并发性：死锁和饥饿

死锁

- 可重用资源
- 可消费资源
- **死锁条件**
  - 互斥 ---》必要条件
  - 占有且等待 ---》必要条件
  - 非抢占 ---》必要条件
  - 循环等待 ---》充分条件
- **预防死锁**：打破上述四个条件，过于严苛，一般不采取
  - 互斥：不可能禁止
  - 占有且等待
    - 进程一次性申请所需要的资源
    - 缺点
      - 阻塞时间长
      - 进程无法事先知道所需要的所有资源
  - 非抢占
    - 释放已经占用的资源
    - 其他进程可以抢占资源
    - 缺点
      - 进程执行可能出错
  - 循环等待
    - 资源类型偏序化
- **避免死锁**
  - 进程启动拒绝
    - 当资源总量>=新开进程需要的资源量+已经启动的进程需要的资源量
  - 资源请求拒绝/银行家算法
    - **资源的申请导致从安全状态转为非安全状态，则会拒绝此次申请**
    - **安全状态**：若在某一时刻，系统能按某种进程顺序，如$\{P_1, P_2, \cdots, P_n\}$，为每个进程分配其所需的资源，直至最大需求，使每个进程均可顺利完成，则称此时系统的状态为安全状态，称这样的一个进程序列$\{P_1, P_2, \cdots, P_n\}$为安全序列。
    - **非安全状态**：若在某一时刻,系统中不存在一个安全序列,则称系统处于不安全状态。
    - 用到的数据结构
      - `matrix C`：n*m，n为进程数，m为资源种类数，表示每个进程对于各种资源的需求量
      - `matrix A`：n*m 表示每个进程已经分配得到的资源量
      - `vector R`：每种资源的总量
      - `vector V`：每种资源没有被分配的量

        - 缺点：
            - 必须事先声明每个进程请求的最大资源
            - 所考虑的进程无关
            - 分配的资源数目固定
            - 占有资源时，进程不能退出
- **死锁检测**
    - 操作系统周期性地检查**循环等待**条件
    - ```
      eg:
          R1  R2  R3  R4  R5              R1  R2  R3  R4  R5     R1  R2  R3
      R4  R5
      P1  0   1   0   0   1          P1  1   0   1   1   0      2   1   1
      2   1
      P2  0   0   1   0   1          P2  1   1   0   0   0        Resource
      Vector R
      P3  0   0   0   0   1          P2  0   0   0   1   0      R1  R2  R3
      R4  R5
      P4  1   0   1   0   1          P4  0   0   0   0   0      0   0   0
      0   1
          Request matrix Q             Allocation matrix A       Available
      Vector V
           （还需要申请的资源）

      1. 首先P4没有申请资源，因此没有发生死锁
      2. P3还需要申请1个R5，可以被剩余资源V满足，因此没有发生死锁
      3. 将P3的资源释放出来，V=[0,0,0,0,1]+[0,0,0,1,0]=[0,0,0,1,1]
      4. P1 和 P2 还需要申请的资源无法被剩余资源V=[0,0,0,1,1]满足，P1和P2发生死锁
      ```

- **死锁恢复**
    - 终止进程
        - 终止所有进程
        - 连续终止进程直到不再存在死锁
    - 抢占资源：**需要设置检查点，以使得资源被抢占的进程回退到前面获得该资源的位置**
        - 连续抢占资源直到不再存在死锁
    - 根据最小代价原则进行选择

哲学家就餐问题
      ```