Review — Today

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 kernel mode? What is an interrupt?
7. What is the difference between a process and a thread?
8. What is the difference between a kernel thread and a user-level thread?

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?

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?

Things you should be able to do:

1. Given a synchronization problem, write semaphore and monitor solution.
2. Justify that your solution works (mutual exclusion, no deadlock, progress, etc.)

Deadlock

Topics you should understand:

1. What are the four necessary conditions for deadlock to occur?
2. What is the difference between deadlock detection and deadlock prevention?
3. After detecting deadlock, what options are conceivable for recovering from deadlock?
4. What is a safe state? What is the difference between an unsafe state and a deadlocked state?

Things you should be able to do:

1. Write deadlock free code.
2. Determine if a piece of code can deadlock.
3. Use the Banker's algorithm.

General Skills

- You should be able to read C++ and 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 NT, ...
- I highly recommend actually taking the practice exams, rather than simply looking over the questions and answers.