Operating Systems 2 (CS3523)

Theory Assignment 2

SURAJ TELUGU

CS20BTECH11050

Variant Question 8.4. Sol

Drawbacks for using Object F as Lock for A, B, C, D, E resources:

- This scheme **does not use all the resources efficiently** for example assume P1 wants to use resource A, B and P2 wants to use resources C, D, E. P1 has acquired lock for object F and using A, B during this time P2 cannot use C, D, E even though these resources are unused.
- In this scheme there is mutual exclusion of threads at every instant and multiple threads cannot access the resources at same time. Therefore, it indirectly **prevents the concept of Multi – Threading**.
- Once a thread acquires lock for object F all the other threads wait until it completes so the **CPU is also not used efficiently** and the performance of system is decreased.
- Though deadlocks are prevented in this scheme the efficiency and performance of system are decreased drastically.
- Hence, the containment solution is not an efficient solution for preventing deadlocks.

Question 8.6. Sol

8.6A. Sol

Arguments for installing deadlock – avoidance algorithm:

- By installing deadlock – avoidance algorithm (like banker's algorithm) we can ensure that no deadlocks occur and all the jobs continue smoothly.
- The Values of run times are more accurate since there are no deadlocks since no job is aborted in between.
- There is also no need for an external devices or operators to check and reboot the system if deadlock occurs.
- Even if the number of jobs is increased which may result in frequent deadlocks by installing deadlock – avoidance algorithm we can avoid all such deadlocks and execute the jobs perfectly.

8.6B. Sol

Arguments for not installing deadlock – avoidance algorithm

- As per the given values calculating additional cost system without deadlock – avoidance algorithm and the system with deadlock – avoidance algorithm.

Cost of deadlocks: (loss due to deadlocks)

($2/job x 10 jobs x ½ (half complete)) x 2 (deadlock/month) = **$20/month**

Cost of additional run time used by system with the deadlock – avoidance algorithm:

10% of 5000 jobs x $2/job = **$1000/month** (loss due to deadlock – avoidance algorithm)

- As per above observations system installed with deadlock – avoidance algorithm creates additional loss $980 compared to system without deadlock – avoidance algorithm. Since the deadlocks occur infrequently, and they cost little when they do occur.
- Since deadlock – avoidance algorithm uses additional CPU time it results in less idle time for system thus if a greater number of jobs are given system cannot perform efficiently.

Question 8.22. Sol

Proof by Contradiction:

Let the threads be {T1, T2, T3} and the same type resources be {R1, R2, R3, R4}

Assume that the described system is deadlocked

As per **hold and wait** condition each thread should have acquired one resource and is requesting for another resource.

Since there are 3 processes and 4 resources one process can acquire two resources use them and release them which can then be used by other waiting processes.

For example, if T1 acquires R1 and requests for another resource and T2 acquires R2 and requests for another resource then T3 which has acquired R3 can also acquire R4 and use them and release R3, R4. Since resources are same type R1, R2, R3, R4 are identical and they can be also be used by T1, T2.

Therefore, **Circular wait is prevented** and the system is deadlock free.

Question 8.24. Sol

- **Given Version of Dining Philosophers Problem:**

The chopsticks are placed at the centre of the table and any two of them can be used by a philosopher. Assume that requests for chopsticks are made one at a time.

This problem can be assumed as a problem with 5 threads and 5 resources where each thread requires 2 resources to execute.

- **Rule for Deadlock prevention:**

Whenever a philosopher (thread) requests for the first chopstick (resource), he is not allowed to hold (acquire) chopstick only if there is no other philosopher with two chopsticks (every philosopher holds only one stick) and if there is only one chopstick remaining.

- **Explanation:**

The deadlock occurs only when all the philosophers try to hold the chopstick at the same time which results into endless circular waiting and starvation. For avoiding such problem using above rule when the last philosopher tries to hold chopstick when each philosopher has 1 chopstick, he is not allowed to hold chopstick which gives the chance for one of the four philosopher which breaks the circular waiting chain making the solution deadlock free.

Question 8.28 Sol

The total available instances of resources A, B, C, D:

$(\sum Allocation[i][j] + \sum Available[i][j])_{4x1}$ (sum for i = 0 to 4 and j = 0, 1, 2, 3, 4) = **[15 11 10 11]**

From the given snapshot of the system calculate need matrix

$[Need]_{5 \times 4}$ = $[Max]_{5 \times 4}$ - $[Allocation]_{5 \times 4}$

	Allocation				Max				Available (after execution)				Need			
Process	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D
T0	3	1	4	1	6	4	7	7	**11**	**8**	**8**	**10**	3	3	3	6
T1	2	1	0	2	4	2	3	2	**13**	**9**	**8**	**12**	2	1	3	0
T2	2	4	1	3	2	5	3	3	**4**	**6**	**3**	**9**	0	1	2	0
T3	4	1	1	0	6	3	3	2	**8**	**7**	**4**	**9**	2	2	2	2
T4	2	2	2	1	5	6	7	5	**15**	**11**	**10**	**11**	3	4	5	4

8.28A. Sol

Consider the following safe sequence obtained using banker's algorithm

Initial Available resources = (2, 2, 2, 4)

Since the available resources (2, 2, 2, 4) are sufficient for T2, T3 either T2 or T3 can start be the starting process of safe sequence

Let T2 be starting process of the safe sequence.

After execution of T2 the available resources are (4, 6, 3, 9) any process other than T4 can execute now as their need has met.

Let T3 be next process of safe sequence.

After execution of T3 the available resources are (8, 7, 4, 9) still T4 cannot be executed all other processes can be executed.

Let T0 be next process of safe sequence.

After execution of T0 the available resources are (11, 8, 8, 10). Now any process can be executed with the current available resources.

Let T1 be next process of safe sequence.

After execution of T1 the available resources are (13, 9, 8, 12).

Let T4 be next process of safe sequence.

After execution of T4 the available resources are (15, 11, 10, 11).

Therefore, one of the possible safe sequences using banker's algorithm can be:

(T2, T3, T0, T1, T4)

8.28B. Sol

No, the request (2, 2, 2, 4) from T4 cannot be acquired immediately because the initial available resources are (2, 2, 2, 4) and requested resources = available. After acquiring the Allocation of T4 = (4, 4, 4, 5) and available = (0, 0, 0, 0) and Need is (1, 2 ,3, 0). After updating the need of all the processes is greater than available resources. The system cannot acquire a safe state and always remains in unsafe state. Hence, the requested resources cannot be acquired immediately.

8.28C. Sol

Yes, the request (0, 0, 1, 0) from T2 can be acquired immediately because the initial available resources are (2, 2, 2, 4) and requested resources < available. After acquiring the Allocation of T2 = (2, 4, 2, 3) and available = (2, 2, 1, 4). After updating the allocation is less than Max so there is no problem. Hence, the requested resources can be acquired immediately.

8.28D. Sol

Yes, the request (2, 2, 1, 2) from T3 can be acquired immediately because the initial available resources are (2, 2, 2, 4) and request resources < available. After acquiring the Allocation of T3 = (6, 3, 2, 2) and available = (0, 0, 1, 2). After updating the allocation is less than Max so there is no problem. Hence, the requested resources can be acquired immediately.