Name:- Ruin Antil
2301410011
BTech CSE (Cyber Security)

Ans-1 A race conditions occurs when 2 or more agents try to perform actions on a shared resource at the same time, and the final outcome depends on the timing of their actions

Real world example-
2 people trying to deposit & withdraw money from the ~~source~~ same bank account at the same time using offline forms. If both access the balance simultaneously Each might read the old balance, leading to an incorrect final account.

How mutual exclusion fixes it-
It allows 1 person at a time to update the shared resource. For Eg:- the bank processes one transactions at a time using a queue, ensuring correctness

Ans-2 Implementation complexity :-
Peterson's soln - simple conceptually, but difficult to implement correctly in modern architectures due to compiler reordering & caching

- Semaphores:- Slightly more complex logically, but easier to use in real programs and widely supported in OS libraries.

Ans-3 Monitors automatically handle mutual exclusion & condition synchronization, ensuring that only one thread accesses shared data at a time. In multi-core systems, monitors reduce the chance of low-level synchronization errors and simplify code, making concurrency safer & easier to manage.

Ans-4 How starvation occurs:-
If readers keep arriving continuously, writers may never get a chance to write. Similarly, a system giving high priority to writers may cause readers to starve.

Prevention method:-
Use fair scheduling - for eg:- a queue-based solution where readers and writers are served in the order they arrive. This ensures no thread is postponed indefinitely.

Ans-5 Eliminating hold and wait processes to either:-
1. Request all resources at the start, or
2. Release all resources before requesting new ones

Drawback:-
This leads to low resource utilization Processes often hold resources they are not using yet, causing other processes to wait unnecessarily, reducing overall system performance.

Ans-6 Banker's Algorithm Simulation

Total instances :- A=10, B=5, C=7.

| Process | Allocation | 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) Need = Max Allocation

| 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, ) |

(b) Calculate Available = Total Sum

Sum Allocation = (0+2+3+2+0, 1+0+0+1+0, 0+0+2+1+2) = (7, 2, 5)

Available = (10, 5, 7) − (7, 2, 5) = (3, 3, 2)

a) Interrupt per second = $\frac{\text{transfer rate}}{\text{block size}}$

$= \frac{512000}{100}$

$= 5120$ interrupts/s

CPU time per second = interrupts/second × time/interrupt

$= 5120 \times 5 \times 10^{-6}$ s

$= 0.02566$

$= 25.6$ m/s

(b) Use DMA, so device moves data to memory in larger bursts & interrupts only when large chuck complete this greatly reduces interrupt rate.

Ans-9 a) Critical-section :- Shared radar sensor data fusion queue, flight-plan database (shared world model, logging

Suitable IPC for real-time response: else shared memory + real-time locks, real time message queues

b) Detection :- Monitor resource allocation graph periodically in process.

Recovery with minimal disruption :-

1. Preemptive + Rollback
2. Selective almost of non-critical tasks

b) Caching strategy suggestion-

→ Suggested strategy : Client side LRU caching with write-back for read-heavy file & validation TTL.

Justification -

- LRU :- Works well in practice because file access pattern often how temporal locality - recently used file are likely to be ~~overused~~ reused.
- Write back with validations TTL :- For performance, write can be batches (write-back) but use short TTL or invalidation message to ensure consistency.

Ques-8 a) Proposed optimal mix

steps -

- Take one full checkpoint every 10s
- Take incremental checkpoint every 1s b/w full checkpoints.

# Total checkpoint overhead per 10s:

Fulls : 1 x 200ms = 200 ms

Incremental : 9 x 50ms = 450 ms

Total 650 ms per 10s

Average overhead = 65 m/s

classmate
Date
Page

Q4

Ans-9 a) Key Challenges

1) Sudden Bursty traffic: Requests spike order of magnitude within seconds.
2) Geographic latency & data locality: User should be served from the nearest region when possible
3) Hot-spots / skewed load :- Certain products / regions get disproportionate attention.
4) Fairness & SLA guarantees:- Mission critical request (payment) must get prioritized.

2) Durable logs & checkpointing
→ Maintain an append only transactions log with multi-region replication on failover, replay log to reach consistent state.
→ Combine with frequent checkpoint to bound recovery work.

3) Automatic Failover & Health Checks.
→ Global load balancer & control plane perform health probes & route traffic away from unhealthy region automatically.
→ Use traffic shaping & gradual failover to avoid overloading remaining regions.