Computer Science and Engineering Department
University of South Florida
Ph.D. Qualifiers Exam Spring 2009
Operating Systems
January 23, 2009

You have 3 hours to solve 10 problems, all equal weight. If you believe that you cannot answer a question without making some assumptions, state those assumptions in your answer.

Irrelevant verbosity will not gain you points. Clear and crisp answers will be appreciated.

This is a closed books, closed notes exam.

1. Lottery scheduling is a simple, randomized proportional-share scheduling technique. In this scheme, each job has some number of tickets and at each quantum, a lottery is held, the winner of which runs the quantum. The more tickets a job has, the more likely it is to win (e.g., if A has two tickets, 0 and 1, and job B has one, 2, a random number between 0 and 2 inclusive will be chosen. If 0 or 1 are chosen, A will run, and if 2 is chosen, B will run).
   a. You are hired to redesign the lottery scheduling to remove the randomized aspect. Sketch out an algorithm that also provides "proportional sharing", but does so deterministically.
   b. How are the properties of the two algorithms (the original and yours) different in terms of fairness and performance? Is one better than the other? How do both compare to standard multi-level feedback scheduling, again in terms of fairness and performance?

2. A round-robin scheduler must handle both I/O-bound and CPU-bound processes in a fair and efficient fashion. One problem is that I/O-bound processes tend to relinquish the CPU before their quantum expires, but CPU-bound processes keep the CPU for the entire quantum. As a result, I/O-bound processes would tend to receive a smaller share of the CPU time. If the quantum is made small to mitigate this effect, CPU-bound jobs suffer because their throughput will be low due to increased context switching overhead.

   The UNIX BSD scheduler solves the problem by maintaining a FIFO ready queue for each priority level and grants the CPU to the process at the head of the highest priority queue. The priority level of a process in this scheme is not fixed, but manipulated by the scheduler as the process switches between scheduling states. Furthermore, the size of the quantum depends on the priority of the executing process.
   a. Draw a state diagram to show the different states that a process can be in, labeling the arcs with the events that cause a state transition.
   b. When should a process' priority be heightened? Why?
   c. When should a process' priority be lowered? Why?
   d. Should the quantum interval relate to priority level?
   e. Can processes with low or lower priority starve under heavy load?
   f. Describe the load under which this algorithm gives maximal CPU utilization.

3.

a. Define the term “deadlock”. There are four conditions that must hold before deadlock is possible. Name them.
b. Outline an algorithm that detects whether there is a deadlock. The algorithm should be able to cope with multiple types of resources, each type having a limited number of units available.
c. When should the algorithm be invoked? The answer to this question may depend on system characteristics such as the rate of resource requests, the granularity of resources, and the expected rate of deadlock. List three possible choices and discuss the criteria you would use to choose among them.

4. The following questions concern the handling of processes in a protected kernel-based operating system. Assume that the kernel creates and initializes each process to execute a statically linked program from an executable file whose name is passed as an argument to the process create primitive. Note: your answers should ignore the issues of page table structure, the virtual address translation mechanism, and argument handling.
   a. Explain how the kernel initializes physical memory for use by the program. How does the kernel determine the initial contents of the physical memory pages allocated to the new process?
   b. Once the CPU is executing in user mode in the context of the fresh process, what events could cause it to switch back into kernel mode? Give three distinct examples and outline how each affects the context upon re-entry to the kernel.
   c. Once the CPU is in kernel mode as a result of the examples in part (b), what events could cause it to switch back into user mode? List as many distinct examples as you can think of.

5. Copy-on-write is a way for processes to share pages (or segments) that are logically distinct. When one process sends some data to a second process with copy semantic, nothing needs to be copied immediately or perhaps ever.
   a. Explain how standard memory-management hardware can be used to implement copy-on-write cheaply;
   b. Linus Torvald dislikes copy-on-write and claims: "Once you play games with page tables, you are generally better off copying the data". Describe the extra costs of copy-on-write, and explain when it is better to just copy the data directly.
   c. Describe how copy-on-write can be used to improve system performance for:
      i. Message passing;
      ii. Forking a new process;
      iii. File I/O operations.

6. Lamport describes an algorithm to logically order events in a distributed system. In his algorithm, events a and b, in processes i and j, have logical time stamps Ci(a) and Cj(b). If Ci(a) < Cj(b), we do not know if a→b or if a and b are un-ordered. How would you extend the logical time information sent with each message to include enough information that you could disambiguate the case described above?

7. You have been given a user account on a remote machine to which you are porting some "high performance application". All you know about this machine is that it supports the POSIX interface; you have no useful documentation, no one is answering your e-mail about the machine, and any interfaces you know of for acquiring information about the machine appear to be broken.

   To obtain reasonable performance for your application, you believe that it would be useful to know the following properties about the machine or OS:
      a. Number of CPUs
      b. Page size (in bytes)
      c. Amount of physical memory (in bytes)

   For each property, give a brief but precise description of a benchmark program you develop that allows you to infer each property. Also, clearly state any assumptions that you are making about the machine or OS and any limitations of your benchmark.

8. Compare file protection based on user groups (as in Unix) with access-control lists, in which individual users are designated as having specific access privileges to a file. Consider complexity of implementation, ease of use, flexibility, and security in your answer.

9.

a. What is "swap space"?
b. Swap space can be allocated as a special region (partition) on the disk or it could be a special file. Describe the advantages and disadvantages of these two approaches. Be sure to discuss issues related to performance, functionality, and cleanliness of design.

10. One new trend in operating systems is motivated by the wide-spread adoption of mobile devices, such as smart phones, PDAs, etc. Discuss the various aspects of an operating system that need to be re-designed to run on these devices. In your discussion, include topics such as performance metrics for various components of an OS, objectives that may differ from traditional systems, which components can be used unmodified from traditional OS and which components need to be changed. Justify your answers.