**MIDTERM Examination #2**

**operating system concepts**

# 03-60-330-01 *Fall 2009*

### University of Windsor

# School of Computer Science

|  |
| --- |
| Please read carefully before you start1. This is a CLOSED book test; no notes, textbooks, calculators or computer aids are allowed.
2. PRINT your name legibly and clearly with your Student ID in the spaces indicated on the Scantron sheet.
3. You will be asked to sign your name once before leaving the exam room (sign-out). By doing so you state agreement to the terms below**I agree to the above terms and will neither receive nor give unauthorized help on this exam.**
4. All questions are multiple choice. Circle the single response which best answers each question.
5. Place all responses on the approved Scantron marking sheet using **PENCIL**.
6. You are not allowed to give or receive unauthorized help with your test. Any misconduct, as outlined by the Senate bylaw 31 article I, will be reported accordingly.
7. You have 75 minutes to complete this test.
8. You may keep this question paper following the examination. A photocopy of your Scantron sheet will be returned to you.
9. **The maximum mark is 40.**

Good Luck! |

ANSWER SHEET

All questions are Multiple Choice or True/False. In each Multiple Choice question, 4 responses are provided – you are to choose only one response which best answers the question. On the Scantron sheet provided, carefully fill in the circle corresponding to the response letter A, B, C or D (one only).

If an error is made you must carefully erase your response on the Scantron sheet, and then fill in the circle you intend to choose.

|  |  |
| --- | --- |
| 1. | Two short methods that implement the simple semaphore wait() and signal() operations on global variable S include:signal (S) { S++;}and \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ . |
| A) | wait (S) { while (S <= 0); S--;} |
| B) | wait (S) { while (S >= 0); S--;} |
| C) | wait (S) { S--; while (S <= 0);} |
| D) | None of these are correct solutions. |

|  |  |
| --- | --- |
| 2. | Race conditions are prevented by requiring that critical regions be protected by \_\_\_\_\_\_\_\_ . |
| A) | clocks |
| B) | semaphores |
| C) | locks |
| D) | monitors |

|  |  |
| --- | --- |
| 3. | The local variables of a monitor can be accessed by \_\_\_\_\_\_\_\_\_\_\_ . |
| A) | only the operating system kernel threads |
| B) | only the local procedures |
| C) | signals |
| D) | licensed developers |

|  |  |
| --- | --- |
| 4. | 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. |

|  |  |
| --- | --- |
| 5. | The second 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, once a writer is ready, that writer performs its write as soon as possible. |

NOTE: Either A or D are correct. Noted in class to ignore D, however.

|  |  |
| --- | --- |
| 6. | A deadlocked state occurs whenever \_\_\_\_. |
| A) | a process is waiting for I/O to a device that does not exist |
| B) | the system has no available free resources |
| C) | every process in a set is waiting for an event that can only be caused by another process in the set |
| D) | a process is unable to release its request for a resource after use |

|  |  |
| --- | --- |
| 7. | One necessary condition for deadlock is \_\_\_\_, which states that at least one resource must be held in a non-sharable mode. |
| A) | hold and wait |
| B) | mutual exclusion |
| C) | circular wait |
| D) | no preemption |

|  |  |
| --- | --- |
| 8. | In a system resource-allocation graph, \_\_\_\_. |
| A) | a directed edge from a process to a resource is called an assignment edge |
| B) | a directed edge from a resource to a process is called a request edge |
| C) | a directed edge from a process to resource is called a request edge |
| D) | None of the above |

|  |  |
| --- | --- |
| 9. | A cycle in a resource-allocation graph is \_\_\_\_. |
| A) | a necessary and sufficient condition for deadlock in the case that each resource has more than one instance |
| B) | a necessary and sufficient condition for a deadlock in the case that each resource has exactly one instance |
| C) | a sufficient condition for a deadlock in the case that each resource has more than once instance |
| D) | is neither necessary nor sufficient for indicating deadlock in the case that each resource has exactly one instance |

|  |  |
| --- | --- |
| 10. | Which of the following is most often used by operating systems to handle deadlocks? |
| A) | Pretend that deadlocks never occur |
| B) | Use protocols to prevent or avoid deadlocks |
| C) | Detect and recover from deadlocks |
| D) | None of the above |

