分成几大类，整体上采用资源顺序分配法
- 对每类资源根据其特点选择最适合的方法

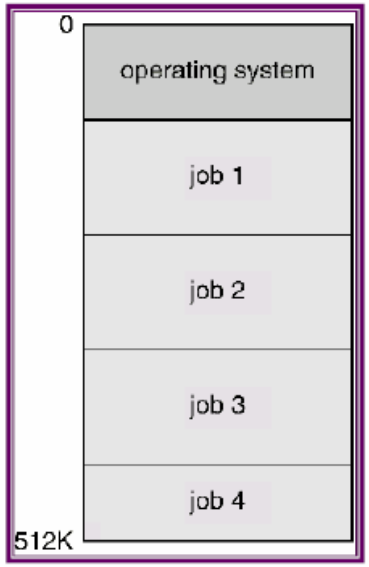
- 例如：
  - 主存、处理机 —— 剥夺法
  - 辅存 —— 预分配法
  - 其他 —— 死锁检测

预习作业

- 下面两张图分别是两种操作系统的物理内存布局图。
- 请你发挥想象，画出第三张图，表示出当代操作系统（例如Linux）所采用的虚拟页式管理方式下的内存布局图
- 可以参考任何资料，复杂度和详细程度不限
- 用ppt画图，给出尽可能清晰的说明文字



简单批处理OS



多任务批处理OS

?

多任务交互OS
虚拟页式管理


Q&A