Name - Shubham
Roll no. - 2301730100
COURSE - B Tech CSE AI/ML
Sec - B

DELTA Page No.
Date / /

Assignment - 3

1Q Explain race conditions with a real world example outside of computing and show how Mutual exclusion addresses?

⇒ A Race conditions happen when Two or more processes Try to do the same task at the same time and the final Result Depends on who finishes first.

- Example: Imagine two people with drawing money from the same banks account at the same time each checks balance (₹1000) Both Decides to take ₹1000 and both succeed to leaving the amount at ₹-1000 (wrong Result).

- Mutual Exclusion: it ensures only one person accesses the account at a Time, preventing overlap and Keeping the Data correct.

2Q Compare peterson's solution and semaphores in terms of Implementation complexity and Hardware dependency.

⇒ Peterson's solution is a Software-based Method, for Two processes using shared Variables. it simple But works only Two processes and Depend on Busy waiting (CPU Time wasted).

- Semaphores are more general, used for Multiple processes and supported by Hardware atomic Operations. they are easier to implement at system level and less error-prone than peterson's manual code approach.

3Q Monitors provide automatic Mutual Exclusion, Meaning only one process can access shared Data at a Time

2301730100

DELTA Page No.
Date / /

without Manually Handling semaphores.
In a Multi-core system, this Reduces the chance of programming error like forgetting to Release and lock and Make synchronization simpler, safer and more efficient across Multiple cores.

4Q Starvation occurs when writers Keep waiting because Readers continuosly access the shared data, never giving writers a turn.
To prevent it, use writer priority - once a writer is waiting, no new Readers are allowed to enter until the writer finishes. This ensures fair access for both Readers and writers.

5Q To prevent "Hold and wait" a process Must Request all Resources at once before Starting execution.
The Drawbacks is Resource wastage - a process May hold Some Resources idle while waiting for others, Reducing system efficiency and Resource utilizatio

6Q Dining philosphers problem : five philosphers sit around a circular Table blw each pair philosphers is one fork. each philospher needs Two forks (one (on the left and one on the Right) to eat.

- Dining Deadlock situation : if all philosphers pick up their left fork at the same time then all will wait for their Right fork which is already taken by their Neighbour As a Result, no one can continue and the system Reaches a Deadlock - where everyone

is waiting forever.

Avoiding Deadlocks :-

(1) Allowing only four philosphers to try to pick up forks at the same time (using a semaphores for entry) this ensures at least one philosphers can eat and Release forks.

(2) changing the order in which forks are picked up for example, some philosphers pick the left forks first and other pick the Right first this breaks the circular wait.

7(a) a) critical sections :-

→ shared flight Data Table (where Radar updates air craft positions)

→ flight path Database (Used by path calculation and communication Modules)

These Must be Mutual Exclusive - only one process can update them at a Time to avoid wrong flight info.

→ Suitable IPC Mechanism :-

- Use Semaphores or Message queues for synchronization
- Message Queues are best for Real-Time Systems - they allows fast, safe data Transfer without over writing shared Memory.

b) if Deadlock occurs :-

Example - Radar process holds access to flight Data and flight path process waits for it, while holding

2301730100

DELTA Page No.
Date / /

another needed Resource.

Detection :-

→ use a Resource allocation graph to check circular wait conditions.

→ The system regularly Monitors for blocked processes.

Recovery :-

→ preempt (Release) one processes Resource like Temporarily Suspending Radar update) and let the other complete.

→ Restart the affected process after freezing Resource.

8) a) Step-1 find number of interuppts per second.

= (Total bytes per second) ÷ (Bytes per interupt)

= 512,000 ÷ 100

= 5,120 interrupts per Seconds.

Step-2 Total CPU time spent.

= (Interrupts per second) x (time per interrupt)

= 5120 x 5 μs

= 25,600 μs = 0.0256 per Second.

CPU spends 2.56% of its Time Handling interrupts.

b) Use direct Memory Access (DMA) - its Transfers Data Directly btw Device and Memory without CPU improvement - This Reduces interrupt frequency and CPU overhead, while Keeping the same Transfer Rate.

9) Bankers's Algorithm Simulation

A system has the following Resources :-

2301730100

DELTA Page No.
Date / /

Total Instance : A = 10, B = 5, C = 7.
Allocation & Max Tables

process	Allocation (A, B, C)	Max (A, B, C)
P_0	(0, 1, 0)	(7, 5, 3)
P_1	(2, 0, 0)	(3, 2, 2)
P_2	(3, 0, 2)	(9, 0, 2)
P_3	(2, 1, 1)	(4, 2, 2)
P_4	(0, 0, 2)	(5, 3, 3)

a) Calculate the Need Matrix.

process	Need (A, B, C)
P_0	(7, 4, 3)
P_1	(1, 2, 2)
P_2	(6, 0, 0)
P_3	(2, 1, 1)
P_4	(5, 3, 1)

Available (initial)
sum allocated = A : 7, B : 2, C : 5
Available = (10-7, 5-2, 7-5) = (3, 2, 2)

b) find a process whose need ≤ Available, finish it, add its Allocation to avoid, Repeat.

1. P_1 needs (1, 2, 2) ≤ (3, 3, 2) → finish → Available becomes (5, 3, 2)
2. P_3 needs (2, 1, 1) ≤ (5, 3, 2) → finish → Available becomes (7, 4, 3)
3. P_0 needs (7, 4, 3) ≤ (7, 4, 3) → finish → Available becomes (7, 5, 3)
4. P_2 needs (5, 0, 0) ≤ (7, 5, 3) → finish → Available becomes (10, 5, 5)
5. P_4 finish next

So, a safe sequence is $P_1 \rightarrow P_3 \rightarrow P_0 \rightarrow P_2 \rightarrow P_4$
conclusion → system is an safe State.

c) if P_1 requests (1, 0, 2) checks whether it can be granted immediatly.

2301730100

DELTA Page No.

Date / /

1. check Requst ≤ P1's need : (1,0,2) ≤ (1,2,2) → yes
2. check Requst ≤ Available : (1,0,2) ≤ (3,3,2) → yes
3. Tentatively allocate and test safety
   - New Available = (3,3,2) − (1,0,2) = (2,3,0)
   - After that the system still has a safe completion order ($P_1 \rightarrow P_3 \rightarrow P_0 \rightarrow P_2 \rightarrow P_4$).

yes → the Requst can be granted immediatly (system Remains Safe).