ch8 死锁

死锁问题

- 原因

  资源竞争

  还取决于各进程推进的速度和对资源请求的顺序

- 例子

系统模型

死锁特点

1. 资源互斥
2. 占有并等待
3. 非抢占
4. 循环等待

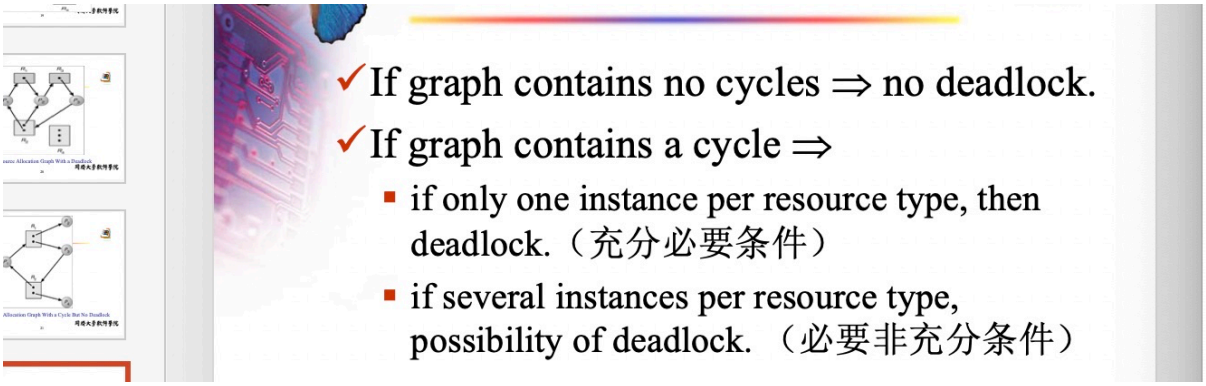
资源分配图

过程：申请资源，画申请边 -> 申请得到满足，变成分配边 -> 结束使用，释放资源，删除边

没有环的图一定不死锁

有环的图可能死锁 可能不死锁

具体要看环涉及到的资源类型中资源实例的数量



解决死锁的方法

1. 避免死锁。 -> 死锁预防、死锁避免
2. 允许死锁，然后恢复 -> 死锁检测、死锁恢复
3. 忽视问题，假装思索没有发生

死锁预防

只要四个必要条件中其中给一个不满足

- 互斥

  共享资源不要求互斥访问

- 占有并等待

  保证进程请求一个资源时，它不能占有其他资源

  协议一：一次性申请所有资源

  协议二：申请新资源时必须释放已有资源

  缺点：1. 资源利用率较低 2. 可能发生饥饿

- 非抢占

  允许抢占资源

  重新执行时，恢复被抢占的资源

- 循环等待

  对资源进行完全排序

  进程只按照递增需申请资源

死锁预防的缺点：1. 设备利用率低 2. 系统吞吐率低

死锁避免

依靠算法

比死锁预防算法要起低，只需要了解进程使用资源的情况

- 有关概念

  死锁避免：进程提供**如何申请资源**的**附加信息**，如每种资源实例的最大需求，构造**算法**确保系统不会进入死锁状态

  资源分配状态：

  - 安全状态 一定不会死锁
  - 不安全状态 可能会死锁

  安全序列：能避免死锁的，为每个进程分配资源的顺序

  避免方法：确保系统不会进入不安全状态

  两个算法：1. 资源分配图算法 2. 银行家算法

  P187 例题

- **资源分配图算法**

  需求边：将来某时刻申请资源。虚线表示。

  循环检测算法 -> 检测图中是否有环(n^2级，n是进程数量)

  如果没有环，则在安全状态；如果有环，则Pi需要等待其资源申请

- ~~银行家算法~~

死锁检测

如果上面两种方式都不采用，就有可能死锁。这种情况下，系统应该提供：

1. 检测是否出现死锁的算法
2. 从死锁中恢复的算法

检测算法：

1. **等待图**（只适用于每种资源一个实例的情况）

   基于资源分配图，删除所有资源节点，合并边。

   死锁 == 等待图中有一个环

   检测死锁的方法：维护等待图，周期性的对图搜索（n^2级别，n是节点数）

2. 检测算法

死锁恢复

- 进程终止

  通过终止进程收回进程的资源，注意如果进程更新了文件，要重新设置文件

  - 全部终止
  - 部分终止

    一次终止一个直至取消死锁

    每次选择代价最小的进程，策略选择问题

- 资源抢占

  三个问题：

  - 选择一个牺牲品
  - 回滚
  - 饥饿

    如果依靠代价选择牺牲品，可能导致总是某个进程被回滚，饥饿。可以设置最大回滚次数

组合方式

组合预防、避免、检测三种方式