New York University

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

Quiz 8

- 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. Which of the following best explains why file system operations are not naturally atomic regarding disk persistence?
   (A) Modern hard drives cannot guarantee the atomicity of even a single sector write.
   (B) File system metadata and data are often spread across multiple distinct sectors or blocks.
   (C) The operating system disables write-back caching during critical updates, causing fragmentation.
   (D) Inodes and bitmaps are stored in the same sector, causing race conditions during updates.

2. Which of the following correctly contrasts Redo logging and Copy-on-Write regarding block overwrite?
   (A) Both techniques rely on overwriting blocks in place.
   (B) Neither technique ever overwrites blocks in place.
   (C) Redo logging never overwrites blocks; CoW overwrites blocks in place.
   (D) Redo logging overwrites blocks in place after checkpointing; CoW never overwrites blocks.

3. How does the NTFS file system utilize a combination of Redo and Undo logging to manage memory?
   (A) It uses Redo logging for data and Undo logging for metadata.
   (B) It uses Undo logs to allow early flushing and Redo logs for fast recovery.
   (C) It uses Undo logging during startup and Redo logging during normal operation.
   (D) It writes Undo logs to RAM and Redo logs to the disk.

4. Which system call is the primary mechanism used by a debugger to control, observe, and modify a target process on Linux?
   (A) `ioctl`
   (B) `setjmp`
   (C) `ptrace`
   (D) `mmap`

5. When a debugger needs to "step over" a software breakpoint it has just hit, what sequence of operations must it perform?
   (A) Jump over the instruction -> Continue execution.
   (B) Restore original instruction -> Single-step -> Re-insert breakpoint -> Continue.
   (C) Write NOP -> Single-step -> Write INT 3 -> Continue.
   (D) Move the instruction pointer (RIP) forward by one byte -> Continue.

6. If a debugger wants to perform a "backtrace" (stack unwind) on x86-64 without external debug info, which register is most critical to follow the chain of stack frames?
   (A) `%rip`
   (B) `%rsp`
   (C) `%rbp`
   (D) `%rax`