CSC 256/CSC 456
Final Exam
December 11th, 2012
Total: 60 points

This exam is closed book, closed notes. Please be concise in, but always explain, your answers. An answer without a corresponding explanation may *not* receive credit even if it is correct. While I expect more than just the name of a particular technique if that is the answer to a question, extraneous remarks may count against you. On the flip side, exceptionally good answers may receive extra credit.

There are a total of 60 points (budgeting 1 point per minute will allow you to pace yourself, with a little extra time to spend on questions where you need a little more thought - some questions will take more or less time, but you should use this estimate to keep on track - a good test taking strategy is to do a quick scan and start with the questions you feel most confident about).

While I have tried to make the questions as clear as possible, in the event that you need to make an assumption to proceed, please explicitly write down your assumption/s. In the interest of fairness, the proctor has been instructed not to attempt to answer any questions during the exam.

1. [10 points] Real-time systems: 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 two periodic processes: process A requests 25 seconds of CPU time every 50 seconds, while process B requests 35 seconds of CPU time every 75 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 any 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.

2. [20 points] File and I/O systems:

   (a) [5 points] As system designer, you have a choice of handling an application's page faults or file accesses that require data to be retrieved from disk by either polling the disk status register or using interrupts. Which would you pick and why?

   (b) [5 points] Through your homework assignment, you discovered that the average seek time (using a first-come-first-serve policy) for a disk with N cylinders when back-to-back requests for sectors with independent and equal probability of being resident in any cylinder are made, is less than $N/2$. Explain the intuition behind this result.

   (c) [10 points] For your file system, you have chosen a block size of 4096 bytes, and have made a decision to use 2-level indexed allocation, with 4-byte block pointers.

      i. [3 points] What is the maximum file size allowed?

ii. [3 points] What is the maximum file system size?

iii. [4 points] Suppose you have a disk subsystem that can hold more data than the above maximum file system size. List at least two ways in which you might utilize all the available space.

3. [15 points] Memory management:

   (a) [5 points] The Intel x86 processors support 4MB pages in addition to standard 4 KB pages. Other architectures have similar features. Large pages increase TLB (translation lookaside buffer) reach and help reduce page table size. However, incorrect use of a large page can lead to poor memory utilization and potential page thrashing, because data accesses are scattered widely across the address space. How effective is the use of large pages for the following five program structures/access patterns, assuming that each of these structures is fairly large: *stack*, *hashed symbol table*, *sequential search*, *binary search*, and *executable code*? Explain your answer.

   (b) [5 points] For the following page reference string, what are the number of page faults using 1) first-in-first-out (FIFO) and 2) the second chance (clock) replacement policies, assuming your memory can hold 3 page frames and that they are initially empty? Show your work.

   ```
   7 0 1 2 0 3 0 4 2 3
   ```

   (c) [5 points] Single address space operating systems have been proposed that take advantage of 64-bit addressing to eliminate the need for separate per-process virtual address spaces. Files and data for all processes are mapped into a single address space. Protection is achieved via protection identifiers associated with the translation lookaside buffer entries. Name at least two advantages of using a single address space for all processes.

4. [15 points] Multiprocessor operating systems and virtual machines:

   (a) [5 points] Scalable multicore operating systems for multicore chips with large numbers of cores that we have discussed in class, such as factored OS and Corey, dedicate cores to specific operating system functionality. What is the advantage of this approach? What is a disadvantage (hint: why don't operating systems for smaller-scale multicores or multiprocessors use this approach)?

   (b) [5 points] You are designing a multiprocessor operating system for a system targeted to run many parallel applications (ones that make use of multiple processors/cores at the same time in order to speed up their computation). Recall that the default Linux scheduler uses one ready queue per processor, with each processor making independent scheduling decisions by picking processes to execute from its own ready queue. How might you change this default design to improve the performance of the system?

   (c) [5 points] Virtual machines have become popular in the last few years in order to facilitate server consolidation, provide strong performance and fault isolation, and to allow legacy application execution. Typically, guest operating systems (guest OSes) run on top of the virtual machine monitor and are not allowed to execute in superuser mode. The Intel x86 machines (prior to VT-x support) would allow some sensitive instructions executed by the guest OS (those that require execution in superuser mode to access protected resources) to fail silently. Explain why this might be an issue as well as one possible approach to overcoming this challenge.