

1. T0时刻资源分配情况

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

T0时刻安全性：

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Work | | | Need | | | Allocation | | | Work+Allocation | | | Finish |
|  | A | B | C | A | B | C | A | B | C | A | B | C | true |
| 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 |
| 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 |

T0时刻存在安全序列P4->P2->P3->P5->P1，故系统是安全的

1. P2请求资源：P2发出请求向量Request2(0,3,4)，系统按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)，让P2等待

1. 在（2）的基础上，P4请求资源：P4发出请求向量Request4(2,0,1)，系统按银行家算法进行检查：

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

Request4(2,0,1)<=Available(2,3,3);

系统先假定可为P4分配资源，修改Available，Allocation4，Need4，因此资源变化为：

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

再利用安全性算法检查此时系统是否安全：

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Work | | | Need | | | Allocation | | | Work+Allocation | | | Finish |
|  | A | B | C | A | B | C | A | B | C | A | B | C | true |
| 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 |

存在安全序列P4->P2->P3->P5->P1，故系统是安全的，可以进行分配

1. 在（3）的基础上，P1请求资源：P1发出请求向量Request1(0,2,0)，系统按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);

系统先假定可为P4分配资源，修改Available，Allocation1，Need1，因此资源变化为：

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

进行安全检查：可用资源Available(0,1,2)已不能满足任何进程的需要，系统进入不安全状态，此时系统不分配资源