CMS 330 Final Exam Study Guide, Spring 2019

General overview:

- The exam will be about 25% material from before the midterm and about 75% after the midterm.
- You will have the full 2 hour exam period for the exam, but the exam will be designed out of 100 points (which equates to about an hour and 40 minutes in my system of 1pt == 1min).
- No calculators, closed book, etc.
- The most commonly missed questions on the midterm were #2, #5, and #6. I recommend you begin by studying those.
- You should review your labs in addition to your homeworks/projects. All of these are fair game, even if they were not graded.

Topics:

- Concurrency/multi-threaded programs
- What are the similarities and differences between threads and processes? In particular, think about address spaces & the stack(s).
- What is atomicity and why is it necessary?
- Locks, different types and implementations
- Define and understand critical regions, race conditions, and mutual exclusion
- Write a pseudocode description of the test-and-set instruction. Show how to use test-and-set to write a working implementation of a spin-based lock.
- Be able to look at code (for example, of a data structure implementation), identify the critical region, and add locks to implement mutual exclusion.
- What is deadlock and give a pseudocode example.
- Describe the four conditions that must exist if deadlock is possible.
- Describe strategies for removing one or more of the precipitating conditions for deadlock.
- Describe, understand, and know how to solve the Dining Philosophers problem. Be able to describe tradeoffs amongst different correct solutions.
- Describe and understand semaphores
- Write pseudocode definitions of the sem_init, sem_wait, and sem_post methods.
- Know how to use a semaphore to: implement a lock, use wait and post to make one thread stop and wait until it receives a signal from another thread, synchronize access to a shared buffer between writer and reader threads (the correct initialization values are important in each case; pay attention to initialization values for semphores!).
- What is a “page”?
- Explain and give pros/cons of: base & bounds, segmentation, paging. Explain the relationship between the virtual address space and physical memory for each of them.
- Explain the purpose of the TLB and swap space.
- Be able to perform basic paging calculations (see project 4)
- Explain how hard links and soft links are implemented by the file system.

- Explain how cp, rm, and mv interact with system calls and the file system (inode numbers, etc).
- Why can't you hard link directories?
- Draw the layout diagram of our very simple file system (VSFS).
- Know how to map basic file system operations like opening, reading, and writing onto the on-disk structures. For example, list the sequence of on-disk elements that must be accessed to open the file: /foo/baz.txt.
- Suppose you have an inode with 12 direct pointers, 1 indirect pointer, 1 doubly indirect, and 1 triply indirect pointer. The data block size is 4 KB and data block addresses are 4 bytes in size. What is the maximum file size that can be addresses on this system?
- What is a socket?
- List the important system calls required to initialize client and server side sockets. Related to this: draw the diagram showing the TCP client-server socket interactions.
- You DO NOT need to memorize the actual code required to initialize sockets, but you should be able to write code to read and write from, them once they're created (review the sockets lab)
- Explain what is meant by "everything is file" in Unix systems.
- Explain the client-server architecture.
- What are the advantages of structuring the Internet as a de-centralized network?
- What is packet-switching and what are its advantages over "whole" transmission of files/resources?
- A user types in a URL like www.rollins.edu in their web browser and eventually sees the page rendered in the same browser. Using what you know of **BOTH** the internet and the web, explain what must happen for the original request to be processed and transformed into the fully rendered page. (This will be a very long and detailed answer. Think carefully about each step.)
- List and summarize the six principles of REST system design.
- During this class we have often talked about virtualizing things (virtualizing memory, virtualizing the CPU, etc). Explain how virtualization plays a crucial role in OS functionality
- During this class we have often talked about separating the mechanism by which something is accomplished from the policy of why something happens. Give several examples of this separation that we have talked about this semester, and discuss why this separation plays a crucial role in OS functionality.