Exam Review

OS Structure

- Definition of what an OS does
- Technology improvements and impact on OS design.
- Hardware review
  - hardware features (traps, interrupts, ...)
- OS kernel architecture
  - monolithic, layered, microkernel, modular
- System calls and how they work

Processes and Threads

Topics you should understand:

1. What is a process?
2. What is a process control block? What is it used for? What information does it contain?
3. What execution states can a process be in? What do they mean? What causes a process to change execution states?
4. How does the OS keep track of processes?
5. What is a context switch? What happens during a context switch? What causes a context switch to occur?
6. What is the difference between a process and a thread?
7. What is the difference between a kernel thread and a user-level thread?
8. Advantages and disadvantages of user-level and kernel-level threads
9. Threading models: 1-1, many-to-one etc
10. How are processes created? Fork() and Exec()
    - Write pseudo-code for process creation using fork

CPU Scheduling

Topics you should understand:

1. What are FCFS, Round Robin, SJF, Multilevel Feedback Queue, and Lottery Scheduling algorithms?
2. What are the advantages and disadvantages of each?
3. What is preemptive scheduling? What is non-preemptive scheduling? Which scheduling algorithms can be preemptive?
4. What is a time slice? What effect does a very small time slice have? What effect does a very large time slice have?
5. What is an I/O bound process? What is a CPU bound process? Is there any reason to treat them differently for scheduling purposes?

CPU Scheduling

Things you should be able to do:

1. Given a list of processes, their arrival time, the lengths of their CPU and I/O bursts, and their total CPU time, you should be able to compute their completion time and waiting time for each scheduling algorithm we have discussed.
2. Given a variation to a scheduling algorithm we studied, discuss what impact you would expect that variation to have.

Synchronization

Topics you should understand:

1. Why do we need to synchronize processes/threads?
2. What is mutual exclusion?
3. What is a critical section?
4. What is a lock? What do you need to do to use a lock correctly?
5. What is a semaphore? What are the three things a semaphore can be used for?
6. What is a monitor? What is a condition variable? What are the two possible resumption semantics after a condition variable has been signaled? What are the advantages and disadvantages of each?
7. What is busy waiting?
8. How can interrupts be manipulated to support the implementation of critical sections? What are the advantages and disadvantages?
9. What is test&set? How can a test&set instruction be used to support the implementation of critical sections? What are the advantages and disadvantages?
10. How to implement locks using test& set or interrupt disabling?

Synchronization

Things you should be able to do:

1. Given some code that uses locks, semaphores, or monitors, you should be able to explain whether you believe it works. In particular, does it guarantee mutual exclusion where appropriate, does it avoid starvation, and does it avoid deadlock?

General Skills

- You should be able to read Java code.
- You will be asked to write pseudo code with synchronization.
- You will **not** be asked detailed questions about any specific operating system, such as Unix, Windows, Mac OS X ...