银行家算法

有三类资源A(17)、B(5)、C(20)，有5个进程P1-P5。T0时刻系统状态如下：

|  |  |  |
| --- | --- | --- |
|  | 最大需求 | 已分配 |
| P1 | 5 5 9 | 2 1 2 |
| P2 | 5 3 6 | 4 0 2 |
| P3 | 4 0 11 | 4 0 5 |
| P4 | 4 2 5 | 2 0 4 |
| P5 | 4 2 4 | 3 1 4 |

(1)T0时刻是否为安全状态，给出安全系列。

(2)T0时刻，P2：Request(0,3,4)，能否分配，为什么？

(3)在(2)的基础上P4：Request(2,0,1)，能否分配，为什么？

(4)在(3)的基础上P1：Request(0,2,0)，能否分配，为什么？

解：(1)

T0时刻的资源分配情况：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 进程  资源情况 | Max | Allocation | Need | Available |
| A B C | A B C | A B C | A B C |
| P1 | 5 5 9 | 2 1 2 | 3 4 7 | 2 3 3 |
| P2 | 5 3 6 | 4 0 2 | 1 3 4 |  |
| P3 | 4 0 11 | 4 0 5 | 0 0 6 |  |
| P4 | 4 2 5 | 2 0 4 | 2 2 1 |  |
| P5 | 4 2 4 | 3 1 4 | 1 1 0 |  |

T0时刻的安全性，根据已分配和A(17),B(5),C(20)可以求出work = (2 3 3)

利用安全性算法对T0时刻的资源分配情况进行分析(如下表)可知，在T0时刻存在着一个安全序列{P4,P2,P5,P3,P1}，故系统是安全的。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 进程  资源情况 | Work | Need | Allocation | Work+Allocation | Finish |
| A B C | A B C | A B C | A B C |
| P4 | 2 3 3 | 2 2 1 | 2 0 4 | 4 3 7 | True |
| P2 | 4 3 7 | 1 3 4 | 4 0 2 | 8 3 9 | True |
| P5 | 8 3 9 | 1 1 0 | 3 1 4 | 11 4 13 | True |
| P3 | 11 4 13 | 0 0 6 | 4 0 5 | 15 4 18 | True |
| P1 | 15 4 18 | 3 4 7 | 2 1 2 | 17 5 20 | True |

(2)P2请求资源：P2发出请求向量Request(0,3,4)，系统按银行家算法进行检查：

①Request2(0,3,4)<=Need2(1,3,4);

②Request2(0,3,4)>Available(2,3,3)，无足够资源，让P2等待，故系统不分配资源。

(3)P4请求资源：P4发出请求向量Request(2,0,1)，系统按银行家算法进行检查：

①Request4(2,0,1)<=Need4(2,2,1);

②Request4(2,0,1)<=Available(2,3,3);

更新资源分配情况：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 进程  资源情况 | Max | Allocation | Need | Available |
| A B C | A B C | A B C | A B C |
| P1 | 5 5 9 | 2 1 2 | 3 4 7 | 0 3 2 |
| P2 | 5 3 6 | 4 0 2 | 1 3 4 |  |
| P3 | 4 0 11 | 4 0 5 | 0 0 6 |  |
| P4 | 4 2 5 | 4 0 5 | 0 2 0 |  |
| P5 | 4 2 4 | 3 1 4 | 1 1 0 |  |

利用安全性算法对更新后资源分配情况进行分析(如下表)可知，更新后存在着一个安全序列{P4,P2,P3,P5,P1}，故系统是安全的。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 进程  资源情况 | Work | Need | Allocation | Work+Allocation | Finish |
| A B C | A B C | A B C | A B C |
| P4 | 0 3 2 | 0 2 0 | 4 0 5 | 4 3 7 | True |
| P2 | 4 3 7 | 1 3 4 | 4 0 2 | 8 3 9 | True |
| P3 | 8 3 9 | 0 0 6 | 4 0 5 | 12 3 14 | True |
| P5 | 12 3 14 | 1 1 0 | 3 1 4 | 15 4 18 | True |
| P1 | 15 4 18 | 3 4 7 | 2 1 2 | 17 5 20 | True |

(4)P1请求资源：P1发出请求向量Request(2,0,1)，系统按银行家算法进行检查：

①Request1(0,2,0)<=Need1(3,4,7);

②Request1(0,2,0)<=Available(0,3,2);

更新资源分配情况：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 进程  资源情况 | Max | Allocation | Need | Available |
| A B C | A B C | A B C | A B C |
| P1 | 5 5 9 | 2 3 2 | 3 2 7 | 0 1 2 |
| P2 | 5 3 6 | 4 0 2 | 1 3 4 |  |
| P3 | 4 0 11 | 4 0 5 | 0 0 6 |  |
| P4 | 4 2 5 | 4 0 5 | 0 2 0 |  |
| P5 | 4 2 4 | 3 1 4 | 1 1 0 |  |

work=(0,1,2)

此时，P1-P5所有的 Need 都大于 work,已不能满足任何进程的需要,系统进入不安全状态，因此不能分配资源。