Solutions for Tutorial Session N°4: Deadlock

Objective: To understand the concept of deadlock in operating systems, identify the conditions under which it occurs, and explore the main techniques used for deadlock prevention, avoidance, detection, and recovery.

Exercise 1

Consider a system with 5 processes (P0, P1, P2, P3, P4) and 3 resource types (R0, R1, R2). The state of the system is:

MAX

	R0	R1	R2
P0	7	5	3
P1	3	2	2
P2	9	0	2
P3	2	2	2
P4	4	3	3

ALLOCATION

	R0	R1	R2
P0	0	1	0
P1	2	0	0
P2	3	0	2
P3	2	1	1
P4	0	0	2

AVAILABLE

R0	R1	R2
3	3	2

1. How many instances are there for each resource in this system?
2. Calculate the **NEED** matrix.
3. Determine if the system is in a safe state using the **Banker's algorithm**.
4. Assuming the system receives the following requests sequentially, determine at each moment whether the request can be granted or not (**Banker's algorithm)**:
    - At time t1, P1 makes the following request: (1, 0, 2).
    - At time t2, P4 makes the following request: (3, 3, 0).
    - At time t3, P0 makes the following request: (0, 2, 0).

SOLUTION

1)

Resource	**Number of instances (AVAILBLE + ALLOCATION)**	
R0	3+2+3+2 =	**10**
R1	3+1+1=	**5**
R2	2+2+1+2=	**7**

2) NEED = MAX – ALLOCATION:

	R0	**R1**	**R2**
P0	**7**	**4**	**3**
P1	**1**	**2**	**2**
P2	**6**	**0**	**0**
P3	**0**	**1**	**1**
P4	**4**	**3**	**1**

3) Banker's Safety algorithm:

Work

Steps	R0	R1	R2
Step 0	3	3	2
Step 1	5	3	2
Step 2	7	4	3
Step 3	7	4	5
Step 4	7	5	5
Step 5	10	5	7

Finish

P0	P1	P2	P3	P4
F	F	F	F	F
F	T	F	F	F
F	T	F	T	F
F	T	F	T	T
T	T	F	T	T
T	T	T	T	T

The system is in a safe state: the sequence P1, P3, P4, P0, P2 is safe.

4)

P1 makes the following request: (1, 0, 2)
(1, 0, 2) ≤ **NEED[1]** and (1, 0, 2) ≤ **AVAILABLE**

We assume that the request is granted and update the data:

- **Available = Available − Request[i]**
- **Allocation[i] = Allocation[i] + Request[i]**
- **Need[i] = Need[i] − Request[i]**

NEED

	R0	R1	R2
P0	7	4	3
P1	**0**	**2**	**0**
P2	6	0	0
P3	0	1	1
P4	4	3	1

ALLOCATION

	R0	R1	R2
P0	0	1	0
P1	**3**	**0**	**2**
P2	3	0	2
P3	2	1	1
P4	0	0	2

AVAILABLE

R0	R1	R2
2	**3**	**0**

Afterwards, we apply the **safety algorithm**:

Work

Steps	R0	R1	R2
Step 0	2	3	0
Step 1	5	3	2
Step 2	7	4	3
Step 3	7	4	5
Step 4	7	5	5
Step 5	10	5	7

Finish

P0	P1	P2	P3	P4
F	F	F	F	F
F	T	F	F	F
F	T	F	T	F
F	T	F	T	T
T	T	F	T	T
T	T	T	T	T

- We find that the sequence P1, P3, P4, P0, P2 is safe, so P1's request can be granted.
- The OS confirms the changes made to the **ALLOCATION** and **NEED** matrices, and the **AVAILABLE** vector.

<u>P4 makes the following request: (3, 3, 0)</u>

(3, 3, 0) ≤ **NEED[4]**
but (3, 3, 0) > **AVAILABLE**

Therefore, this request cannot be granted, and P4 remains waiting (blocked).

<u>P0 makes the following request: (0, 2, 0)</u>

(0, 2, 0) ≤ **NEED[0]** and (0, 2, 0) ≤ **AVAILABLE**

We assume that the request is granted and update the data:

- **Available = Available − Request[i]**
- **Allocation[i] = Allocation[i] + Request[i]**
- **Need[i] = Need[i] − Request[i]**

NEED

	R0	R1	R2
P0	**7**	**2**	**3**
P1	0	2	0
P2	6	0	0
P3	0	1	1
P4	4	3	1

ALLOCATION

	R0	R1	R2
P0	**0**	**3**	**0**
P1	3	0	2
P2	3	0	2
P3	2	1	1
P4	0	0	2

AVAILABLE

R0	R1	R2
2	**1**	**0**

Now, we apply the **safety algorithm**:

Work

Steps	R0	R1	R2
Step 0	2	1	0

Finish

P0	P1	P2	P3	P4
F	F	F	F	F

- There is no **i** such that **Finish[i] == false** and **Need[i] ≤ Work**, so the system is not in a safe state; all 5 processes are in deadlock. Therefore, the request cannot be granted.
- The OS keeps (restores) the previous values of the **ALLOCATION** and **NEED** matrices, and the **AVAILABLE** vector.

Exercise 2

Consider the following systems. Use the **deadlock detection algorithm** to determine whether the processes in each system are in deadlock or not:

1) **System A :**

ALLOCATION

	R0	R1	R2	R3
P0	1	0	1	0
P1	2	0	0	1
P2	0	1	2	0

REQUEST

	R0	R1	R2	R3
P0	2	3	0	1
P1	2	0	1	0
P2	2	1	4	0

AVAILABLE

R0	R1	R2	R4
5	2	3	1

SOLUTION

The execution of the **deadlock detection algorithm** will reveal the following contents:

For i = 0..n-1, if Allocation[i] ≠ 0, then Finish[i] = false; otherwise Finish[i] = true
➔ so here Finish = (F, F, F)

Work

Steps	R0	R1	R2	R3
Step 0	5	2	3	1
Step 1	7	2	3	2

Finish

P0	P1	P2
F	F	F
F	T	F

The algorithm execution ends with the **Finish** vector containing two false values: **P0 and P2 are in deadlock.**

Note: The OS must apply one of the recovery techniques to resolve this deadlock situation.

2) **System B :**

ALLOCATION

	R0	R1	R2
P0	0	1	0
P1	2	0	0
P2	3	0	2
P3	2	1	1
P4	0	0	2

REQUEST

	R0	R1	R2
P0	0	0	0
P1	2	0	2
P2	0	0	0
P3	1	0	0
P4	0	0	2

AVAILABLE

R0	R1	R2
0	0	0

SOLUTION

The execution of the algorithm gives us the following steps:

Work

Steps	R0	R1	R2
Step 0	0	0	0
Step 1	0	1	0
Step 2	3	1	2
Step 3	5	2	3
Step 4	5	2	5
Step 5	7	2	5

Finish

P0	P1	P2	P3	P4
F	F	F	F	F
T	F	F	F	F
T	F	**T**	F	F
T	F	**T**	**T**	F
T	F	**T**	**T**	**T**
T	**T**	**T**	**T**	**T**

The sequence P0, P2, P3, P4, P1 will set **Finish[i]** to true for all **i**, so the system is not in deadlock.