OS W8L1 - Midterm Review

Class	Operating Systems
Created	@Oct 19, 2020 5:34 PM
Materials	07 - Midterm Revision.pdf
Reviewed	☐
Type	Lecture

Exam Admin

- If anything is ambiguous, state your assumptions! Z said he'd try to make it as clear as possible since we *cannot ask questions during the exam*
- Exam opens at 9:30am ET on October 31st (lecture time) and submission is required **in PDF form** 24 hours later
    - Just as homeworks and labs showed up as assignments for download, the midterm will be the same
    - You can hand write or type your exam, just make it clear and legible
- Many questions can have multiple right answers, so back your explanation
- Make answers "laser focused" → Keep them short, concise, and accurate; no irrelevant answers or information in your answers
- Solutions are key, concept of question is fairly secondary
- Submission will be PDF via NYU Classes
- Exam designed to take about 2 hours
- There will be penalties for wrong answers and irrelevant information added

    - Do not add context to your question for information that is not asked for. For example, if you're asked if there's a deadlock, don't define deadlocks and processes prior to your answer

Review Problems

- Problem 1
    - Four conditions are a necessity for a deadlock, but can still be met and not cause a deadlock
    - One process with one thread can not request more than one resource
- Problem 2
    - Deadlocks are very bad with even a small few processes involved since other resources may be relying on them
    - Resource doesn't need to be in the chain to be deadlocked
- Problem 3
    - Every process has at *least* one thread (even if created, but not executed, it still has a thread)
    - Every thread always has 1 stack and a program counter (the program counter is required for each stack). For heaps, there is 1 per process
- Problem 4
    - Can't go from ready to blocked since there is nothing being executed (and thus you can't block a process that is not running)
- Problem 5
    - Blocked to ready is a state change that occurs *not* in execution
    - OS is the only thing that can change a process state. For example, if done via a child, the OS is the one told to change the state and thus changes it — The child itself does not change the parent state.
- Problem 6
    - No way to finish because the max value of x is 5

    - Resource 3 isn't even relevant → Resource 5 won't allow us to finish Process A so we don't really need to calculate x
- Problem 7
    - Very slow → Threads in different processes
    - Slow → Threads in kernel level and same process
    - Fast → Threads in same process on the user level