Operating System
Assignment - 4

Ayush Gupta
2301010206 (BTech (SE (core))

Ans1) Race Conditions and Mutual Exclusion

→ Race Condition: Occurs when the result of operations depends on the specific, unpredictable order of multiple processes/ thread of accessing a shared resource.

→ Mutual Exclusion: Solves this by ensuring only one processes can enter the critical section (the code that manipulates the shared resource) at a time. The first process acquires a lock, performs the complete operation, and releases the lock, forcing the second process to wait.

Ans2) Peterson's Solution vs Semaphores

Feature	Peterson's Solution	Semaphores
→ Implementation Complexity	Low to Moderate. Pure Software logic using shared variables	Moderate to high. Requires as kernel support for waiting queues and system calls (wait() / signal).
→ Hardware Dependency	Low. Primarily dependent on memory consistency / barriers on modern CPUs	High. Relies on atomic hardware instructions (like Test And Set) for kernel implementation.
→ Scope	Limited to mutual exclusion for two processes	General Synchronization for N process.

Ans3) Advantage of Monitors in Multi-Core Systems

→ Monitors automatically encapsulate the shared data and all necessary locking / synchronization logic within a single construct

→ This design ensure that synchronization is centralized and compiler-managed, making it virtually ~~impo~~ impossible for a programmer to forget to acquire or release a lock that could lead to deadlocks or subtle errors across multiple cores.

Ans4) Reader - Writer Starvation and Prevention

→ Starvation: In a Readers - Preference solution, if a continuous stream of new Readers keeps arriving, they are always given priority over a waiting writer. The writer may be ~~ide~~ indefinitely delayed because the resource is always busy with reading.

→ Prevention: Implement a writer - preference or fair scheduling policy. A writer - preference scheme prevents any new readers ~~for~~ from starting once a writer is waiting. After the currently active Readers finish, the waiting writer is waiting. After the currently active readers finish, the waiting.

Ans5) Drawback of Eliminating "Hold and wait".

→ Practical ~~Dead~~ Drawback: Low Resource utilization. A process must hold resources from the start, even if it won't use them untill the very end. For that entire duration, the resource sits idle, unavailable to other processes that could be using it, lading to poor system throughout and efficiency.

Ans6) Distributed deadlock detection simulation

a) Global wait - for Graph (WFG)

Combine the local fragments ($P_1 \to P_2$, $P_3 \to P_4$, $P_2 \to P_5$, $P_5 - P_6$, $P_6 \to P_1$):

$P_1 \to P_2 \to P_5 \to P_6 \to P_1$ and $P_3 \to P_4$

b) Deadlock Detection and Processes Involved

→ Deadlock exists : Yes

→ Reason: There is a cycle in the Global WFG

→ Processes Involved: P_1, P_2, P_5 and P_6.

stream.

(8) Suitable Distributed Algorithm.
The Chandy - Misra - Haas (CMH) Algorithm
→ Mechanism : A process waiting for a resource initiates a probe message. If the probe returns to its initiator, a cycle is confirmed.

Ans7) Distributed File System Performance
a) Expected file Access Time
$E[T] = (5ms \times 0.7) + (25ms \times 0.3) = 3.5ms + 7.5ms = 11.0ms$

b) Caching Strategy
→ Suggested Strategy : Client-Side Caching with Write-Back Policy.
→ Justification : Remote Access (25ms) is very expensive. Write Back caching minimizes remote access penalty by ~~sto~~ satisfying subsequent reads locally and batching writes before sending them to the server.

Ans8) Checkpoint optimization for RPO.
a) Optimal Mix Proposal (Over 10 secs)
The RPO is 1 second

Type	Count	Overhead (ms)
Full Checkpoint (FC)	2 (at 0s and 10s)	$2 \times 200 = 400$
Incremental (IC)	9 (every 1s in between).	$9 \times 50 = 450$
Total overhead		850 ms

b) Explanation of Reasoning
- RPO Constraint: An incremental checkpoint (IC) must run every 1 second.
- Minimal overhead: This mix maximizes the use of the low cost ICs (50ms) while high cost FCs (200ms)

Ans 9) E-Commerce Case Study (4)

a) Scheduling & Load Balancing

→ Challenge: Heterogeneous load and high synchronization delay

→ Algorithm: Receiver - Initiated Load Sharing.

→ Justification: Decentralized approach where underloaded sites ask for work, ~~gain~~ quickly smoothing flash sale spikes without centralized delay.

b) Fault Tolerance Strategy

→ Strategy: Active Replication using Active - Active Geo-Redundancy.

→ Impact: Low RPO: Synchronous / Near - Sync data replication ensures minimal / zero data loss.

Low RTO: Global Load Balancer immediately routes traffic from the failed region to the healthy, active region, ensuring instant service availability.

Sai
26/11/25