黄玟瑜
19335074
huangmy73@mail2.sysu.edu.cn

作业六

操作系统原理, 2021 春

2021-05-12

完成教材习题 6.3、6.5、6.6、6.15、6.18。

习题 6.3

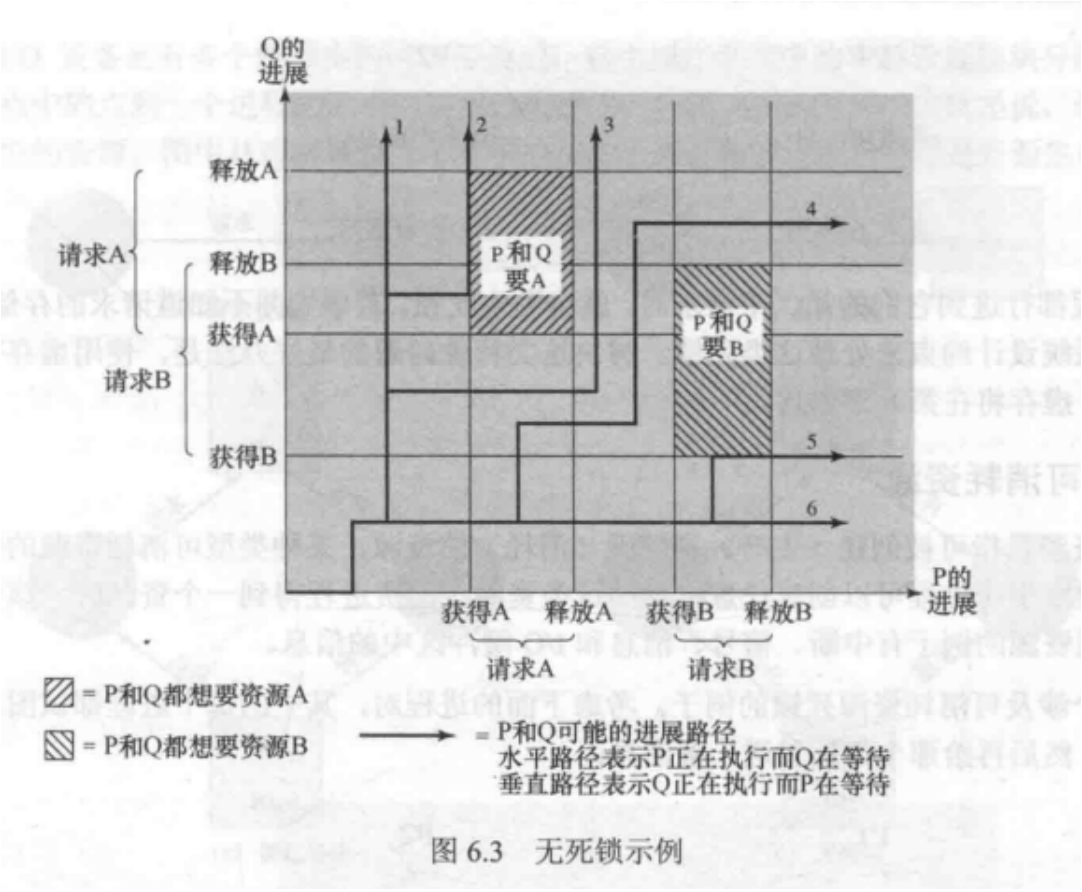


图 6.3　无死锁示例

1. Q 获得 B，然后获得 A；再后释放 B，然后释放 A；当 P 恢复执行时，它可以获得全部资源。

2. Q 获得 B，然后获得 A；P 执行并阻塞在对 A 的请求上；Q 释放 B，然后释放 A；当 P 恢复执行时，它可以获得全部资源。

3. Q 获得 B；然后 P 获得 A，再后 P 释放 A；然后 Q 获得 A；再后 Q 释放 B，然后释放 A；当 P 恢复执行时，它可以获得全部资源。

4. P 获得 A；然后 Q 获得 B; 再后 P 释放 A；然后 Q 获得 A；再后 Q 释放 B; 然后 P 获得 B，再后释放 B；最后 Q 再释放 A，故不会导致死锁。

5. P 获得 A，再后释放 A；然后 P 获得 B；Q 执行并阻塞在对 B 的请求上；P 释放 B；当 Q 恢复执行时，它可以获得全部资源。

6. P 获得 A，再后释放 A；然后获得 B，再后释放 B；当 Q 恢复执行时，它可以获得全部资源。

习题 6.5

