**题目**：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，给出安全序列。

(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？

**答：**（1）Need = Max - Allocation

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

根据已分配和A(17)、B(5)、C(20)，得，在T0时刻，Work=Available = (2 3 3)

分析：因为P4 Need < work ，所以，给P4分配,work = work + 已分配 = (4 3 7)

因为P2 Need < work ，所以，给P2分配,work = work + 已分配 = (8 3 9)

因为P3 Need < work ，所以，给P3分配,work = work + 已分配 = (12 3 14)

因为P5 Need < work ，所以，给P5分配,work = work + 已分配 = (15 4 18)

因为P1 Need < work ，所以，给P1分配,work = work + 已分配 = (17 5 20)

所以利用安全性算法对T0时刻的资源分配情况进行分析，有：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 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）因为P2:Request(0,3,4) > work(2 3 3)

所以，P2:Request(0,3,4) 时不能分配

（3）因为P4:Request(2,0,1) < work(2 3 3)

所以，更新资源分配表，为:

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

此时，同理，得 work = (0 3 2)

分析：因为P4 Need < work ，所以，给P4分配,work = work + 已分配 = (4 3 7)

因为P2 Need < work ，所以，给P2分配,work = work + 已分配 = (8 3 9)

因为P3 Need < work ，所以，给P3分配,work = work + 已分配 = (12 3 14)

因为P5 Need < work ，所以，给P5分配,work = work + 已分配 = (15 4 18)

因为P1 Need < work ，所以，给P1分配,work = work + 已分配 = (17 5 20)

所以利用安全性算法对T0时刻的资源分配情况进行分析，有：

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

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

即，在P4：Request(2,0,1) 得情况下，资源能分配。

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

更新资源分配表:

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

此时 work = (0 1 2)，所以P1-P5所有的 Need 都大于 work,已不能满足任何进程的需要,

则，系统进入不安全状态，因此不能进行资源分配。

2018013404

方可心