Name:

Operating Systems (COP 6611)
Spring 2012
Final Exam

Closed books, notes, cell phones, PDAs, iPods, laptops, etc. No headphones, please. You do not need a calculator.

You have 120 minutes to solve 7 problems for a total of 90 points. You receive a final-exam-week gift of 10 points just by signing your name on your exam paper. Total points: 100.

As with any exam, you should read through the questions first and start with those that you are most comfortable with. Answers will be graded for substance and clarity. Redundancy and vague generalities are considered bad form. Partial credit will be offered for meaningful progress towards solving the problems.

1	
2	
3	
4	
5	
6	
7	
Total	

1. (15 points) **Short attention span:** Answer the following questions (3 points each) on general concepts in operating systems.

    a. Compare the overhead of context switching between two threads in different processes and two processes. Explain.

    b. Can multiprogramming be useful in a single-user system? Explain your answer.

    c. Which of the following instructions should be privileged?
        1. Set value of timer
        2. Read the clock
        3. Clear memory
        4. Turn off interrupts
        5. Switch from user to kernel mode
        6. Access I/O device

    d. What is the purpose of system calls?

    e. In a system with threads, is there normally one stack per thread or one stack per process? Explain.

2. (15 points) Distributed File Systems:
    a. Present the GFS consistency model and how it was achieved through design.
    b. Compare GFS consistency model and design with that of AFS/Ceph (choose whichever file system you prefer).

3. (10 points) Scheduling:
   a. Compare and contrast Round Robin scheduling with Shortest Job First (SJF or SRTF) scheduling. Briefly discuss the strengths and weaknesses of each scheme with respect to the usual goals of a CPU scheduler. Why do most modern CPU schedulers combine Round Robin and SJF by giving CPU priority to I/O bound jobs?
   b. What is priority inversion, and why is it bad? Illustrate with an example. Use your example to illustrate one technique that avoids priority inversion.

4. (10 points) Under what circumstances does the wait-die scheme perform better than the wound-wait scheme for granting resources to concurrently executing transactions?

5. (10 points) Consider a file system that supports contiguous, linked and indexed allocations. What criteria should be used in deciding which strategy is best utilized for a particular file?

6. (15 points) In the Unix operating system, access control for files is enforced when a file is opened (e.g., in the open call which specified read or write mode). Access to files is controlled based on read/write/execute bits that are stored by the operating systems for each file. Once a file is opened successfully, reads and writes are processed without performing further checks. Would you characterize the Unix file access control scheme based on access control lists (ACLs) or capabilities, or a combination of the two? Explain your answer.

7. (15 points) True or False[1]? Support your answer with a brief statement.

Synchronization. Define a race as follows: a race exists if and only if two conflicting accesses to some shared variable occur concurrently (i.e., neither happened-before the other). The following statements pertain to multithreaded programs using mutexes to eliminate races.

a. If two executions of a race-free program on the same input perform the same sequence of acquire and release events in the same order, then the executions perform the same sequence of accesses to shared variables, and both executions produce the same final result.

b. If a program uses down() and up() primitives correctly, then mutex synchronization induces a total ordering on the down() and up() events.

[1] Sorry, I couldn't resist!