|  |  |
| --- | --- |
| 11. | Which of the following statements is true? |
| A) | A safe state is a deadlocked state. |
| B) | A safe state may lead to a deadlocked state. |
| C) | An unsafe state is necessarily, and by definition, always a deadlocked state. |
| D) | An unsafe state may lead to a deadlocked state. |

|  |  |
| --- | --- |
| 12. | Suppose that there are 10 resources available to three processes. At time 0, the following data is collected. The table indicates the process, the maximum number of resources needed by the process, and the number of resources currently owned by each process. Which of the following correctly characterizes this state?Process Maximum Needs Currently Owned P0 10 4 P1 3 1 P2 6 4 |
| A) | It is safe. |
| B) | It is not safe. |
| C) | The state cannot be determined. |
| D) | It is an impossible state. |

|  |  |
| --- | --- |
| 13. | The circular-wait condition for a deadlock implies the hold-and-wait condition.  |
| A) | True |
| B) | False |

|  |  |
| --- | --- |
| 14. | If a resource-allocation graph has a cycle, the system must be in a deadlocked state. |
| A) | True |
| B) | False |

|  |  |
| --- | --- |
| 15. | In some circumstances, a system can be in a frozen state but not in a deadlocked state. |
| A) | True |
| B) | False |

|  |  |
| --- | --- |
| 16. | In order to solve the critical section problem it is necessary to satisfy the condition that \_\_\_\_\_\_\_\_ . |
| A) | A thread may be executing in its critical section if another thread is currently executing in its critical section. |
| B) | Only those threads that are executing in their critical sections can participate in the decision on which process will enter its critical section next.  |
| C) | 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. |
| D) | All of the above. |

|  |  |
| --- | --- |
| 17. | \_\_\_\_\_\_\_ refers to the situation where, for a set of processes, every process in the set must be waiting for an event that can be caused only be another process in the set. |
| A) | Deadlock |
| B) | Starvation |
| C) | Locking |
| D) | Blocking |

|  |  |
| --- | --- |
| 18. | One way to ensure that a circular-wait condition never holds is to \_\_\_\_\_\_\_\_\_ ? |
| A) | apply a deadlock prevention policy. |
| B) | impose a total ordering of all resource types and to require that each process requests resources in an increasing order of enumeration.  |
| C) | assign each resource type a unique integer number to distinguish those occurring at the same time in the ordering. |
| D) | None of these responses is correct. |

|  |  |
| --- | --- |
| 19. | A system has two types of resources, R1 and R2, both of which have two instances of their respective resource types. There are four processes competing for resources. Assume that the following system state exists:P1 : R1 is requested, R2 is allocatedP2 : R1 is allocatedP3 : R1 is allocated, R2 is requestedP4 : R2 is allocatedThe system state \_\_\_\_\_\_\_ . |
| A) | contains a cycle, and it is deadlocked |
| B) | contains a cycle but it is not deadlocked |
| C) | contains no cycles and is not deadlocked |
| D) | contains no cycles and is deadlocked |

|  |  |
| --- | --- |
| 20. | 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 |

|  |  |
| --- | --- |
| 21. | 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 |

|  |  |
| --- | --- |
| 22. | A semaphore \_\_\_\_. |
| A) | can be modified simultaneously by multiple threads  |
| B) | is accessed through only one standard operation |
| C) | is essentially an integer variable |
| D) | cannot be used to control access to a thread's critical sections |

|  |  |
| --- | --- |
| 23. | A spinlock \_\_\_\_. |
| A) | is never advantageous for thread scheduling management |
| B) | will ultimately result in a context switch when a process must wait on a lock |
| C) | does not require a context switch when a process must wait on a lock |
| D) | is useful when locks are expected to be held for long amounts of time |

|  |  |
| --- | --- |
| 24. | 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] |

|  |  |
| --- | --- |
| 25. | A(n) \_\_\_ type presents a set of programmer-defined operations that are provided mutual exclusion within the set. |
| A) | transaction |
| B) | signal |
| C) | binary |
| D) | monitor |

|  |  |
| --- | --- |
| 26. | A transaction \_\_\_\_. |
| A) | performs multiple logical functions |
| B) | is a single instruction |
| C) | is a single operation |
| D) | performs a single logical function |