在如下条件下考虑银行家算法。

6 个进程：P0～P5

4 种资源：A（15 单位），B（6 单位），C（9 单位），D（10 单位）

时间 T0 时的情况：

可用资源向量

A	B	C	D
6	3	5	4

	当前已分配				最大需求			
进程	A	B	C	D	A	B	C	D
P0	2	0	2	1	9	5	5	5
P1	0	1	1	1	2	2	3	3
P2	4	1	0	2	7	5	4	4
P3	1	0	0	1	3	3	3	2
P4	1	1	0	0	5	2	2	1
P5	1	0	1	1	4	4	4	4

a.

对资源 A，当前已分配 9 个单位，9（当前已分配）+6（可用）=15（资源 A 总量），正确；

对资源 B，当前已分配 3 个单位，3（当前已分配）+3（可用）=6（资源 B 总量），正确；

对资源 C，当前已分配 4 个单位，4（当前已分配）+5（可用）=9（资源 C 总量），正确；

对资源 D，当前已分配 6 个单位，6（当前已分配）+4（可用）=10（资源 D 总量），正确；

综上，题目中的可用资源向量是正确的。

b.

需求 = 最大需求-当前已分配

需求矩阵如下：

	A	B	C	D
P0	7	5	3	4
P1	2	1	2	2
P2	3	4	4	2
P3	2	3	3	1
P4	4	1	2	1
P5	3	4	3	3

c.

