Computer Science and Engineering Department
University of South Florida
Ph.D. Qualifiers Exam Fall 2010
Operating Systems

You have 180 minutes to solve 8 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. **Mode and Context** 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. Explain how the kernel initializes the context (program counter PC and stack pointer SP) before switching the CPU to user mode to start the fresh process for the first time. How are the initial contents of these registers determined?
    c. 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.
    d. Once the CPU is in kernel mode as a result of the examples in part (c), what events could cause it to switch back into user mode? List as many distinct examples as you can think of, and outline briefly how each affects the CPU context.

2. **Scheduling**: What is priority inversion, and why is it bad? Illustrate with an example. Use your example to illustrate one technique that avoids priority inversion.

3. **Virtual memory and hardware support:** Operating systems that support virtual memory rely on memory management facilities provided by the hardware. One facility common on RISC processors is a *software-controlled* translation buffer (TLB). The TLB is a small hardware cache of virtual-to-physical address translations. The hardware directly handles references to addresses whose translations are present in the TLB; any other memory reference requires intervention by system software to load the needed translation into the TLB. Processors with software-controlled TLBs provide special instructions to control the TLB. These instructions might include:

   tlb_load_entry: load an entry (i.e., a virtual page address and its corresponding physical page address) into the TLB.
   write_enable and **write_disable**: enable/disable write privilege for a virtual page. Any attempt to write to a page with a write-disabled translation will generate a trap to system software.
   tlb_invalidate_entry: remove a translation from the TLB.

   Explain how and when the TLB control instructions might be used by the operating system software to implement a virtual memory operating system such as Unix.

   a. **tlb_load_entry**

   b. **write_enable** and **write_disable**

   c. **tlb_invalidate_entry**

4. **Performance:** Sketch rough graphs ("back of napkin") illustrating the relationships among the following quantities. Note any important assumptions underlying your analysis.
   a. Page fault rate as a function of page size. What page sizes are typical for current processors and operating systems?
   b. Disk access latency as a function of rotation speed. What rotation speeds and access latencies are typical for current disk drives?
   c. Peak disk read bandwidth and read latency as a function of read block size (request size). What peak bandwidths are typical for current disk drives?
   d. Peak read bandwidth, I/O operation throughput (IOPS), and access latency as a function of the number of drives in a RAID storage unit.
   e. I/O block cache hit ratio as a function of cache size, for random references uniformly distributed across the I/O block space.
   f. I/O block cache hit ratio as a function of cache size, for sequential references.

5. **Distributed file systems**:
    a. Most distributed file systems cache recently used file data in client memory. What are the performance benefits of file caching? What are the performance costs?
    b. File caching introduces the problem of cache consistency when files are shared across the network. Explain the problem and demonstrate it with an example.
    c. Outline a plausible scheme for dealing with the problem of cache consistency in a distributed file system. You may ignore the problem of failures in your answer (but see next question). Your scheme need not be identical to that used in any specific distributed file system.
    d. What are the limitations of the file caching scheme you described for previous question? Explain how your scheme can handle failures (crashes), assuming that systems fail only by stopping, discarding the contents of their memory, and restarting.

6. **Virtual machines:**
    a. Explain the fundamental idea behind a virtual machine.
    b. List some benefits of and costs introduced by virtual machines. (You may want to consider the technical motivations for the recent revival of virtual machines.)
    c. Would it make sense to run virtual machines on mobile devices? Discuss at least two reasons for or against.

7. **Memory:** Can the size of the working set of a process affect how much CPU time it gets and how often it gets the CPU? Support your answers with specific examples.

8. **Trends:** Apple originally did not include support for multi-tasking in its iphone OS. Why do you think they made this design decision? The newer iphone OS, however, includes support for multi-tasking. What were the technical arguments that might have prompted Apple engineers to make this change? What (if any) problems might this give to the user of the device?