学号：2016011962

姓名：朱家琪

**死锁与饥饿：**

**死锁：**指一组进程中的每一个进程无限期等待被该组进程中的另一个进程所占有且永远不会释放的资源。

**饥饿：**指系统不能保证进程等待时间上界的现象。即进程总是在等待被调度却得不到调度的机会，导致进程一直无法执行。

（以上定义均来自网络）

通俗地说，**死锁就像**彼此握有对方人质的一些国家，每个国家都不敢首先行动，都在等待其他国家的下一步动作；**饥饿就像**被动等待雨水的植物，如果天不下雨，那么植物也只能一直等待直到枯死也毫无办法。

**银行家算法：**

（1）根据题意可得，在时刻资源的分配情况如下图所示：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 资源  进 程 情况 | Max | Allocation | Need | Available |
| A B C | A B C | A B C | A B C |
|  | 5 5 9 | 2 1 2 | 3 4 7 | 2 3 3 |
|  | 5 3 6 | 4 0 2 | 1 3 4 |  |
|  | 4 0 11 | 4 0 5 | 0 0 6 |  |
|  | 4 2 5 | 2 0 4 | 2 2 1 |  |
|  | 4 2 4 | 3 1 4 | 1 1 0 |  |

根据安全性算法可进行如下计算：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 资源  进程 情况 | Max | Need | Allocation | Work+Allocation | Finish |
| A B C | A B C | A B C | A B C |
|  | 4 2 5 | 2 2 1 | 2 0 4 | 4 3 7 | true |
|  | 4 2 4 | 1 1 0 | 3 1 4 | 7 4 11 | true |
|  | 5 5 9 | 3 4 7 | 2 1 2 | 9 5 13 | true |
|  | 5 3 6 | 1 3 4 | 4 0 2 | 13 5 15 | true |
|  | 4 0 11 | 0 0 6 | 4 0 5 | 17 5 20 | true |

故可得时刻为安全状态，求出的一个安全序列为{}

（2）对于：Request(0,3,4)能否分配:

①第一步：：Request(0,3,4)≤:Need(1,3,4)，满足条件

②第二步：：Request(0,3,4)≥Available(2,3,3)，这里不能满足分配条件，因此该请求不能被分配

（3）由于（2）中分配作废，恢复了原来的资源分配状态，因此此时系统状态与时刻相同。

对于：Request(2,0,1)能否被分配：

①第一步：：Request(2,0,1)≤:Need(2,2,1)，满足条件

②第二步：：Request(2,0,1)≤Available(2,3,3)，满足条件

③第二步：系统试探着把资源分配给进程：

此时Available从(2,3,3)变为(0,3,2)

:Allocation从(2,0,4)变为(4,0,5)

:Need从(2,2,1)变为(0,2,0)

④第四步：系统执行安全性算法，检查此次资源分配后系统是否处于安全状态。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 资源进程 情况 | Max | Need | Allocation | Work+Allocation | Finish |
| A B C | A B C | A B C | A B C |
|  | 4 2 5 | 0 2 0 | 4 0 5 | 4 3 7 | true |
|  | 4 2 4 | 1 1 0 | 3 1 4 | 7 4 11 | true |
|  | 5 5 9 | 3 4 7 | 2 1 2 | 9 5 13 | true |
|  | 5 3 6 | 1 3 4 | 4 0 2 | 13 5 15 | true |
|  | 4 0 11 | 0 0 6 | 4 0 5 | 17 5 20 | true |

可得此次资源分配后系统仍然处于安全状态，因此正式按③中的资源进行分配。

（4）在（3）的分配资源后基础上，对：Request(0,2,0)能否分配进行检查：

①第一步：：Request(0,2,0)≤：Need(3,4,7)，满足条件

②第二步：：Request(0,2,0)≤Available(0,3,2)，满足条件

③第三步：系统试探着把资源分配给进程：

此时Available从(0,3,2)变为(0,1,2)

:Allocation从(2,1,2)变为(2,3,2)

:Need从(3,4,7)变为(3,2,7)

④第四步：系统执行安全性算法，检查此次资源分配后系统是否处于安全状态。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 资源  进程 情况 | Max | Need | Allocation | Work+Allocation | Finish |
| A B C | A B C | A B C | A B C |
|  |  |  |  |  | False |

由于此次资源分配后系统不处于安全状态，因为本次试探分配作废，恢复原来的资源分配状态，让进程等待。