
进程管理
死锁
死锁的概念
死锁的定义
多个进程因竞争资源而等待对方手里的资源，互相等待，无法向前推进
死锁产生的原因
1.系统资源的竞争
2.进程推进顺序非法，请求和释放资源的顺序不当
3.信号量的使用不当
对不可剥夺资源的不合理分配、请求
死锁产生的必要条件
互斥条件
对互斥资源的争抢才会导致死锁
不剥夺条件
进程获得的资源未使用完前，不能被其它进程强行夺走，只能主动释放
请求并保持条件
进程至少保持一个资源，又提出新的被其它进程占用资源请求，此时被阻塞，但又不放弃自己资源
循环等待条件
存在一种进程资源的循环等待链
循环等待未必死锁，死锁一定有循环等待
若同类资源数>1不会死锁
死锁、饥饿、死循环的区别
死锁
至少两个进程一起死锁，一定处于阻塞态
饥饿
可以只有一个进程饥饿，饥饿进程可能阻塞(等IO设备)可能就绪态(长期得不到处理机)
操作系统(管理者)问题
死循环
可以只有一个进程发生死循环，死循环进程可以上CPU运行(处于运行态)，由代码逻辑的错误导致
被管理者(程序)的问题
死锁的处理策略
预防死锁
避免死锁
死锁的检测及解除
死锁预防
1.破坏互斥条件
方案
把只能互斥使用的资源改造为允许共享使用，如SPOOLing
缺点
适用情况少 可行性低，(为了系统安全必须保护这种互斥性)
2.破坏不剥夺条件
方案
方案1.主动释放，新请求不满足释放所有资源
方案2.操作系统协助剥夺，考虑进程优先级
缺点
实现复杂，释放已获得的资源可能造成前一阶段工作失效
反复地申请和释放资源会增加系统开销，降低系统吞吐量
方案1中主动释放资源可能也会一直发生，导致饥饿
3.破坏请求并保持条件
方案
静态分配法，运行前一次性申请完所有资源，且一直拥有；运行后不再请求新的资源
缺点
有些资源可能只需要保持很短时间，运行期间如果一直资源浪费，利用率低，导致饥饿
4.破坏循环等待条件
方案
给资源编号，必须按编号从小到大的顺序一次申请完资源
缺点
不方便增加新设备，顺序不一致而浪费资源，用户编程麻烦
死锁避免
系统安全状态与死锁的关系
当系统进入不安全状态后，可能进入死锁状态
不安全状态未必死锁
只要系统处于不安全状态，系统必不会出现死锁
找到一个安全序列，系统就安全
安全序列
如果系统按照这种序列分配资源，则每个进程都能顺利完成。
银行家算法
数据结构
可用资源向量Available[m]
最大需求矩阵Max[i][j]
分配矩阵Allocation[i][j]
需求矩阵Need[i][j]
算法步骤
1.检查此次请求，是否超过之前剩余的需求量Need
Request <= Need
2.检查次数系统剩余的可用资源，是否能满足这次需求
Request <= Available
3.试着分配，更新各个矩阵
更新Avaiable、Allocation、Need
4.用安全性算法检查，此次分配是否会导致系统进入不安全状态
安全性算法
手算
能满足的一次性把拥有的资源全部归还，加入安全序列，重复至终
算法步骤
1.初始安全序列为空，工作向量work = 银行家算法本轮第三个步骤中尝试分配后的Available
work = Available
2.检查当前剩余可用资源，是否能满足某一个进程的最大需求，如果可以就把该进程加入安全序列，并把该进程持有的资源全部回收。
回收：work += Allocation
3.不断重复上述步骤，看最终是否能让所有进程都加入安全序列
死锁的检测和解除
如何检测
数据结构：资源分配图
两种节点
进程节点
资源节点
两种边
进程节点->资源节点 (请求边)
资源节点->进程节点 (分配边)
死锁检测算法
依次消除与不阻塞进程相连的边，直到无边可消
不阻塞进程：申请资源数还足够
死锁定理：若资源分配图不可完全简化，说明剩余进程发生死锁
如何解除
资源剥夺法
撤销进程法
进程回退法
一旦发生立即解除
进程优先级、已执行时间、剩余执行时间、拥有资源数、交互/批处理
考虑进程的因素