|  |  |  |  |
| --- | --- | --- | --- |
|  | 最大需求（Max） | 已分配（Allocation） | Need |
| P1 | 5 5 9 | 2 1 2 | 3 4 7 |
| 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 |

（1）

时刻资源分配：

在T0时刻，Work=Available = (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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Work | Need | Allocation | Work+Allocation | Finish |
| P4 | 2 3 3 | 3 4 7 | 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 | 2 2 1 | 3 1 4 | 15 4 18 | TRUE |
| P1 | 15 4 18 | 1 1 0 | 2 1 2 | 17 5 20 | TRUE |

所以，T0时刻为安全状态,存在安全序列为:{ P4 P2 P3 P5 P1}

（2）P2:Request(0,3,4) > work(2 3 3)

P2:Request(0,3,4) 时不能分配

（3）P4:Request(2,0,1) < work(2 3 3)

得到新的资源分配表：

|  |  |  |  |
| --- | --- | --- | --- |
|  | 最大需求（Max） | 已分配（Allocation） | Need |
| P1 | 5 5 9 | 2 1 2 | 3 4 7 |
| 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 |

用安全性算法对T0时刻的资源分配情况进行分析

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Work | Need | Allocation | Work+Allocation | Finish |
| 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 |

T0时刻为安全状态,存在安全序列为:{ P4 P2 P3 P5 P1}，

在P4：Request(2,0,1) 情况下，资源能分配。

（4）work更新为(0 3 2) P1:Request(0,2,0) < work(0 3 2)

新表：

|  |  |  |  |
| --- | --- | --- | --- |
|  | 最大需求（Max） | 已分配（Allocation） | Need |
| P1 | 5 5 9 | 2 3 2 | 3 2 7 |
| 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)，所有的 Need 都大于 work,不能满足任何进程

进入不安全状态，不能进行资源分配。