Review for CS-450 and CS-550 Final Exam

Fall 2005

Below are the principal areas of coverage that we have had through the entire semester, including what was covered on the Mid-Term and on the various quizzes. The Final Exam is cumulative, so you can expect to see many of the issues revisited that have already been raised both on the Mid-Term and on the quizzes. In addition, there are several topics listed in the review for the Mid-Term but were **not** covered in the Mid-Term. These topics are **very** likely to be included in the Final Exam. If you know the listed material well, you should get a high score on the exam, and if you did not perform hitherto at the level that you had expected, then here is your opportunity to redeem yourself.

NOTE: Page numbers for Nutt's text and detailed chapter/section identifiers are not provided. You should be able to look those up for yourself.

1. Which design goals for an Operating System are most important in which computer environments?

2. Sequence of historical progression of OS concepts and of memory layout in conjunction with hardware developments.

3. Major terminology and concepts of OS.

4. The four principal functions implemented by the kernel of the Operating System: Process, Thread, and Resource Management, Device Management, File Management, and Memory Management (Nutt fig. 3.10).

5. Multi-Programming/Multi-Tasking and Multi-Processing: understanding what each concept means and what are the differences between them.

6. Concurrency and Simultaneity: understanding what each concept means and what are the differences between them.

7. The differences between Hard- and Soft-Real-Time environments and the constraints that they impose upon system performance.

8. Review of hardware, principal events in boot sequence, and understanding both why and how is the modern OS event-driven (i.e., interrupt-driven).

9. The Fetch-Decode-Execute Cycle and its augmentation/modification to enable the recognition and handling of interrupts.

10. The processing of interrupts and of traps, and the difference between a trap and an interrupt.

11. Storage hierarchy and storage devices, including especially disk drives and magnetic tape.

12. Main Memory, Cache Memory, and the Translation Lookaside Buffer.

13. Privileged instructions, processor modes and the control of processor mode, and the role of mode change in the servicing of system calls and interrupts.

14. Groups of functions provided by the OS; and services provided by the OS. The Application Program Interface (API), and the Command-Line Interpreter.

15. System calls and their execution sequence.

16. Device-independent and Device-dependent portions of the Device Manager; in particular, mechanism for incorporating reconfigurable device drivers (Nutt's Figure 5.8-5.19).

17. Sequencing of execution of disk accesses: First-Come-First-Served, Shortest Seek-Time first, Scan, Look, Circular-Scan and Circular-Look. **In particular, be able to figure out, for a given sequence of track access request, in what sequence would the requests be executed under each I/O-scheduling algorithm.**

18. The Job or Process, possible process states. In particular, you should know which state transitions are permissible and which are not.

19. Contents of the Process Control Block (PCB) and of the Process Image for single-threaded processes.

20. The sequence of events in Context-Switching.

21. Creation and termination of processes; parent processes and child processes: *UNIX* fork(), wait(), and join()

22. Scheduling, especially three kinds of Process Scheduling (Long-Term, Medium-Term, and Short-Term, which kind(s) **must** an operating system have and which are present in only some, but not all, operating systems.

23. Threads, thread structures, relationship between multiple threads belonging to a single process, advantages of multiple threads for a single process, and especially the differences between User-Level threads and Kernel-Level threads.

24. CPU (short-term) scheduling, goals of CPU scheduling, circumstances under which the short-term scheduler is invoked, and operations of the scheduler.

25. Various algorithms used for CPU scheduling: First-Come First-Served (FCFS), Shortest Job First (SJF, both its preemptive variant (Shortest Remaining Time, SRT) and its non-preemptive variants), Round Robin Scheduling, Priority Scheduling (both preemptive and non-preemptive), Multi-Level Queue Scheduling, and the prediction of next CPU-burst time (simple arithmetic averaging of all previous bursts, assumption that next burst time will be identical to previous burst time, and at least a qualitative (but NOT a quantitative) understanding of exponential averaging). **In particular, be able to trace out the execution of various processes in accordance with the various CPU-scheduling algorithms.**

26. Two hardware synchronization approaches: Test-and-Set, and Swap or Exchange. Also, OS-provided semaphores and their use, in particular the counting semaphore.

27. Deadlock and starvation, **the four conditions necessary for deadlock to occur, and the four strategies for dealing with deadlock (Deadlock Prevention, Deadlock Avoidance, Deadlock-Detection-and-Resolution, and Disregard-for-the-Possibility-of-Deadlock).** The banker's algorithm.

28. Various memory management schemes, including: Single-Task-At-A-Time, Partitioned Memory with Equally-Sized Fixed Partitions, Partitioned Memory with Fixed Partitions of Multiple Sizes, Dynamically-Partitioned Memory; Segmented Memory, Paged Memory (whole process at a time), and Virtual Memory, including Demand-Paged Virtual Memory and its variant which is **both** segmented **and** paged. You should also be familiar with the structure and organization of per-process page tables (direct, two-stage, and three-stage), as well as of the system-wide inverted page table used in some systems.

29. In conjunction with the various memory-management schemes, you should understand how and at what stage from the time of program creation to the time of program execution are addresses resolved (compile time, link time, load time, or dynamically at run time).

30. In demand-paged virtual memory systems, you should be familiar with and in particular have a working knowledge of several of the major schemes used to determine which referenced pages are retained in physical memory and which are replaced. **The particular algorithms which you should know are Belady's Optimal algorithm, the Least Recently Used (LRU) algorithm, the First-in-First-Out (FIFO) algorithm, and Peter Denning's Working-Set algorithm.**

31. I also intend to address several issues in the assigned reading in Nutt's text, chapters 1 through 8, 10 through 13, and 20 that we did not get a chance to cover in class. The issues that I will raise will most probably though not necessarily come from the homework questions that I assigned. Please note that this will **not** be a major portion of the Final Exam. Thus, if you have been struggling all semester long to keep up and need to spend a lot of time reviewing what was listed in the prior paragraphs of this review, then it would probably not be worth your time at this stage to go back and re-read the entire assigned portion of the textbook. However, **if** you feel fairly comfortable with what we covered in class and don't require an extensive review of that material, then a comprehensive review of the assigned text readings can get you a few extra points on the exam.

NOTE: I strongly suggest that you not overdo the studying to the point that you become sleep-deprived. It is more important to be well-rested so that you can operate quickly and get the most benefit from what you know, than to try to squeeze in a few extra points of knowledge at the expense of tiring yourself out and performing poorly because of fatigue.