CSCI 315 Operating Systems Design
Midterm Exam 1 Study Guide

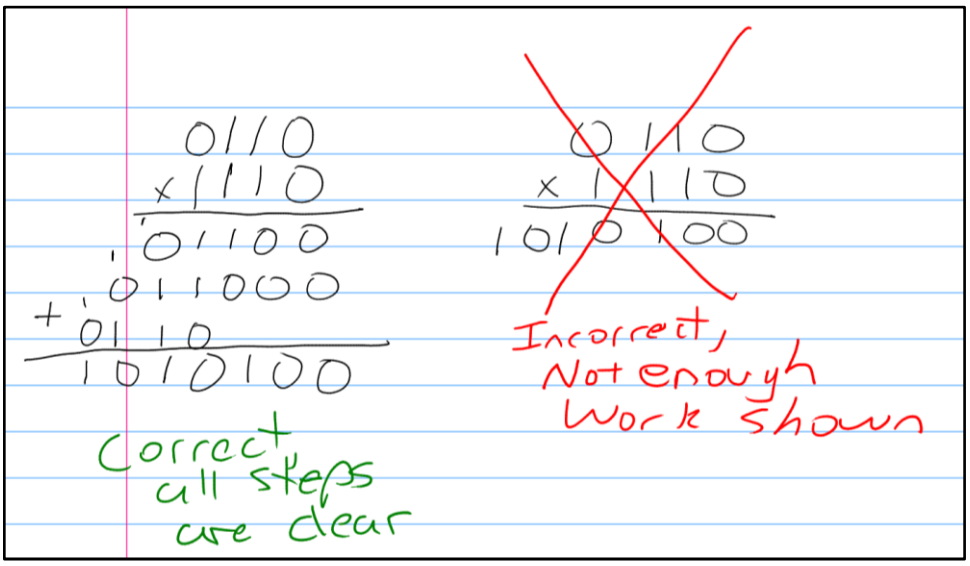
Fall 2021
Prof Meng

Exam Instructions

This exam is timed. You must complete the exam in a 180-minute duration. The timer will start when you click the button and download the exam file from Moodle. But you can do it any time during the given 50-hour period. The exam will be available on Moodle at **9:00 am Wednesday, October 6th**. And the exam will be due at **11:00 am Saturday, October 9th**. **Make sure you take the trial exam** first to experience how the system works, if you haven't done timed exams on Moodle before.

The exam on Moodle will be given in two versions, a Word and a PDF. **Please make sure only submitting a PDF file**. If you take a photo of your exam, make sure convert the image to PDF before submission. Many free conversions are available on smartphones or over the internet. I will feed the exams through Gradescope for grading, which needs PDF files.

The exam will be open notes and open book with a specified amount of time to complete. As such, when there is any calculation, or code comprehension, or writing of any code in the exam, you must show the detailed steps you followed to arrive at your answer. For example, below shows two ways to compute 6 x 14 in binary. One will receive full credit, the other will receive no credit. When in doubt, error on the side of showing more work than you think you need. (This is a random sample question, not CSCI 315 question.)



You must work on your own. Do not discuss the exam with anyone else. The instructor will be available online to answer any questions.

Exam Study Guid

- Review all assigned readings from the textbook and external readings.
  - SGG (10th edition): 1.1 through 1.6, 2.1 through 2.7, 3.1 through 3.4, 4.1 through 4.4, 4.6, 5.1 through 5.3, 6.1 through 6.7, 7.1
  - SGG (9th edition): 1.1 through 1.9, 2.1 through 2.6, 3.1 through 3.6, 4.1 through 4.4, 4.6, 5.1 through 5.8, 6.1 through 6.3.

- Go through labs, quizzes, and activities. Make sure that you have a solid understanding of the topics they address. The relevant C programming is an important part of this phase of the course. While we do not expect students to memorize the details of system calls and functions, the concepts and their basic C syntax are required.

- This document doesn't mean to give an exhaustive coverage of what might appear in the exam, but it will be useful as a self-check list for your preparation.

---

1. Identify the following concepts.
   - operating system
   - multiprogramming
   - time sharing/multitasking
   - process
   - thread
   - PCB
   - context switch
   - thread-safe library calls
   - scheduling
   - dual mode operation
   - mode bit
   - memory protection
   - starvation
   - CPU burst
   - I/O burst
   - race condition
   - critical region (critical section)
   - mutual exclusion
   - atomicity
   - busy waiting
   - test-and-set
   - compare-and-swap
   - semaphore (counting semaphore; binary semaphore/mutex)
   - inter-process communication (IPC) with pipes and sockets
   - shared memory
   - deadlock

2. Describe the "big picture" purpose of an operating system.

3. In what regards the design decisions that drove the creation of UNIX system calls and library functions:
   • What motivates the "need" for these two levels of service (system calls vs library functions)?

4. Explain how multiprogramming might improve the performance of an operating system.

5. Given an algorithm to solve the critical section problem, identify whether it meets the three fundamental requirements of mutual exclusion, progress, and bounded waiting. (Be ready to explain the concept behind each of these requirements.)

6. Construct a scenario in which two process that share a common memory location might run into execution problems (*hint:* race conditions).

7. What is the difference between *user mode* and *kernel mode*? How does the system keep track of which mode it's operating in?

8. What information about processes does the operating system need to store in order to implement multiprogramming? Where is this information stored?

9. What are the different types of inter-process communication? How do they work?

10. What are the steps an operating system takes to handle an interrupt?

11. List the states that a process may be in. Draw a diagram showing the possible transitions between these states and identifying what causes state transitions.

12. Contrast UNIX process and POSIX Pthread threads. Discuss similarities and differences. Identify how a programmer might choose between the two mechanisms to implement concurrent programs.

13. Consider the system calls of **fork()**, **exec()**, **exit()**. Identify their functionality. Describe the effect of one or more of them in the execution of a multi-threaded program.

14. Explain how the **pipe()** system call works. Show how a programmer creates pipes to enable the communication between two processes.

15. Explain how to use UNIX manual pages (**man**) to find how to use system calls or library functions. When issuing a **man** command, what are the common parameters you need to supply? What do they mean?

16. Explain differences and similarities between pipes and TCP sockets.

17. What happens during a context switch? Why can frequent context switches be a problem?

18. How can one estimate the length of CPU bursts so that this information can be used in scheduling of processes and/or threads?

19. Explain how a semaphore works. Why do semaphore variables assume different initial values? Explain your answer in the context of the Consumer-Producer Problem.

20. What are the classic process/thread synchronization problems we discussed in lecture and lab? Describe briefly the differences among those problems and be prepared to outline solutions for them.

21. Explain how each of the following scheduling algorithms works. Be able to discuss how each one affects CPU utilization, job throughput, turnaround time, waiting time, and response time. Discuss how preemption might apply to each of the algorithms and what effects it might produce on the performance of the system.
    - first come, first served
    - shortest job first
    - shortest remaining time first
    - round robin
    - priority
    - multilevel queue

22. How does the use of preemption and priority levels affect the behavior and the performance of the scheduling algorithms above?

23. Given a set of process with specified arrival times and CPU burst lengths, for one or more of the scheduling algorithms:
    - draw a Gantt chart of the execution of the processes,
    - calculate performance metrics of the scheduling algorithm,
    - apply the performance metrics to reason on how well the scheduler works and propose alternative scheduling for the given scenario.

24. What are the various criteria we use to study the performance of a CPU scheduling algorithm?