|  |  |
| --- | --- |
| 27. | Suppose that there are 15 resources available to three processes. At time 0, the following data is collected. The table indicates the process, the maximum number of resources needed by the process, and the number of resources currently owned by each process. Which of the following correctly characterizes this state?Process Maximum Needs Currently Owned P0 15 10 P1 5 3 P2 10 7 |
| A) | It is safe. |
| B) | It is not safe. |
| C) | The state cannot be determined. |
| D) | It is an impossible state. |

|  |  |
| --- | --- |
| 28. | A claim edge of a resource allocation graph \_\_\_\_\_\_\_\_\_\_ . |
| A) | is identical to a request edge |
| B) | resembles an allocation edge in direction but is represented in the graph by a dashed line |
| C) | indicates that a process may request a resource at some time in the future |
| D) | requires that the claiming process be given high priority in scheduling |

|  |  |
| --- | --- |
| 29. | Which of the following statements is true? |
| A) | A safe state is a deadlocked state. |
| B) | A safe state may lead to a deadlocked state. |
| C) | An unsafe state is necessarily, and by definition, always a deadlocked state. |
| D) | An unsafe state may lead to a deadlocked state. |

NOTE: This question is identical to Q11 – you should not have answered this, but will not be penalized twice if you did.

|  |  |
| --- | --- |
| 30. | Which of the following scheduling algorithms must be nonpreemptive? |
| A) | SJF |
| B) | RR |
| C) | FCFS |
| D) | priority algorithms |

|  |  |
| --- | --- |
| 31. | The \_\_\_\_ scheduling algorithm is designed especially for time-sharing systems. |
| A) | SJF |
| B) | FCFS |
| C) | RR |
| D) | Multilevel queue |

|  |  |
| --- | --- |
| 32. | Which of the following is true of multilevel queue scheduling? |
| A) | Processes can move between queues. |
| B) | Each queue has its own scheduling algorithm. |
| C) | A queue cannot have absolute priority over lower-priority queues. |
| D) | It is the most general CPU-scheduling algorithm. |

|  |  |
| --- | --- |
| 33. | One way to ensure that a circular-wait condition never holds is to \_\_\_\_\_\_\_\_\_ ? |
| A) | apply a deadlock prevention policy. |
| B) | assign each resource type a unique integer number to distinguish those occurring at the same time in the ordering. |
| C) | impose a total ordering of all resource types and to require that each process requests resources in an increasing order of enumeration.  |
| D) | None of these responses is correct. |

NOTE: This question is identical to Q18 (except for order of responses). You should not have answered this question, but if you did, there is no penalty.

|  |  |
| --- | --- |
| 34. | The Producer-Consumer problem is related to \_\_\_\_\_\_\_\_\_ . |
| A) | the handling of process state queues  |
| B) | the scheduling of process states |
| C) | the allocation of resources to process states |
| D) | Both A and C are correct answers. |

|  |  |
| --- | --- |
| 35. | A process control block should contain \_\_\_\_\_\_\_\_\_\_ . |
| A) | the process ID  |
| B) | locations to store register values |
| C) | a list of all open files |
| D) | All of these responses are correct |

|  |  |
| --- | --- |
| 36. | Protocols to prevent hold-and-wait conditions typically also prevent starvation. |
| A) | True |
| B) | False |

|  |  |
| --- | --- |
| 37. | A deadlock-free solution eliminates the possibility of starvation. |
| A) | True |
| B) | False |

|  |  |
| --- | --- |
| 38. | Turnaround time refers to the amount of time \_\_\_\_\_\_ . |
| A) | that CPU utilization is minimized |
| B) | to execute a particular process |
| C) | a process has been waiting in the ready queue |
| D) | it takes from when a request was submitted until the first action is produced |

|  |  |
| --- | --- |
| 39. | A thread does not share with its peer threads its \_\_\_\_\_\_ |
| A) | code section |
| B) | data section |
| C) | operating-system resources |
| D) | semaphore |

|  |  |
| --- | --- |
| 40. | Synchronization of message passing between processes is assured by using \_\_\_\_\_\_ |
| A) | buffering |
| B) | queuing |
| C) | blocking |
| D) | mutexing |