New York University

CSCI-UA.0202-003: Operating Systems (Undergrad): Fall 2025

Quiz 4

- Write your full name on both:
  - the bubble sheet in the “Name” field
  - the quiz booklet
- Write your NYU NetID on the quiz booklet and the bubble sheet in the “ID” field
- Use a #2 pencil to fill in your answers on the bubble sheet (preferred, but you can also use a pen)
- This quiz contains 6 questions only. Each question has choices from A to D
- Fill the bubbles completely by darkening the entire circle, as shown in the example
- Only mark answers for questions 1-6. Do not mark any bubbles beyond question 6
- Choose only one answer per question
- Submit your bubble sheet **together with your quiz booklet**

Name:

NetId:

1. The software bug that caused the Tyler, Texas accidents was a race condition. This overdose occurred because:
(A) A hardware microswitch failed, sending an ambiguous position message to the computer.
(B) An 8-bit variable used as a counter overflowed, causing a safety check to be bypassed.
(C) An operator corrected a data entry error for the treatment mode very quickly (within 8 seconds), causing machine parameters to be set incorrectly.
(D) The operator pressed the "P" key to proceed after a malfunction, which was against operating procedures.

2. For a deadlock to occur, four necessary conditions must be met. Which of the following is one of these four conditions?
(A) Priority Inversion.
(B) Hold and Wait.
(C) Preemptive Scheduling.
(D) Starvation.

3. The software bug responsible for the second Yakima accident in 1987 involved a shared variable named `Class3`. What was the nature of this flaw?
(A) `Class3` was not protected by a lock, allowing two threads to write to it simultaneously.
(B) `Class3` was a 1-byte variable that was repeatedly incremented, and on the 256th pass, it would overflow to zero, causing a critical collimator position check to be skipped.
(C) The operator would manually enter an incorrect value for `Class3` in a service mode, which would persist into treatment mode.
(D) `Class3` would be set correctly, but a different task would overwrite its value before the safety-checking task could read it.

4. What is the primary trade-off when choosing between coarse-grained and fine-grained locking?
(A) Coarse-grained locking has lower run-time overhead per lock operation, while fine-grained locking avoids starvation but has higher overhead.
(B) Coarse-grained locking is simpler to implement but limits available parallelism, while fine-grained locking allows more parallelism but is significantly more complex to design correctly.
(C) Coarse-grained locking makes deadlocks less likely, while fine-grained locking can eliminate deadlocks entirely if all locks are acquired in a globally-defined order.
(D) Coarse-grained locking protects larger amounts of data, making it more secure against data races, while fine-grained locking is less secure due to the increased number of critical sections.

5. In low-level programming for multi-core systems, what is the purpose of a memory barrier instruction (like `MFENCE`)?
(A) To ensure a specific memory write is atomic, preventing it from being "torn" into multiple, smaller writes by the hardware.
(B) To immediately flush all modified cache lines associated with a process to main memory, ensuring data persistence in case of a crash.
(C) To prevent the compiler from reordering memory access instructions around the barrier, serving a similar purpose as the `volatile` keyword in C/C++.
(D) To enforce an ordering constraint on the CPU, ensuring that memory operations before the barrier appear to other CPUs to have executed before memory operations after the barrier.

6. Which of the following scheduling scenarios is ONLY possible on a system using a preemptive scheduler?
(A) A process issues an I/O request and moves to a waiting state, allowing the next process in the ready queue to run.
(B) A long-running batch job runs to completion without interruption after it is dispatched to the CPU.
(C) A running process is moved back to the ready queue by the operating system as a result of an interrupt, allowing another process to run.
(D) A process executes for a short time and then terminates, allowing the scheduler to select a new process to run.