	A	B	C	D
P0	9	5	5	5
P1	2	2	3	3
P2	7	5	4	4
P3	3	3	3	2
P4	5	2	2	1
P5	4	4	4	4

最大需求

	A	B	C	D
P0	2	0	2	1
P1	0	1	1	1
P2	4	1	0	2
P3	1	0	0	1
P4	1	1	0	0
P5	1	0	1	1

当前已分配

	A	B	C	D
P0	7	5	3	4
P1	2	1	2	2
P2	3	4	4	2
P3	2	3	3	1
P4	4	1	2	1
P5	3	4	3	3

需求

A	B	C	D
15	6	9	10

资源总量

A	B	C	D
6	3	5	4

可用资源向量

初始状态

	A	B	C	D
P0	9	5	5	5
P1	0	0	0	0
P2	7	5	4	4
P3	3	3	3	2
P4	5	2	2	1
P5	4	4	4	4

最大需求

	A	B	C	D
P0	2	0	2	1
P1	0	0	0	0
P2	4	1	0	2
P3	1	0	0	1
P4	1	1	0	0
P5	1	0	1	1

当前已分配

	A	B	C	D
P0	7	5	3	4
P1	0	0	0	0
P2	3	4	4	2
P3	2	3	3	1
P4	4	1	2	1
P5	3	4	3	3

需求

A	B	C	D
15	6	9	10

资源总量

A	B	C	D
6	4	6	5

可用资源向量

P1 运行完成

	A	B	C	D
P0	9	5	5	5
P1	0	0	0	0
P2	0	0	0	0
P3	3	3	3	2
P4	5	2	2	1
P5	4	4	4	4

最大需求

	A	B	C	D
P0	2	0	2	1
P1	0	0	0	0
P2	0	0	0	0
P3	1	0	0	1
P4	1	1	0	0
P5	1	0	1	1

当前已分配

	A	B	C	D
P0	7	5	3	4
P1	0	0	0	0
P2	0	0	0	0
P3	2	3	3	1
P4	4	1	2	1
P5	3	4	3	3

需求

A	B	C	D
15	6	9	10

资源总量

A	B	C	D
10	5	6	7

可用资源向量

P2 运行完成

	A	B	C	D
P0	0	0	0	0
P1	0	0	0	0
P2	0	0	0	0
P3	3	3	3	2
P4	5	2	2	1
P5	4	4	4	4

最大需求

	A	B	C	D
P0	0	0	0	0
P1	0	0	0	0
P2	0	0	0	0
P3	1	0	0	1
P4	1	1	0	0
P5	1	0	1	1

当前已分配

	A	B	C	D
P0	0	0	0	0
P1	0	0	0	0
P2	0	0	0	0
P3	2	3	3	1
P4	4	1	2	1
P5	3	4	3	3

需求

A	B	C	D
15	6	9	10

资源总量

A	B	C	D
12	5	8	8

可用资源向量

P0 运行完成

	A	B	C	D
P0	0	0	0	0
P1	0	0	0	0
P2	0	0	0	0
P3	0	0	0	0
P4	5	2	2	1
P5	4	4	4	4

最大需求

	A	B	C	D
P0	0	0	0	0
P1	0	0	0	0
P2	0	0	0	0
P3	0	0	0	0
P4	1	1	0	0
P5	1	0	1	1

当前已分配

	A	B	C	D
P0	0	0	0	0
P1	0	0	0	0
P2	0	0	0	0
P3	0	0	0	0
P4	4	1	2	1
P5	3	4	3	3

需求

A	B	C	D
15	6	9	10

资源总量

A	B	C	D
13	5	8	9

可用资源向量

P3 运行完成

	A	B	C	D
P0	0	0	0	0
P1	0	0	0	0
P2	0	0	0	0
P3	0	0	0	0
P4	0	0	0	0
P5	4	4	4	4

最大需求

	A	B	C	D
P0	0	0	0	0
P1	0	0	0	0
P2	0	0	0	0
P3	0	0	0	0
P4	0	0	0	0
P5	1	0	1	1

当前已分配

	A	B	C	D
P0	0	0	0	0
