

COS40003 – Concurrent Programming

Correction Report: Question Test, Semester 2, 2025

Name: Nur E Siam

Student ID: 103842784

Question 9 (True/False Section)

Original Question:

“If a resource allocation graph contains a cycle, there will be a deadlock.”

Correct Answer: False

Explanation:

A cycle in a resource allocation graph indicates the possibility of a deadlock, but not a certainty.

- If each resource type has only one instance, then a cycle does imply a deadlock.
- However, if there are multiple instances of a resource type, a cycle can exist without causing deadlock.

Therefore, the correct answer is False, because a cycle is a necessary but not sufficient condition for deadlock.

Question 3 (Short Answer Section)

Original Question:

In Java, if a class A has a static variable b, how can you ensure a thread can safely access the static variable?

Correct Answer:

- Method 1: Use synchronized (A.class) to lock on the class object so only one thread can access the static variable at a time.
- Method 2: Use a static ReentrantLock to explicitly control access and ensure thread safe operations on the static variable.

Explanation:

The static variable belongs to the class, meaning all threads share it. To prevent multiple threads from accessing or modifying it at the same time, synchronizing on the class object (A.class) ensures mutual exclusion so that only one thread can access it at a time. Alternatively, a static ReentrantLock provides fine grained control over lock acquisition and release, avoiding interference and ensuring thread safe access to the shared variable.

Week 10 Test (Total Marks Capped at 40)

PART A. MULTIPLE-CHOICE (20 marks, single answer, please put answers in [])

- [A] 1. Concurrent computing can be considered that computing tasks are done simultaneously _____.
A. from observer's point of view B. from system's point of view
- [D] 2. Which of the following computing paradigm support parallel computing tasks? _____.
A. distributed computing B. clustering computing C. grid computing D. all of the above
- [C] 3. Which of the following process state transition cannot happen? _____.
A. ready->running B. running->ready C. blocked->running D. running->blocked E. none of the above
- [A] 4. In Unix, which system call can create a new process? _____.
A. fork B. create C. exec D. both A and C
- [C] 5. Which of the following system calls does not return control to the calling point after finished? _____.
A. fork B. wait C. exec D. none of the above
- [D] 6. Which statement correctly corresponds to the turnaround time of a job? _____.
A. the total waiting time for a process to finish the execution B. the total time spent in the ready queue
C. the total time spent in the running queue D. the total time from the submission till the completion
- [B] 7. Which of the following scheduling strategy is most likely to suffer from starvation? _____.
A. first come first serve B. preemptive shortest job first C. round robin D. multi-level feedback queue
- [B] 8. Which one of the following is not a valid state of a thread?
A. running B. parsing C. ready D. blocked
- [C] 9. Which of the following does a thread not own a private copy? _____.
A. program counter B. stack C. heap D. stack pointer E. none of the above
- [D] 10. The number of the threads in the pool can be decided on factors such as: _____.
A. number of CPUs B. amount of physical memory C. expected number of client requests D. all of the above
- [B] 11. A unit we do not want to interrupt is regarded as: _____.
A. single B. atomic C. static D. none of the above
- [D] 12. Which of the following is required by a good lock? _____.
A. can prevent multiple threads from entering a critical section B. low overhead added by using the lock
C. ensure each thread contending the lock gets a fair shot at acquiring it once it is free D. all of the above
- [D] 13. Which is the least important issue of implementing a lock by disabling interrupts? _____.
A. requires too much trust in applications B. does not work on multiple processors
C. causes other important interrupts ignored D. very inefficient when implemented
- [B] 14. The main disadvantage of spinlocks is that: _____.
A. they are not sufficient for many process B. they require busy waiting
C. they are unreliable sometimes D. they are too complex to implement for programmers
- [C] 15. Which hardware function or method can help implement a fair spin lock? _____.
A. test-and-set B. compare-and-swap C. fetch-and-add D. all of A,B,C E. none of A,B,C
- [A] 16. If processes P1, P2, and P3 are all requesting resource R, but the operating system allows P1 and P2 to access R repeatedly while P3 continues to wait for R, that is called: _____.
A. starvation B. livelock C. deadlock D. busy waiting
- [C] 17. Which of the following structure is the main part of a condition variable? _____.
A. a boolean variable B. an integer variable C. a queue D. a stack
- [A] 18. Which of the following needs a mutex/lock reference to be passed in? _____.
A. pthread_cond_wait() B. pthread_cond_signal() C. both D. neither
- [D] 19. Which of the following is not used to help with concurrent control? _____.
A. lock B. condition variable C. semaphore D. livelock
- [D] 20. In the Dining Philosopher problem, changing the order of one philosopher's fork-pickup actions, is to tackle which condition to prevent deadlock? _____.
A. mutual exclusion B. hold-and-wait C. no pre-emption D. circular wait E. none of the above

