Computer Science and Engineering Department
University of South Florida
Ph.D. Qualifiers Exam Fall 2013
Operating Systems

You have 3 hours to solve 7 problems. 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.

Problem 1 (10 points): Distributed systems

Can you guarantee that distributed communications are not compromised? If so, how? If not, why not?

Problem 2 (15 points): OS features

a) What operating system challenges do devices like the Nook, Kindle and Ipad introduce, compared to traditional computer-based operation systems? What simplifications do they introduce?
b) A multi-user operating system can be described as interrupt driven. Would this be true of a phone operating system (and if so, how)? How about the operating system for a camera's image auto-stabilization function, which is done by a dedicated chip?

Problem 3 (20 points): Distributed file systems

Distributed file systems use caching to improve performance and scalability. NFSv3 uses block caching with validation, AFS uses whole-file caching with callbacks, and other systems use block caching with file leases (e.g., NQ-NFS and NFSv4). Suppose that there is a fixed universe of files, each client generates requests at a fixed rate, client failures occur with some fixed probability per time unit, and all files have a fixed probability of reference for each request. How will each of these schemes behave as the number of clients increases and/or as their distance from the server increases? Explain your answer.

Problem 4 (15 points): Interprocess communication

Consider two processes that need to communicate by sending messages.

a) If the processes are running on a *uniprocessor* and use operating system kernel calls (send() and *recv()* routines) to communicate, what would you expect the main costs (i.e., the most time-consuming activities) to be in communicating from process to process? How does the size of a message affect your answer?

b) Suppose that the two processes can share a part of their address spaces, and use this shared space to store their message queues. How does this affect your answer to part (a)?

c) Now suppose that the processes are running on separate processes in a symmetric shared-memory *multi- processor*. Answer part (a) for this new scenario.

Problem 5 (15 points): Thread Scheduling

a) List two main differences between user-level threads and processes.

b) Typical implementation of a thread package works as follows. The package provides user-level thread abstractions to the application, allowing cheap creation and destruction of threads. The package asks the kernel for a number of light-weight processes (also known as kernel-level threads), typically each light-weight process corresponds to one processor in the system. The threads package performs thread scheduling by binding user-level threads to light-weight processes. Explain why such a scheme is unsatisfactory from a user application's point of view.

c) Suggest some enhancements to kernel mechanisms (including new system calls or upcalls) that correct this problem.

Problem 6 (15 points): Scheduling

Computer scientists have studied process and thread scheduling for decades. One reason is that we cannot agree on what to optimize.

1. Give 3 examples of goals for which the schedule can be optimized. For each goal, describe:
    a. a workload where the goal is important
    b. a scheduling algorithm that targets that goal
2. Pick two of the three algorithms from you part a) and explain how best you can integrate them into the same system. Does this achieve both goals under any circumstances? If not, when?

Problem 7 (10 points): Secondary storage

You have to choose between a “high speed” 512 GB disk and 512GB of flash storage as secondary storage. What are the tradeoffs you would consider? Which would you prefer and why?