Ch 11 死锁

1. 死锁的基本概念

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**进程死锁**，这一组进程就称为**死锁进程**。

如果死锁发生，会浪费大量系统资源，甚至系统崩溃

- 参与死锁的所有进程都在等待资源
- 参与死锁的进程是当前系统中所有进程的子集

出现死锁的原因：**资源数量有限、锁和信号量使用错误**

资源的使用方式：申请—分配—使用—释放

可重用资源：可以被多个进程多次使用

- **可抢占资源**与**不可抢占资源**
- 处理器、I/O 部件、内存、文件、数据库、信号量

可消耗资源：只可使用一次的可以创建和销毁的资源

- 信号、中断、消息

活锁与**饥饿**

活锁

- 加锁
- 轮询
- 没有进展也没有阻塞

饥饿

- 资源分配策略决定

产生死锁的必要条件

- **互斥使用**（资源独占）
    - 一个资源每次只能给一个进程使用
- **占有且等待**（请求和保持，部分分配）
    - 进程在申请新的资源的同时保持对原有资源的占有
- **不可抢占**（不可剥夺）
    - 资源申请者不能强行从资源占有者手中夺取资源，只能由占有者自愿释放
- **循环等待**
    - 存在一个进程等待队列

2. 资源分配图

资源分配图（RAG）

用有向图描述系统资源和进程的状态

二元组 $G=(V,E)$

V : 结点集合，分为 P(进程)、R(资源) 两部分

E : 有向边的集合，其元素为有序二元组

系统由若干类资源构成，一类资源称为一个资源类；每个资源类中包含若干个同种资源，称为资源实例

资源类：用方框表示

资源实例：用方框中的实心黑圆点表示

进程：用圆圈中加进程名表示

分配边：

- 资源实例 → 进程的一条有向边

申请边：

- 进程 → 资源类的一条有向边

死锁定理

- 如果资源分配图中没有环路，则系统中没有死锁，如果图中存在环路，则系统中可能存在死锁
- 如果每个资源类中**只包含一个资源实例**，则**环路是死锁存在的充分必要条件**

化简：

1. 找一个非孤立点进程结点且只有分配边，去掉分配边，将其变为孤立结点
2. 再把相应的资源分配给一个等待该资源的进程，即将某进程的申请边变为分配边

3. 解决死锁

解决死锁的方法

- 不考虑：鸵鸟算法
- **不让死锁发生**
    - **死锁预防**

      静态策略：设计合适的资源分配算法，不让死锁发生
    - **死锁避免**

      动态策略：以不让死锁发生为目标，跟踪并评估资源分配过程，根据评估结果决策是否分配
- **让死锁发生**
    - **死锁检测与解除**

4. 死锁预防

在设计系统时，通过确定资源分配算法，**排除产生死锁的可能性**

具体的做法是：**防止产生死锁的四个条件中任何一个发生**

4.1 破坏“互斥使用/资源独占”条件

资源转换技术：把独占资源变为共享资源

4.2 破坏“占有且等待”条件

实现方案 1：要求每个进程在运行前必须一次性申请它所要求的所有资源，且仅当该进程所要资源均可满足时才给予一次性分配

问题：资源利用率低，“饥饿”现象

实现方案 2：在允许进程动态申请资源前提下规定，一个进程在申请新的资源不能立即得到满足而变为等待状态之前，必须释放已占有的全部资源，若需要再重新申请

4.3 破坏“不可抢占”条件

实现方案：虚拟化资源

- 当一个进程申请的资源被其他进程占用时，可以通过操作系统抢占这一资源（两个进程优先级不同）
- 局限性：适用于状态易于保存和恢复的资源

4.4 破坏“循环等待”条件

通过定义资源类型的线形顺序实现

实施方案：资源有序分配法

- 把系统中所有资源编号，进程在申请资源时必须严格按资源编号的递增次序进行，否则操作系统不予分配

5. 死锁避免

在系统运行过程中，对进程发出的每一个系统能够满足的资源申请进行动态检查，并根据检查结果决定是否分配资源，若分配后系统发生死锁或可能产生死锁，则不予分配，否则予以分配。

安全状态：如果存在一个由系统中所有进程构成的安全序列，则系统处于安全状态。

安全序列

一个进程 $\{P_1, \cdots, P_n\}$ 是安全的，如果对于每一个进程 $P_i (1 \leq i \leq n)$，它以后尚需要的资源量不超过系统当前剩余资源量与所有进程当前占有资源量之和，系统处于安全状态。

安全状态一定没有死锁发生

不安全状态：不存在一个安全序列

不安全状态一定会导致死锁

6. 银行家算法

系统具有的特征：

1. 在固定数量的进程中共享数量固定的资源
2. 每个进程预先指定完成工作所需的最大资源数量
3. 进程不能申请比系统中可用资源总数还多的资源
4. 进程等待资源的时间是有限的
5. 如果系统满足了进程对资源的最大需求，那么进程应该在有限的时间内使用资源，并将资源归还给系统

7. 死锁的检测与解除

死锁检测

- 允许死锁发生，操作系统不断监视系统进展情况，判断死锁是否发生
- 一旦死锁发生则采取专门的措施，解除死锁并以最小的代价恢复操作系统运行

检测时机：

- 当进程由于资源请求不满足而等待时检测死锁
- 定时检测
- 系统资源利用率下降时检测死锁

死锁解除

以最小的代价恢复系统的运行

1. 撤销所有死锁进程
2. 进程回退再启动
3. 按照某种原则逐一撤销死锁进程
4. 按照某种原则逐一抢占资源

8. 哲学家就餐问题