PART B. TRUE-FALSE (T/F) QUESTIONS (10 marks, please put answers in [])

- [T] 1. Round robin falls under the category of preemptive scheduling.
 - [T] 2. In theory, preemptive shortest job first scheduling can always have better average turnaround time than non-preemptive shortest job first.
 - [F] 3. In multi-level feedback queue, jobs of the same priority will be scheduled by first come first serve.
 - [T] 4. In Java, ReentrantLock can provide concurrency functionalities that synchronization blocks cannot provide.
 - [F] 5. If one thread of a process is holding the lock on a mutex, and another thread of the process attempts to lock the mutex, the whole process is blocked.
 - [F] 6. If there is no thread waiting on a condition variable, when pthread_cond_signal() is called, the signal will be remembered, and the next thread to wait on the condition variable will not block.
 - [F] 7. When using lock & condition variable for concurrency control, it is recommended to always signal() before unlock(&mutex), because, otherwise, it may cause deadlocks.
- We can use resource graph to help detect deadlocks.
- [T] 8. If a resource allocation graph has no cycles, there will be no deadlock.
 - [T] 9. If a resource allocation graph contains a cycle, there will be a deadlock.
 - [T] 10. Reboot is considered as an option to recover from deadlocks if deadlocks are rare.

PART C. SHORT-ANSWER QUESTIONS (10 marks, please put answers in _____)

1. What is the advantage of a shorter scheduling quantum? _____ Better responsiveness and _____ shorter response time for interactive jobs _____
2. What is the advantage of a longer scheduling quantum? _____ Lower context-switch overhead, better CPU efficiency _____
3. In Java, if a class A has a static variable b, how can you ensure a thread can safely assess the static variable? Method 1 _____ Guard with synchronized or a ReentrantLock. _____ Method 2 _____ Use volatile / Atomic (AtomicInteger, AtomicReference) _____
4. We often see code like “while (flag == 0) Pthread_cond_wait(&cond, &mutex)”. Here why it is better to use “while”, rather than “if (flag == 0) Pthread_cond_wait(&cond, &mutex)” _____ To re-check the predicate after wakeup (handles spurious wakeups and races so the condition truly holds before proceeding). _____
5. A semaphore is initialised as 2. Now its value is -1. This means how many threads are waiting? _____ 1 thread is waiting _____
6. What concurrency primitive tools can you use to fix atomicity violation bugs and order violation bugs? Atomicity violation _____ Use a mutex/lock or atomic instructions _____ Order violation _____ Use condition variables/semaphores/barriers to enforce ordering _____
7. Assume the following code will run to complete (ignoring syntax errors),
how many processes will be generated? _____ 24 (23 new + original) _____ how many lines will be printed? _____
First printf once and final printf by 24 processes = 25 lines _____

```
int main(void) {
    printf("this is one line\n");
    pid_t pid = fork();
    pid = fork();
    pid = fork();
    if (pid == 0)
        fork();
    fork();
    printf("this is one line\n");
}
```

PART D. SUDOKU (1 mark) – optional

	2							
		6						3
	7	4		8				
					5			2
	8			4			1	
6		3				7	8	
				1				
5				9				
						4		