银行家算法

1. 背景
   1. 多个实例
   2. 每个进程必须事先声明使用的最大量
   3. 当一个进程请求资源，它可能要等待
   4. 当一个进程得到所有的资源，它必须在有限的时间释放它们
2. 数据结构（n为进程的数目，m为资源类型的数目）
   1. Available（available[j]=k表示资源Rj有k个有效实例）
   2. Max（max[i,j]=k表示进程Pi最多可请求k个Rj）
   3. Allocation（allocation[i,j]=k表示进程Pi已经占有了k个Rj）
   4. Need（need[i,j]=k表示进程Pi还需要k个Rj）

Need[i,j] = max[i,j] - allocation[i,j]

1. 过程
   1. 进程i提出请求，request[i,j]=k表明进程i请求k个Rj
   2. 若该进程的request＞need，出错中断
   3. 若对该资源的request＞available，进程i等待
   4. 同时满足request≤need和request≤available的话，系统试着按照request分配资源 => available = available - request

Allocation = allocation + request

Need = need – request

* 1. 用安全算法进行检查，看系统是否处于安全状态

若安全则真正分配，否则回到原状态。

1. 安全算法
   1. 定义Work和finish（每种资源对应一个work，表示先在系统中该种资源可用的实例数目，work[j]=available[j]）（每个进程对应一个finish，表示该进程是否执行结束）
   2. 找进程i，满足①finish[i] = false （该进程尚未完成）

②need i ≤ work （该进程的要求是系统可以满足的）

若找到：work = work + allocation i

Finish[i] = true

再去找进程i

若没找到，检查所有finish[i]，如果全为true，则现在系统状态是安全的。