银行家算法



解：

1. T0时刻资源分配表：

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 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 |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 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 | T |
| P2 | 4 | 3 | 7 | 1 | 3 | 4 | 4 | 0 | 2 | 8 | 3 | 9 | T |
| P3 | 8 | 3 | 9 | 0 | 0 | 6 | 4 | 0 | 5 | 12 | 3 | 14 | T |
| P5 | 12 | 3 | 14 | 2 | 2 | 1 | 3 | 1 | 4 | 15 | 4 | 18 | T |
| P1 | 15 | 4 | 18 | 3 | 4 | 7 | 2 | 1 | 2 | 17 | 5 | 20 | T |

安全序列：P4-P2-P3-P5-P1，可以安全分配。

（2）P2:Request2(0,3,4)用银行家算法进行检查：

Request2(0,3,4)<=Need2(1,3,4)

Request2(0,3,4)>Available2(2,3,3)

P2等待，故不能完成分配。

（3）在（2）的基础上，P4:Request4(2,0,1)，用银行家算法进行检查

Request4(2,0,1)<=Need4(2,2,1)

Request2(2,0,1)<=Available(2,3,3)

系统先假设可以为P4分配资源，并修改表格中的值，由此形成的资源变化情况如下图：

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 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 |
|  |  |  |  |  |  |  |  |  |  | 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 | 2 | 0 | 4 | 2 | 2 | 1 |  |  |  |
|  |  |  |  | 4 | 0 | 5 | 0 | 2 | 0 |  |  |  |
| P5 | 4 | 2 | 4 | 3 | 1 | 4 | 1 | 1 | 0 |  |  |  |

检查是否安全：

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 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 | 2 | 2 | 1 | 3 | 1 | 4 | 15 | 4 | 18 | true |
| P1 | 15 | 4 | 18 | 3 | 4 | 7 | 2 | 1 | 2 | 17 | 5 | 20 | true |

安全序列：P4-P2-P3-P5-P1

此时处于安全状态，可以进行分配。

（4）由（3）可知Need1(3,4,7)、Available(0,3,2)

P1:Request1(0,2,0)，系统用银行家算法进行检查

Request1(0,2,0)<=Need1(3,4,7)

Request1(0,2,0)<=Available(0,3,2)

系统先假设可以为P1分配资源，并修改表格中的值，由此形成的资源变化情况如下图：

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 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 |
|  |  |  |  | 2 | 3 | 2 | 3 | 2 | 7 | 0 | 3 | 2 |
|  |  |  |  |  |  |  |  |  |  | 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 | 2 | 0 | 4 | 2 | 2 | 1 |  |  |  |
|  |  |  |  | 4 | 0 | 5 | 0 | 2 | 0 |  |  |  |
| P5 | 4 | 2 | 4 | 3 | 1 | 4 | 1 | 1 | 0 |  |  |  |

检查是否安全

此时的Available(0,1,2)无法满足任何一个进程，无法找到安全序列，故无法分配。