Name : Kashish Pundir
Roll No : 2301420006
Course : B Tech CSE (DS)

19/11/2025.

OS Assignment No. 3

☆ Part-A :

Q-1. Explain race conditions with a real-world ex outside of computing & show how mutual exclusion addresses it.

Ans. Two people sharing one wallet decide independently to withdraw cas

- Person A looks, sees $100, decides to withdraw $80.
- Person B at same time looks, also sees $100, decides to withdraw $50. If both act concurrently without coordination both withdrawals would succeed & the bank account becomes -ve or overdraft happens - final state depends on interleaving of steps. Final outcome depends on timing of concurrent actions.

⇒ Mutual Exclusion fix : Make wallet critical section protected by a lock. Only one person may "open" the wallet at a time.

- Person A acquires lock, reads $100, withdraws $80, update balance to $20, releases lock.
- Only after the lock is released can Person B acquire it, read updated $20 & proceed. It forces a serial ordering of critical updates, eliminating unsafe interleaving that caused the race.

Q-2 Compare Peterson's Solution & semaphores in terms of implementation complexity & hardware dependency.

Ans. Peterson's Solution : In terms of complexity, very simple algorithmically for 2 processes. For hardware dependency, it requires atomic reads/writes to shared memory locations & assumes sequentially consistent memory that reads are indivisible - no special CPU instruction needed.

Semaphores : In terms of complexity, higher level abstraction i.e. more complex to implement but are hardware/OS-backed & portable to multi-core/multiprocess systems.

Q-3 Producer-consumer problem can be solved using either semaphores or monitors. Identify one advantage of using monitors in a multi-core system.

Sol. Monitors encapsulate data & synchronization together, allowing runtime or language implementation to provide fairness & efficient blocking

On multi-core systems monitors can be implemented with kernel support that put threads to sleep & wake them efficiently on target core, reducing CPU spinning & cache-coherence traffic compared to naive semaphore / busy-wait approaches.

Q-4. For Reader-Writer problem, show how starvation can occur & describe one method to prevent it.

Ans. If writers are rare & readers continuously arrive, a writer may never get exclusive access because new readers keep acquiring read lock. Conversely, if writers are given strict priority, readers can starve. So, starvation occurs when scheduling policy always favors other class.

Prevention method — Fair policy - use a queuing mechanism that orders requests by arrival time (FIFO) or implement priority aging: increase priority of waiting writers over time so they eventually preempt incoming readers.

Q-5 In deadlock prevention, the "Hold & wait" condition can be eliminated. Explain one practical drawback of doing so in OS.

Ans. Elimination means require processes to request all resources they will ever need before starting.

Drawbacks: It forces processes to claim max. possible resources up front, resulting in severe resource underutilization & poor concurrency - many resources remain reserved but idle, reducing throughput & causing poorer overall system utilization. Also it's often impossible to know all future needs in advance for real workloads.

Q-5 ... concurrent system, a [illegible]

Part B (Numerical Based)

Q-6. ~~[illegible]~~ / Distributed Deadlock Detection Simulation:

Three sites S1, S2 and S3 have following wait for graph fragments:

- S1 : P1 → P2, P3 → P4
- S2 : P2 → P5, P5 → P6
- S3 : P6 → P1

a) Combine these fragments to form global wait-for graph

b) Detect if a deadlock exists & list processes involved

c) Suggest one distributed algorithm that could be applied to detect it

Sol. a) Graph : P1 → P2 → P5 → P6 (with arrow from P6 back to P1)   P3 → P4 (disconnected)

b) Since cycle exists, deadlock exists. Process involved in deadlock are : P1, P2, P5, P6 (P3 & P4 are not part of cycle)

c) Chandy-Misra-Haas (edge chasing (probe) algorithm): Standard distributed dealock detection method for wait for graphs

⇒ • Any process that suspects it might be deadlocked can initiate detection by sending probe messages along wait for edges

- A probe is a tripel (initiator, sender, receiver).

Q-7. Distributed file System Performance : A DFS has following characteristics:

- Average local file access time : 5ms
- Average remote file access time : 25 ms
- Probability of file being remote : 0.3

a) Calculate expected file access time

b) Suggest one caching strategy to improve performance & justify.

Sol. a) $T_{local} = 5ms$

Expected access time $(E[T]) = P_{local} \cdot T_{local} + P_{remote} \cdot T_{remote} = 0.7 \times 5 + 0.3 \times 25$
$= 11 ms$

b) Client side read cache (LRU) with short leases - each client keeps a small local cache of recently read files.

After caching, $P_{local, new} = 0.7 + 0.3 \times h$

$$E[T]_{new} = P_{local, new} \cdot 5 + (1 - P_{local, new}) \cdot 25$$

If h = 0.8 (80% of remote file accesses hit cache):

$$P_{local, new} = 0.7 + 0.3 \times 0.8 = 0.94$$

$$E[T]_{new} = 0.94 \times 5 + 0.06 \times 25 = 4.7 + 1.5 = 6.2 ms$$ (Drop from 11)

Q-8. In concurrent system, a full checkpoint takes 200 ms, while an incremental checkpoint takes 50 ms. System must maintain recovery capability within 1 sec recovery point objective (RPO).

a) Propose an optimal mix of checkpointing methods over a 10-second period to meet RPO with minimal overhead.

b) Explain your reasoning.

Ans a) To meet an RPO of 1s you must create atleast one checkpoint every 1 second. full checkpoints are 4x more expensive than incrementals.

Total overhead = $10 \times 50 = 500$ ms over 10s.

This is minimal-overhead solution because replacing any incremental with full increases overhead.

b) RPO only constraints time since last checkpoint, not the type. So, meeting ≤ 1s between checkpoints constraint while minimizing cost means choosing cheapest checkpoint type for each checkpoint.

→ Practical caveats:

- Recovery Time objective
- Chain robustness
- Maintenance windows

Q-9 Case Study → Global E-Commerce platform:

A global e-commerce company runs its services using a distributed OS deployed across multiple continents.

a) Identify main distributed scheduling challenges for handling flash sales events & suggest an algo. suitable for load balancing

b) Propose fault tolerance strategy that ensures service availability even in event of regional data center failures, considering load recovery time objective RTO & RPO.

Sol a) Challenges:

- Massive load
- Hotspots & skew
- Latency sensitivity
- Stateful sessions
- Coordination overhead
- Capacity heterogeneity

Algos:

① Global traffic routing
② Regional load balancer + scheduler
③ Hot-key handling & session affinity

b) Keep services available even if an entire region fails.

Strategies:

① Stateless services
② Stateful services
③ Database architecture choices
④ Recovery & failover mechanics.
⑤ Backups & checkpoints