Name: Nimra Nasir
Reg No: 19-CP-35

Operating System

Lab: 13 (Example)

Examples:

Example: 1

Consider the following table of system.

Process	Allocation				Max				Available			
	R_1	R_2	R_3	R_4	R_1	R_2	R_3	R_4	R_1	R_2	R_3	R_4
P_1	0	0	1	2	0	0	1	2	2	1	0	0
P_2	2	0	0	0	2	7	5	0				
P_3	0	0	3	4	6	6	5	0				
P_4	2	3	5	4	4	3	5	6				
P_5	0	3	3	2	0	6	5	2				

Compute Need matrix.
Is the system in safe state? Justify

Answer:

Need matrix [i] = Max [i] − Allocated [i]

Process	Allocation				Max				Need			
	R_1	R_2	R_3	R_4	R_1	R_2	R_3	R_4	R_1	R_2	R_3	R_4
P_1	0	0	1	2	0	0	1	2	0	0	0	0
P_2	2	0	0	0	2	7	5	0	0	7	5	0
P_3	0	0	3	4	6	6	5	6	6	6	2	2
P_4	2	3	5	4	4	3	5	6	2	0	0	4
P_5	0	3	3	2	0	6	5	2	0	3	2	0

Is the system is safe state?

By applying the Banker's Algorithm:

let Avail = Available , ie Avail (2, 1, 0, 0)

Iteration : 1

check all the processes from P_1 to P_5

For P_1 :

if (P_1 Need < Avail) → True

then calculate

Avail = Avail + Allocated [P_1]

(2, 1, 0, 0) (0, 0, 1, 2)

Avail = (2, 1, 1, 2)

Iteration : 1

For P_2 :

if (P_2 Need < Avail) → False

then check for next process

For P_3 :

if (P_3 Need < Avail) → False

then check for next process

Iteration : 1

For P_4 :

if (P_4 Need < Avail) → True

then calculate

Avail = Avail + Allocated [P_4]

(2, 1, 1, 2) (2, 3, 5, 4)

= (4, 4, 6, 6)

Iteration : 1

For P_5 :

if (P_5 Need < Avail) → True

then calculate

Avail = Avail + Allocated [P_5]

(4, 4, 6, 6) (0, 3, 3, 2)

Avail = (4, 7, 9, 8)

Iteration : 2

Check only process P_2 to P_3

For P_2:

if (P_2 Need < Avail) → True

then calculate

Avail = Avail + Allocated [P_2]

(4,7,9,8) (2,0,0,0)

Avail = (6,7,9,8)

Iteration : 2

For P_3:

if (P_3 Need < Avail) → True

then calculate

Avail = Avail + Allocated [P_3]

(6,7,9,8) (0,0,3,4)

Avail = (6, 7, 12, 12) = system capacity

Since, all the processes got True marked, no further iterations are required.

Therefore, **safe sequence = P_1, P_4, P_5, P_2, P_3**

Therefore, the system is in the safe state

Example : 2

5 processes P_0 through P_4

3 resource types A (10 units), B (5 units), C (7 units).

Process	Allocation			Max			Available		
	A	B	C	A	B	C	A	B	C
P_0	0	1	0	7	5	3	3	3	2
P_1	2	0	0	3	2	2			
P_2	3	0	2	9	0	2			
P_3	2	1	1	2	2	2			
P_4	0	0	2	4	3	3			

Process	Allocation			Max			Need		
	A	B	C	A	B	C	A	B	C
P_0	0	1	0	7	5	3	7	4	3
P_1	2	0	0	3	2	2	1	2	2
P_2	3	0	2	9	0	2	6	0	0
P_3	2	1	1	2	2	2	0	1	1
P_4	0	0	2	4	3	3	4	3	1

The system is in a safe state since the sequence $< P_1, P_3, P_4, P_2, P_0 >$ satisfies safety criteria

Now P_1 requests (1,0,2):

check that request $\leq$ Available

(that is, $(1,0,2) \leq (3,3,2)$) $\Rightarrow$ True

Process	Allocation			Need			Available		
	A	B	C	A	B	C	A	B	C
P_0	0	1	0	7	4	3	2	3	0
P_1	3	0	2	0	2	0			
P_2	3	0	2	6	0	0			
P_3	2	1	1	0	1	1			
P_4	0	0	2	4	3	1			