

L3 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: B

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: C

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: C

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: B

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: B

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: B

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

- A) Spinlock
- B) Semaphore
- C) Monitor
- D) Ticket lock

Answer: C

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: B

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: B

10. What happens when `sem_wait()` is called on a semaphore with value 0?

- A) Returns immediately
- B) Increments the value to +1
- C) Blocks until `sem_post()` is called
- D) Causes a segmentation fault

Answer: C

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: B

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: A

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

- A) 0
- B) 1
- C) Buffer size
- D) Number of threads

Answer: C

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: B

15. 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: B

16. 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: B

17. 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: B

18. 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: A

19. Which synchronization method maintains state between signals?

- A) Condition variables
- B) Semaphores
- C) Spinlocks
- D) Mutexes

Answer: B

20. A counting semaphore initialized to N allows:

- A) Only one thread to access a resource
- B) Up to N threads to access a resource simultaneously
- C) Threads to bypass mutex locks
- D) Priority inversion to occur

Answer: B