P1	0	0	0	0
P2	0	0	0	0
P3	0	0	0	0
P4	0	0	0	0
P5	3	4	3	3

需求

A	B	C	D
15	6	9	10

资源总量

A	B	C	D
14	6	8	9

可用资源向量

P4 运行完成

	A	B	C	D
P0	0	0	0	0
P1	0	0	0	0
P2	0	0	0	0
P3	0	0	0	0
P4	0	0	0	0
P5	0	0	0	0

最大需求

	A	B	C	D
P0	0	0	0	0
P1	0	0	0	0
P2	0	0	0	0
P3	0	0	0	0
P4	0	0	0	0
P5	0	0	0	0

当前已分配

	A	B	C	D
P0	0	0	0	0
P1	0	0	0	0
P2	0	0	0	0
P3	0	0	0	0
P4	0	0	0	0
P5	0	0	0	0

需求

A	B	C	D
15	6	9	10

资源总量

A	B	C	D
15	6	9	10

可用资源向量

P5 运行完成

d. 假定同意该请求，则到达如下所示的状态，这个状态是不安全状态，可用资源无法满足任何一个进程，基于避免死锁原则，P5 的请求将被拒绝并且其将被阻塞。

	A	B	C	D
P0	9	5	5	5
P1	2	2	3	3
P2	7	5	4	4
P3	3	3	3	2
P4	5	2	2	1
P5	4	4	4	4

最大需求

	A	B	C	D
P0	2	0	2	1
P1	0	1	1	1
P2	4	1	0	2
P3	1	0	0	1
P4	1	1	0	0
P5	4	2	4	4

当前已分配

	A	B	C	D
P0	7	5	3	4
P1	2	1	2	2
P2	3	4	4	2
P3	2	3	3	1
P4	4	1	2	1
P5	3	4	3	3

需求

A	B	C	D
15	6	9	10

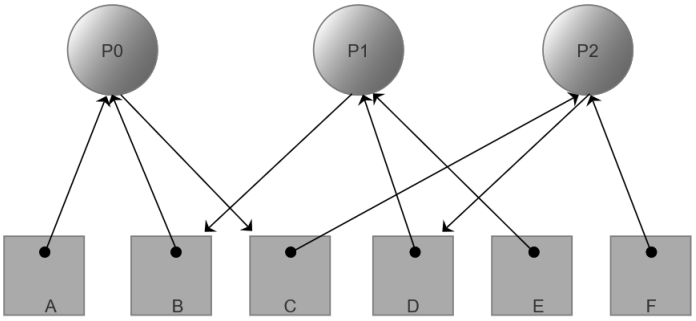
资源总量

A	B	C	D
3	1	2	1

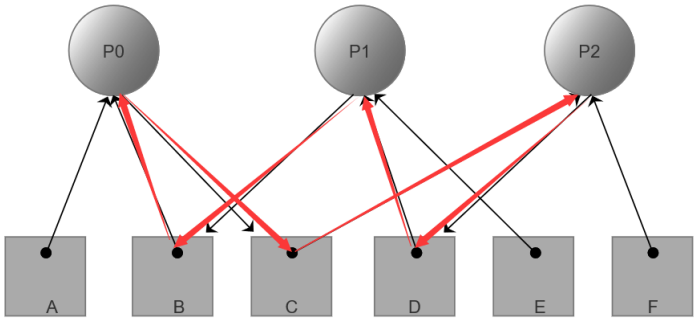
可用资源向量

习题 6.6

a.



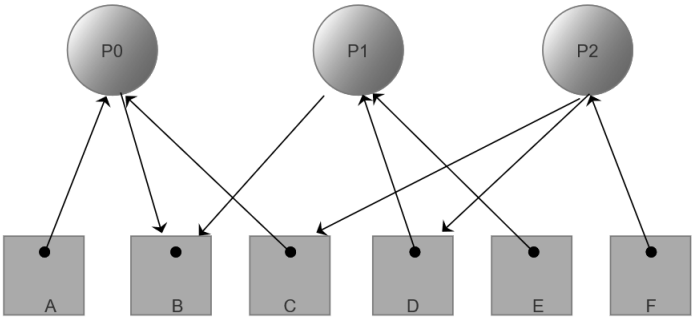
进程 P0 持有资源 B 的同时请求资源 C，进程 P1 持有资源 D 的同时请求资源 B，进程 P2 持有资源 C 的同时请求资源 D，从而导致死锁。如下所示：



b.

修改进程 P0 如下：

```
void P0(){
    while(true){
        get(A);
        get(C);
        get(B);
        // critical region
        // use A, C, B
        release(A);
        release(c);
        release(B);
    }
}
```



进程 P0 占有资源 C，进程 P1 占有资源 D，进程 P0 和进程 P1 都在请求资源 B，进程 P2 将

依次请求 C、D，若进程 P0 先得到资源 B，在它执行完后释放资源 B、C，从而进程 P1 得到资源，执行完毕后释放资源 B、D，最后进程 P2 得到所有资源执行，故不会导致死锁出现。

习题 6.15

$C=\begin{bmatrix}3\\2\\9\\7\end{bmatrix}$，$A=\begin{bmatrix}1\\1\\3\\2\end{bmatrix}$，故 $C-A=\begin{bmatrix}2\\1\\6\\5\end{bmatrix}$。

至少需要 10 单元的资源，此时的一种安全进程序列如下：

初始状态：$C=\begin{bmatrix}3\\2\\9\\7\end{bmatrix}$，$A=\begin{bmatrix}1\\1\\3\\2\end{bmatrix}$，$C-A=\begin{bmatrix}2\\1\\6\\5\end{bmatrix}$，可用资源 =3

进程 0 运行完成：$C=\begin{bmatrix}0\\2\\9\\7\end{bmatrix}$，$A=\begin{bmatrix}0\\1\\3\\2\end{bmatrix}$，$C-A=\begin{bmatrix}0\\1\\6\\5\end{bmatrix}$，可用资源 =4

进程 1 运行完成：$C=\begin{bmatrix}0\\0\\9\\7\end{bmatrix}$，$A=\begin{bmatrix}0\\0\\3\\2\end{bmatrix}$，$C-A=\begin{bmatrix}0\\0\\6\\5\end{bmatrix}$，可用资源 =5

进程 3 运行完成：$C=\begin{bmatrix}0\\0\\9\\0\end{bmatrix}$，$A=\begin{bmatrix}0\\0\\3\\0\end{bmatrix}$，$C-A=\begin{bmatrix}0\\0\\6\\0\end{bmatrix}$，可用资源 =7

进程 2 运行完成：$C=\begin{bmatrix}0\\0\\0\\0\end{bmatrix}$，$A=\begin{bmatrix}0\\0\\0\\0\end{bmatrix}$，$C-A=\begin{bmatrix}0\\0\\0\\0\end{bmatrix}$，可用资源 =10

至此，所有进程运行完毕。

若资源数量少于 10，不妨设有 9 单元，

初始状态：$C=\begin{bmatrix}3\\2\\9\\7\end{bmatrix}$，$A=\begin{bmatrix}1\\1\\3\\2\end{bmatrix}$，$C-A=\begin{bmatrix}2\\1\\6\\5\end{bmatrix}$，可用资源 =2

进程 0 运行完成：$C=\begin{bmatrix}0\\2\\9\\7\end{bmatrix}$，$A=\begin{bmatrix}0\\1\\3\\2\end{bmatrix}$，$C-A=\begin{bmatrix}0\\1\\6\\5\end{bmatrix}$，可用资源 =3

进程 1 运行完成：$C=\begin{bmatrix}0\\0\\9\\7\end{bmatrix}$，$A=\begin{bmatrix}0\\0\\3\\2\end{bmatrix}$，$C-A=\begin{bmatrix}0\\0\\6\\5\end{bmatrix}$，可用资源 =4

此时可用资源不足以使任何一个剩余的进程得以运行，会导致死锁，因此至少需要 10 单元的资源。

习题 6.18

a.

若出现死锁说明对所有就餐的哲学家都出现了：持有一个叉子而想要的另一个叉子被相邻的哲学家持有。对其中一名哲学家，若他是左撇子，则他持有了左边的叉子，想要右边的叉子，右边的叉子被右边的哲学家所持有，而右边的哲学家想要的另一个叉子被相邻的哲学家持有，说明右边的哲学家是左撇子，依次推下去所有的哲学家都是左撇子。对右撇子的哲学家同理，那么所有的哲学家都是右撇子，与假设相矛盾。

b.

假设左撇子 P_j 饥饿，也就是说有一部分人在就餐而 P_j 从不吃。若 P_j 没有左边叉子，这样他的左边邻居 P_i 一定持续的占有叉子而始终不吃完，因此 P_i 是右撇子，他抓住他的右边叉子，但永远得不到他的左边叉子来吃，也就是说 P_i 也饥饿。和 P_j 的情况相同，P_i 的左边邻居也是右撇子，依次推下去所有的哲学家都是饥饿的右撇子，这与假设 P_j 是左撇子相矛盾。因此 P_j 没有右边叉子，P_j 的右边邻居 P_k 持续的占有左边叉子而始终不吃完，因此他是饥饿的左撇子，他的右边邻居也是饥饿的左撇子，依此推下去所有的哲学家都是饥饿的左撇子，与条件矛盾，因此假设不成立，没有人饥饿。