CSC 256/CSC 456
Homework 3
December 4, 2018
Due in class on Tuseday, December 11, 2018

Please write in your own words and be sure to show all your work and write down any assumptions you make.

1. Protection and security:

   (a) [5 points] What are the main differences between access lists and capability lists? Where does the Unix protection mechanism fit in the above schemes?

   (b) [5 points] Discuss ways by which managers of systems connected to the Internet could have limited or eliminated the damage done by the Morris Worm. What are the drawbacks if any of making such changes to the ways in which the system operates?

2. File systems

   (a) [Extra, 5 points] The Sprite Log-Structured File system manages the entire file system as a log by writing all modifications to disk sequentially under the assumption that reads are infrequent (because they are performed on cached data). For what kind of access pattern would this provide better performance than a traditional Unix File system? For what kind of access pattern would the Unix File system perform better?

   (b) [5 points] Solid state disks (persistent storage with no mechanical moving parts) can be constructed using technologies such as FLASH or phase change memory (PCM). These technologies offer random access reads and writes (i.e., access times are independent of the location of the previous block read and written). Access times for these reads and writes are also orders of magnitude lower than for traditional magnetic disks. However, the devices do wear with time - i.e., the number of writes to any given location is finite (although large). Discuss at least two implications of these technology parameters on the design of the file system - i.e., how might you implement a file system differently for these devices? For example, are there any aspects of traditional file system design that can be relaxed? Are there any additional complexities and how might you resolve them?

3. Distributed file systems:

   (a) [5 points] NFS (Network File System) is said to provide a stateless file service while AFS (the Andrew File System) provides a stateful file service. What do these terms mean/what are some of the tradeoffs?

   (b) [5 points] We have discussed the Google File System and the Andrew File System in class. What are the environments for which each is tailored? Describe at least two file system design decisions made by each in order to support its target environment.

4. Real-time scheduling: Recall that a hard real-time system is one that must ensure that any process that is admitted/accepted can receive its requested/anticipated processing time and meet its deadlines.

   (a) [5 points] Such a hard real-time system is asked to schedule three periodic processes: process A requests 50 seconds of CPU time every 100 seconds, process B requests 20 seconds of CPU time every 50 seconds, and process C requests 2.5 seconds of CPU time every 25 seconds. The deadline for each process requires that it complete its CPU burst by the start of its next period. Is this system schedulable (can the real-time system satisfy the schedule)? Explain your answer.

   (b) [5 points] Explain why rate monotonic scheduling (scheduling with a static priority based on shortest period) will or will not work in the above scenario.

5. Multiprocessor OSes: On multicore and multiprocessor systems, the typical Linux scheduler uses per core (per logical processor) scheduling queues with occasional load balancing across queues. Within each queue, a processor typically uses local and independent criteria to determine which process to run next.

   (a) [2.5 points] Why is this two-level strategy used?

   (b) [2.5 points] Given the use of a per-processor ready queue as above, does a processor still need to synchronize access to the ready queue? Why or why not?

   (c) [Extra, 2.5 points] Consider a multicore processor with two cores on which a total of two processes are currently scheduled, one of which is higher priority than the other. Discuss ways in which the lower priority process could somehow interfere with the higher priority process, preventing it from utilizing the resources it requires.

   (d) [Extra, 2.5 points] As a systems designer, you are tasked with ensuring that the above scenarios are disallowed. You may use either hardware or software techniques. Discuss at least one design to control the interference above.

6. Virtual machines: A virtual machine monitor provides execution environments with an interface identical (or nearly identical) to the underlying bare hardware. These execution environments are called virtual machines (VMs). An operating system and its user programs run in each virtual machine. Conventionally, a process page table maps program *logical* addresses to *physical* addresses. However, the inside-VM physical address may not be equal to the real *machine* address in a VM-based system. For proper execution on a processor with hardware-loaded TLBs (such as x86), a page table that maps program logical addresses to real machine addresses must be presented to the hardware. In practice, the virtual machine monitor maintains one such *shadow* page table (logical to real machine address mapping) for each conventional inside-VM process page table (logical to VM physical address mapping).

   (a) [5 points] To maintain the shadow page table, does the virtual machine monitor need to trap updates to conventional process page tables (made by the inside-VM (guest) operating system)? Does the virtual machine monitor need to trap reads to the conventional process page tables made by the guest operating system?

   (b) [5 points] Through operating system instrumentation (e.g., Xen) or binary rewriting (e.g., VMware), the inside-VM (guest) operating system can make explicit calls to notify the virtual machine monitor of page table updates (as opposed to using traps). What are the benefits of this approach?