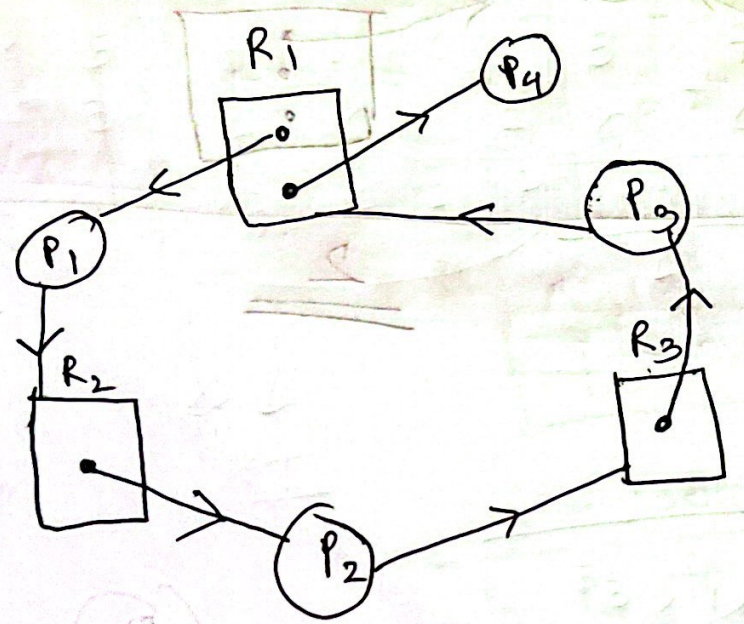


Here, deadlock will not occur. Because,

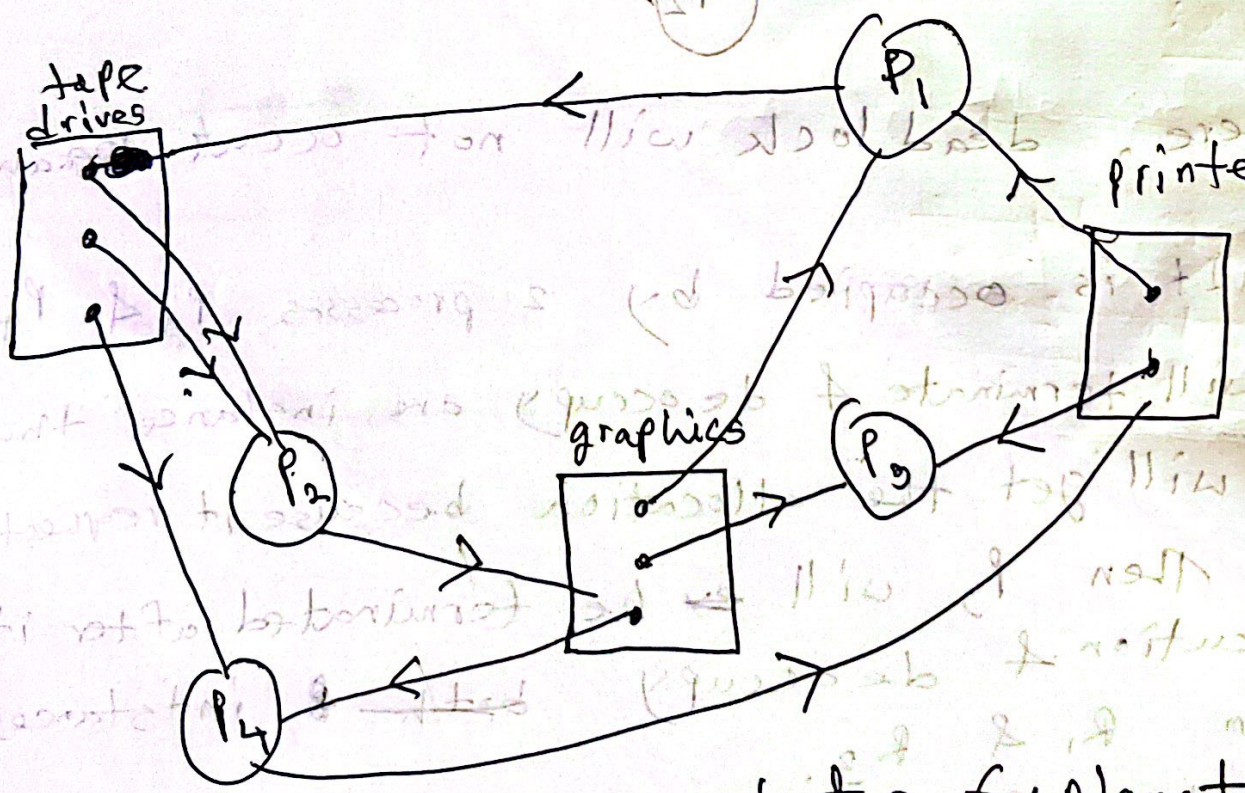
R1: It is occupied by 2 processes P1 & P4. P4 will terminate & deoccupy one instance thus P3 will get the allocation because it requested for it. Then P3 will ~~ex~~ be terminated after its execution & deoccupy ~~both~~ 2 instances from R1 & R3.

R3: Now, R3 will accept req from P2 & after allocation P2 will be executed & deoccupy instances from R3 & R2.

R2: Lastly, R2 will accept req of P1 and it will be executed.

Scanned with CamScanner

2



Not in deadlock state. Explanation same as the previous one. Hint: P_3 will terminate and free the occupied resources.

Scanned with CamScanner

3

Max:				
P_1:	1	6	5	2
P_2:	2	3	6	6
P_3:	0	6	5	2
P_4:	0	6	5	6

Alloc.:			
1	2	3	1
1	3	6	5
0	6	3	2
0	0	1	4
2	11	13	12

Need: Max − Alloc.

0	4	2	1	F T
1	0	0	1	F T
0	0	2	0	T
0	6	4	2	T

Available:

1	5	2	0

P_1	P_2	P_3	P_4
~~F~~ T	~~F~~ T	~~F~~ T	~~F~~ T

Safe sequence:

P_3, P_4, P_1, P_2

Work:

~~1~~ 5	~~2~~ 0		
~~1~~ 11	~~5~~ 2		
~~1~~ 11	~~6~~ 6		
~~2~~ 13	~~9~~ 7		
3	16	15	12

Now, operate banker's algorithm & check whether the system is at safe state or not.

P_1 requests for (0 2 1 0)

1) If req $\leq$ need of P_1

(0 2 1 0) $\leq$ (0 4 2 1) ✓

2) req $\leq$ available

(0 2 1 0) $\leq$ (1 5 2 0) ✓

3) Now, operate banker's algorithm after updating,

i) Available = Available − Req. of P_1

ii) Allocation of P_1 = Allocation of P_1 + Req. of P_1

iii) Need of P_1 = Need of P_1 − Req. of P_1

Scanned with CamScanner

Max:

P_1:	1	6	5	2
P_2:	2	3	6	6
P_3:	0	6	5	2
P_4:	0	6	5	6

Alloc.:

1	4	4	1
1	3	6	5
0	6	3	2
1	0	1	4
0			

Need:

0	2	1	1	F
0	0	1		f
0	2	0		F
0	6	4	2	f

Available:

1	3	1	0

P_1	P_2	P_3	P_4
F	f	F	F

safe sequence:

work:

1 3 1 0

∴ They system is not in the safe state after allocating P_1's request for resources.

Therefore Deadlock occurs after accepting the request.

Scanned with CamScanner