## L2 Synchronization

1. What causes a race condition in concurrent programming?

A) Threads executing different functions

B) Non-atomic operations on shared variables by multiple threads

C) Using mutex locks improperly

D) Single-threaded program execution

Answer:

2. Which hardware primitive atomically sets a memory location to 1 and returns its previous value?

A) Compare-and-Swap

B) Fetch-and-Add

C) Test-and-Set

D) Load-Linked/Store-Conditional

Answer:

3. What problem arises if sem\_wait() and sem\_post() operations are nested within mutex locks in Producer/Consumer code?

A) Improved performance

B) Priority inversion

C) Deadlock

D) Memory leaks

Answer:

4. In the Readers/Writers problem, why might writers starve?

A) Writers have higher priority

B) New readers continuously acquire the lock before writers

C) Semaphores are initialized incorrectly

D) Mutex locks are not used

Answer:

5. What ensures fairness in ticket locks?

A) Test-and-Set instruction

B) Fetch-and-Add atomic operation

C) Compare-and-Swap

D) Disabling interrupts

Answer:

6. Why must pthread\_cond\_wait() be called in a while loop?

A) To prevent deadlock

B) To handle spurious wakeups

C) To improve performance

D) To enforce mutual exclusion

Answer:

7. Which synchronization primitive combines a mutex with condition variables?

A) Spinlock

B) Semaphore

C) Monitor

D) Ticket lock

Answer:

8. In the Dining Philosophers problem, deadlock occurs when:

A) All philosophers think simultaneously

B) Each philosopher holds one fork and waits for another

C) Philosophers use random delay before eating

D) An even number of philosophers exist

Answer:

9. What does sem\_init(&sem, 0, 3) indicate about the semaphore?

A) Binary semaphore for mutual exclusion

B) Counting semaphore allowing 3 concurrent accesses

C) Priority-based semaphore

D) Uninitialized semaphore

Answer:

10. What happens when sem\_wait() is called on a semaphore with value 0?

A) Returns immediately

B) Decrements the value to -1

C) Blocks until sem\_post() is called

D) Causes a segmentation fault

Answer:

11. Which condition variable operation wakes all waiting threads?

A) pthread\_cond\_signal()

B) pthread\_cond\_broadcast()

C) pthread\_cond\_wait()

D) pthread\_cond\_init()

Answer:

12. What is the key difference between Test-and-Set and Compare-and-Swap?

A) TAS modifies memory unconditionally; CAS checks expected value first

B) CAS uses fetch-and-add internally

C) TAS guarantees fairness

D) CAS only works for single-processor systems

Answer:

13. In the Producer/Consumer problem, the emptySlots semaphore is initialized to:

A) 0

B) 1

C) Buffer size

D) Number of threads

Answer:

14. What prevents starvation in the ticket lock implementation?

A) Random backoff

B) FIFO queue based on ticket numbers

C) Priority inheritance

D) Timeout mechanisms

Answer:

15. Which deadlock condition is resolved by enforcing resource ordering?

A) Mutual exclusion

B) Hold-and-wait

C) Circular wait

D) No preemption

Answer:

16. Why are spinlocks inefficient for long critical sections?

A) They use kernel scheduling

B) They cause busy waiting

C) They disable interrupts

D) They leak memory

Answer:

17. In Mesa-style monitors, what happens after pthread\_cond\_signal()?

A) Signaled thread immediately preempts others

B) Signaled thread joins a ready queue

C) All condition variables reset

D) Mutex automatically unlocks

Answer:

18. Which POSIX function initializes a mutex with default attributes?

A) pthread\_mutex\_create()

B) PTHREAD\_MUTEX\_INITIALIZER

C) pthread\_lock\_init()

D) sem\_init()

Answer:

19. What problem does a "room semaphore" solve in Dining Philosophers?

A) Limits concurrent philosophers

B) Enforces fork cleaning

C) Randomizes eating order

D) Increases table size

Answer:

20. Which synchronization method maintains state between signals?

A) Condition variables

B) Semaphores

C) Spinlocks

D) Mutexes

Answer: