Import Settings:
Base Settings: Brownstone Default
Highest Answer Letter: D
Multiple Keywords in Same Paragraph: No

Chapter: Chapter 6

Multiple Choice

1. A race condition ____.
A) results when several threads try to access the same data concurrently
B) results when several threads try to access and modify the same data concurrently
C) will result only if the outcome of execution does not depend on the order in which instructions are executed
D) None of the above

Ans: B
Feedback: 6.2
Difficulty: Medium

2. An instruction that executes atomically ____.
A) must consist of only one machine instruction
B) executes as a single, uninterruptible unit
C) cannot be used to solve the critical section problem
D) All of the above

Ans: B
Feedback: 6.4
Difficulty: Medium

3. A counting semaphore ____.
A) is essentially an integer variable
B) is accessed through only one standard operation
C) can be modified simultaneously by multiple threads
D) cannot be used to control access to a thread's critical sections

Ans: A
Feedback: 6.6.1
Difficulty: Medium

4. A mutex lock ____.
A) is exactly like a counting semaphore
B) is essentially a boolean variable
C) is not guaranteed to be atomic
D) can be used to eliminate busy waiting

Ans: B
Feedback: 6.5
Difficulty: Difficult

5. In Peterson's solution, the ____ variable indicates if a process is ready to enter its critical section.
A) `turn`
B) `lock`
C) `flag[i]`
D) `turn[i]`

Ans: C
Feedback: 6.3
Difficulty: Easy

6. The first readers-writers problem ____.
A) requires that, once a writer is ready, that writer performs its write as soon as possible.
B) is not used to test synchronization primitives.
C) requires that no reader will be kept waiting unless a writer has already obtained permission to use the shared database.
D) requires that no reader will be kept waiting unless a reader has already obtained permission to use the shared database.

Ans: C
Feedback: 6.7.2
Difficulty: Medium

7. A ___ type presents a set of programmer-defined operations that are provided mutual

exclusion within it.
A) transaction
B) signal
C) binary
D) monitor

Ans: D
Feedback: 6.8
Difficulty: Easy

8. _____________ occurs when a higher-priority process needs to access a data structure that is currently being accessed by a lower-priority process.
A) Priority inversion
B) Deadlock
C) A race condition
D) A critical section

Ans: A
Feedback: 6.6.4
Difficulty: Medium

9. What is the correct order of operations for protecting a critical section using mutex locks?
A) `release()` followed by `acquire()`
B) `acquire()` followed by `release()`
C) `wait()` followed by `signal()`
D) `signal()` followed by `wait()`

Ans: B
Feedback: 6.5
Difficulty: Easy

10. What is the correct order of operations for protecting a critical section using a binary semaphore?
A) `release()` followed by `acquire()`
B) `acquire()` followed by `release()`
C) `wait()` followed by `signal()`
D) `signal()` followed by `wait()`

Ans: C
Feedback: 6.6

Difficulty: Easy

11. _____ is not a technique for handling critical sections in operating systems.
A) Nonpreemptive kernels
B) Preemptive kernels
C) Spinlocks
D) Peterson's solution

Ans: D
Feedback: 6.3
Difficulty: Medium

12. A solution to the critical section problem does not have to satisfy which of the following requirements?
A) mutual exclusion
B) progress
C) atomicity
D) bounded waiting

Ans: C
Feedback: 6.2
Difficulty: Medium

13. A(n) _______ refers to where a process is accessing/updating shared data.
A) critical section
B) entry section
C) mutex
D) test-and-set

Ans: A
Feedback: 6.2
Difficulty: Medium

14. _____ can be used to prevent busy waiting when implementing a semaphore.
A) Spinlocks
B) Waiting queues
C) Mutex lock
D) Allowing the `wait()` operation to succeed

Ans: B
Feedback: 6.6.
Difficulty: Easy

15. Assume an adaptive mutex is used for accessing shared data on a Solaris system with multiprocessing capabilities. Which of the following statements is not true?
A) A waiting thread may spin while waiting for the lock to become available.
B) A waiting thread may sleep while waiting for the lock to become available.
C) The adaptive mutex is only used to protect short segments of code.
D) Condition variables and semaphores are never used in place of an adaptive mutex.

Ans: D
Feedback: 6.9.3
Difficulty: Medium

16. What is the purpose of the mutex semaphore in the implementation of the bounded-buffer problem using semaphores?
A) It indicates the number of empty slots in the buffer.
B) It indicates the number of occupied slots in the buffer.
C) It controls access to the shared buffer.
D) It ensures mutual exclusion.

Ans: D
Feedback: 6.7.1
Difficulty: Medium

17. How many philosophers may eat simultaneously in the Dining Philosophers problem with 5 philosophers?
A) 1
B) 2
C) 3
D) 5

Ans: B
Feedback: 6.7.3
Difficulty: Medium

18. Which of the following statements is true?
A) A counting semaphore can never be used as a binary semaphore.
B) A binary semaphore can never be used as a counting semaphore.
C) Spinlocks can be used to prevent busy waiting in the implementation of semaphore.
D) Counting semaphores can be used to control access to a resource with a finite number of instances.

Ans: C
Feedback: 6.6
Difficulty: Medium

19. _____ is/are not a technique for managing critical sections in operating systems.
A) Peterson's solution
B) Preemptive kernel
C) Nonpreemptive kernel
D) Semaphores

Ans: A
Feedback: 6.3
Difficulty: Medium

20. When using semaphores, a process invokes the `wait()` operation before accessing its critical section, followed by the `signal()` operation upon completion of its critical section. Consider reversing the order of these two operations—first calling `signal()`, then calling `wait()`. What would be a possible outcome of this?
A) Starvation is possible.
B) Several processes could be active in their critical sections at the same time.
C) Mutual exclusion is still assured.
D) Deadlock is possible.

Ans: B
Feedback: 6.7
Difficulty: Difficult

21. Which of the following statements is true?
A) Operations on atomic integers do not require locking.
B) Operations on atomic integers do require additional locking.
C) Linux only provides the `atomic_inc()` and `atomic_sub()` operations.
D) Operations on atomic integers can be interrupted.

Ans: A
Feedback: 6.9.2
Difficulty: Medium

22. A(n) ___________ is a sequence of read-write operations that are atomic.
A) atomic integer
B) semaphore
C) memory transaction
D) mutex lock

Ans: C
Feedback: 6.10.1
Difficulty: Medium

23. The OpenMP `#pragma omp critical` directive ___________.
A) behaves much like a mutex lock
B) does not require programmers to identify critical sections
C) does not guarantee prevention of race conditions
D) is similar to functional languages

Ans: A
Feedback: 6.10.2
Difficulty: Medium

24. Another problem related to deadlocks is ____________.
A) race conditions
B) critical sections
C) spinlocks
D) indefinite blocking

Ans: D
Feedback: 6.6.3
Difficulty: Medium

Essay

25. What three conditions must be satisfied in order to solve the critical section problem?

Ans: In a solution to the critical section problem, no thread may be executing in its critical section if a thread is currently executing in its critical section. Furthermore, only those threads that are not executing in their critical sections can participate in the decision on which process will enter its critical section next. Finally, a bound must exist on the number of times that other threads are allowed to enter their critical state after a thread has made a request to enter its critical state.
Feedback: 6.2
Difficulty: Medium

26. Explain two general approaches to handle critical sections in operating systems.

Ans: Critical sections may use preemptive or nonpreemptive kernels. A preemptive kernel allows a process to be preempted while it is running in kernel mode. A nonpreemptive kernel does not allow a process running in kernel mode to be preempted; a kernel-mode process will run until it exits kernel mode, blocks, or voluntarily yields control of the CPU. A nonpreemptive kernel is essentially free from race conditions on kernel data structures, as the contents of this register will be saved and restored by the interrupt handler.
Feedback: 6.2
Difficulty: Medium

27. Write two short methods that implement the simple semaphore `wait()` and `signal()` operations on global variable `S`.

Ans:
```
wait (S) {
  while (S <= 0);

  S--;

}



signal (S) {

  S++;

}
```

Feedback: 6.6
Difficulty: Difficult

28. Explain the difference between the first readers–writers problem and the second readers–-writers problem.

Ans: The first readers–writers problem requires that no reader will be kept waiting unless a writer has already obtained permission to use the shared database; whereas the second readers–writers problem requires that once a writer is ready, that writer performs its write as soon as possible.
Feedback: 6.7.2
Difficulty: Medium

29. Describe the dining-philosophers problem and how it relates to operating systems.

Ans: The scenario involves five philosophers sitting at a round table with a bowl of food and five chopsticks. Each chopstick sits between two adjacent philosophers. The philosophers are allowed to think and eat. Since two chopsticks are required for each philosopher to eat, and only five chopsticks exist at the table, no two adjacent philosophers may be eating at the same time. A scheduling problem arises as to who gets to eat at what time. This problem is similar to the problem of scheduling processes that require a limited number of resources.
Feedback: 6.7.3
Difficulty: Medium

30. What is the difference between software transactional memory and hardware transactional memory?

Ans: Software transactional memory (STM) implements transactional memory entirely in software, no special hardware is required. STM works by inserting instrumentation code inside of transaction blocks and typically requires the support of a compiler. Hardware transactional memory (HTM) implements transactional memory entirely in hardware using cache hierarchies and cache coherency protocols to resolve conflicts when shared data resides in separate caches.
Feedback: 6.10.1
Difficulty: Difficult

31. Assume you had a function named update() that updates shared data. Illustrate how a mutex lock named mutex might be used to prevent a race condition in update().

Ans:
```
void update()
{
     mutex.acquire();

     // update shared data

     mutex.release();
}
```

Feedback: 6.5
Difficulty: Easy

32. Describe the turnstile structure used by Solaris for synchronization.

Ans: Solaris uses turnstiles to order the list of threads waiting to acquire either an adaptive mutex or a reader–writer lock. The turnstile is a queue structure containing threads blocked on a lock. Each synchronized object with at least one thread blocked on the object's lock requires a separate turnstile. However, rather than associating a turnstile with each synchronized object, Solaris gives each kernel thread its own turnstile.
Feedback: 6.9.3
Difficulty: Difficult

33. Explain the role of the variable `preempt_count` in the Linux kernel.

Ans: Each task maintains a value `preempt_count` which is the number of locks held by each task. When a lock is acquired, `preempt_count` is incremented. When a lock is released, `preempt_count` is decremented. If the task currently running in the kernel has a value of `preempt_count` > 0, the kernel cannot be preempted as the task currently holds a lock. If the count is zero, the kernel can be preempted.
Feedback: 6.9.2
Difficulty: Difficult

34. Describe how an adaptive mutex functions.

Ans: An adaptive mutex is used in the Solaris operating system to protect access to shared data. On a multiprocessor system, an adaptive mutex acts as a standard semaphore implemented as a spinlock. If the shared data being accessed is already locked and the thread holding that lock is running on another CPU, the thread spins while waiting for the lock to be released, and the data to become available. If the thread holding the lock is not in the run state, the waiting thread sleeps until the lock becomes available. On a single processor system, spinlocks are not used and the waiting thread always sleeps until the lock becomes available.
Feedback: 6.9.3
Difficulty: Difficult

35. Describe a scenario when using a reader–writer lock is more appropriate than another synchronization tool such as a semaphore.

Ans: A tool such as a semaphore only allows one process to access shared data at a time. Reader–writer locks are useful when it is easy to distinguish if a process is only reading or reading/writing shared data. If a process is only reading shared data, it can access the shared data concurrently with other readers. In the case when there are several readers, a reader–writer lock may be much more efficient.
Feedback: 6.7.2
Difficulty: Medium

36. Explain how Linux manages race conditions on single-processor systems such as embedded devices.

Ans: On multiprocessor machines, Linux uses spin locks to manage race conditions. However, as spin locks are not appropriate on single processor machines, Linux instead disables kernel preemption which acquiring a spin lock, and enables it after releasing the spin lock.
Feedback: 6.9.2
Difficulty: Medium

True/False

37. Race conditions are prevented by requiring that critical regions be protected by locks.

Ans: True
Feedback: 6.4
Difficulty: Medium

38. The value of a counting semaphore can range only between 0 and 1.

Ans: False
Feedback: 6.6
Difficulty: Easy

39. A deadlock-free solution eliminates the possibility of starvation.

Ans: False
Feedback: 6.6.3
Difficulty: Medium

40. The local variables of a monitor can be accessed by only the local procedures.

Ans: True
Feedback: 6.8
Difficulty: Medium

41. Every object in Java has associated with it a single lock.

Ans: True
Feedback: 6.8
Difficulty: Medium

42. Monitors are a theoretical concept and are not practiced in modern programming languages

Ans: False
Feedback: 6.8
Difficulty: Easy

43. A thread will immediately acquire a dispatcher lock that is the signaled state.

Ans: True
Feedback: 6.9.1
Difficulty: Easy

44. Mutex locks and counting semaphores are essentially the same thing.

Ans: False
Feedback: 6.6
Difficulty: Easy

45. Mutex locks and binary semaphores are essentially the same thing.

Ans: True
Feedback: 6.6
Difficulty: Easy

46. A nonpreemptive kernel is safe from race conditions on kernel data structures.

Ans: True
Feedback: 6.2
Difficulty: Medium

47. Linux mostly uses atomic integers to manage race conditions within the kernel.

Ans: False
Feedback: 6.9.2
Difficulty: Medium