## 资源分配情况

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

答：

1. T0时刻为安全状态，安全序列之一为P4-P3-P1-P2-P5
2. 按银行家算法进行检查
3. request2(0,3,4) < need2(1,3,4)
4. request2(0,3,4) > available(2,3,3),不可分配
5. 按银行家算法进行检查
6. request4(2,0,1) < need4(2,2,1)
7. request4(2,0,1) < available(2,3,3)
8. 假设可分配，此时allocation4 = (4,0,5),need4 = (0,2,0),available = (0,3,2)
9. 此时再利用安全性算法进行检查：

此时系统为安全状态，安全序列之一为P4-P2-P3-P5-P1

1. 在（3）的基础上，按银行家算法进行检查
2. request1(0,2,0) < need1(3,4,7)
3. request1(0,2,0) < available(0,3,2)
4. 假设可分配，此时allocation1 = (2,3,2),need1 = (3,2,7),available = (0,1,2)
5. 此时再利用安全性算法进行检查，没有对应的安全序列，故不进行分配。