Akshay Anand
CSE - 6th sem
Roll - 01

Operating System

06/03/18

Q1. Write down the necessary condns for deadlock.

Q2. Consider the following snapshot of a system

Processes	Allocation A B C D	Max A B C D	Available A B C D	Need A B C D
			1 5 2 0	
P_0	0 0 1 2	0 0 1 2		
P_1	1 0 0 0	1 7 5 0		
P_2	1 3 5 4	2 3 5 6		
P_3	0 6 3 2	0 6 5 2		
P_4	0 0 1 4	0 6 5 6		

Answer the following questions based on the Banker's algorithm

i) What is the content of the matrix 'Need'?

ii) Is the system in a safe state?

iii) If a ~~for~~ request from process P_1 arrives for (0, 4, 2, 0).
Can the request be granted immediately?

1. Ans - A deadlock situation can arise if the following four conditions hold simultaneously in a system:

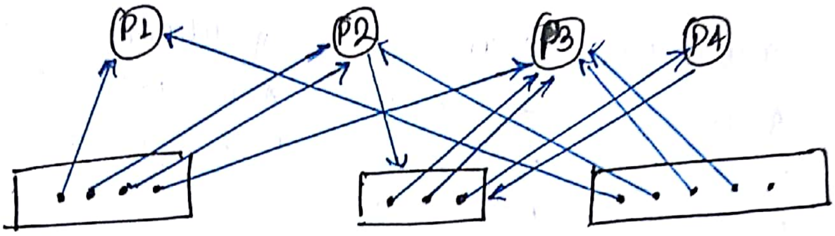
(i) Mutual exclusion: At least one resource must be held in a non-sharable mode, that is, only one process at a time can use the resource.

ii) Hold and wait: A process must be holding one resource and waiting to acquire additional resources that are currently being held by other processes.

iii) No preemption: Resources can not be preempted, that is, a resource can be released only voluntarily by the process holding it, after that process has completed its task.

iv) Circular wait: A set {P_0, P_1, ... P_n} of waiting processes must exist such that P_0 is waiting for a resource held by P_1, P_1 is waiting for resource held by P_2, ... P_{n-1} is waiting for a resource held by P_n, and P_n is waiting for a resource held by P_0.

Q3. From the given resource allocation graph, fill the resource usage table :
current allocation, current request, allocated resources
Then detect whether the system is in deadlock state or not



Q. Ans :

a)

Process	Allocation A B C D	Max A B C D	Need A B C D	Available A B C D
① P_0 ✓	0 0 1 2	0 0 1 2	0 0 0 0	1 5 2 0
⑤ P_1 ✓	1 0 0 0	1 7 5 0	0 7 5 0	+0 0 1 2
② P_2 ✓	1 3 5 4	2 3 5 6	1 0 0 2	1 5 3 2
③ P_3 ✓	0 6 3 2	0 6 5 2	0 0 2 0	+1 3 5 4
④ P_4 ✓	0 0 1 4	0 6 5 6	0 6 4 2	2 8 8 6

Available (continued):
+0 6 3 2
2 14 11 8
+0 0 1 4
2 14 12 12
+1 0 0 0
3 14 12 12

b) Safe sequence
< P_0 P_2 P_3 P_4 P_1 >

∴ The system is in safe state

c)

Processes	Allocation A B C D	Max A B C D	Need A B C D	~~Available A B C D~~	Available A B C D
① P_0 ✓	0 0 1 2	0 0 1 2	0 0 0 0	~~1 5 2 0~~	1 5 2 0
⑤ P_1 ✓	0 4 2 0	1 7 5 0	1 3 3 0	~~+0 0 1 2~~	−0 4 2 0
② P_2 ✓	1 3 5 4	2 3 5 6	1 0 0 2	~~1 5 3 2~~	1 1 0 0
③ P_3 ✓	0 6 3 2	0 6 5 2	0 0 2 0	~~+0 4 2 0~~	+0 0 1 2
④ P_4 ✓	0 0 1 4	0 6 5 6	0 6 4 2	~~1 9 5 2~~	1 1 1 2

~~+1 3 5 4~~
~~2 12 10 6~~
~~+0 6 3 2~~
~~2 18 13 8~~
~~+0 0 1 4~~
~~2 18 14 12~~

Available (continued):
+1 3 5 4
2 4 6 6
+0 6 3 2
2 10 9 8
+0 0 1 4
2 10 10 12
+1 4 2 0
2 14 12 12

Safe sequence : < P_0 P_2 P_3 P_4 P_1 >

∴ The request can be granted immediately.

Processes	Allocation A B C	Request A B C	Available A B C
(1) P_1 ✓	1 0 1	0 0 0	0 0 1
(4) P_2 ✓	2 0 1	0 1 0	+ 1 0 1
(2) P_3 ✓	1 2 2	0 0 0	1 0 2
(3) P_4 ✓	0 1 0	0 1 0	+ 1 2 2
			2 2 4
			+ 0 1 0
			2 3 4
			+ 2 0 1
			4 3 5

Safe sequence

$= \langle P_1\ P_3\ P_4\ P_2 \rangle$

$\therefore$ System is not in deadlock state