CS-736: Mid-term Exam (Spring 2000)

	Points	Total Possible
Part I: Short Answers		50
Part II: Long Answers		50

Name: ____

Part I: Short Questions

1. Describe the "priority inversion" problem (a simple example is good enough). What mechanism does the lottery scheduling system use solve this problem?

2. Which of the following is the least appropriate in describing a scheduler activation?
   (a) An in-kernel execution context
   (b) A more functional user-level thread library
   (c) A better abstraction of a multiprocessor
   (d) A communication medium between user-level and the kernel

3. Certain pieces of code inside the Nucleus kernel would be called when a user process was communicating with an "external process". This code is most similar to what entity in modern operating systems?
   (a) Device driver code
   (b) Process scheduler code
   (c) File system i-node lookup code
   (d) Virtual memory mapping code

4. With resource containers, "scheduler bindings" are used by the kernel. Briefly, what are scheduler bindings, and why are they needed?

5. File access (reading data from disk) in Multics is most similar to which system (in terms of what the user's code would look like, in particular):
   (a) Unix
   (b) Hydra
   (c) Pilot
   (d) Nucleus

6. Semaphores, as invented by Dijkstra, were used in two entirely different ways in the construction of the THE system. Describe both ways.

7. Which of the following is *not* an assumption made by the AutoRAID storage array designers?
   (a) Only a fraction of all data in the array is active at any given time
   (b) RAID-1 performance is lower than RAID-5 performance
   (c) Redundancy-level storage hierarchy should not be implemented in the file system
   (d) The active set of data in the array changes slowly over time
8. Based on the comparison between Mach and Ultrix that we read about, which of the following is true?
   (a) The operating system's virtual memory page mapping policy does not affect performance
   (b) Dedicating a separate cache for both user programs and the kernel does not improve performance very much
   (c) Programs running on top of Mach always run more slowly than programs running on top of Ultrix
   (d) Ultrix has a slightly higher MCPI than Mach in most cases
9. Describe the basic idea behind the Scavenger in the Pilot system. Is it used for performance, correctness, or both? What entity in LFS is this most similar to?
10. The basic structure of the Mach operating system is most similar to:
    (a) THE
    (b) Hydra
    (c) Nucleus
    (d) Unix
11. Assume our workload for a RAID-5 array consists of only "small" (single block size) writes. What is the maximum percent of peak bandwidth that could be delivered by the RAID under such a workload?
    (a) 25%
    (b) 33%
    (c) 50%
    (d) 75%
12. In the Unix file system, file names are stored in:
    (a) Directory i-node
    (b) Directory data block
    (c) File i-node
    (d) File data block

13. Assume that you are allowed to create a single file, using sequential or random writes, and then read that file back, using sequential or random reads. Which combination of writes and then reads would lead to LFS performance that is much *worse* than FFS performance? Why?

14. In Vax/VMS, the per-process resident set lists were managed with a simple FIFO policy. When removed from that list, pages were placed onto either a free list or a modified page list. How were those lists managed?

15. In Disco, the "illusion" provided by the virtual machine monitor to the operating system had to be "broken" sometimes, to improve performance of the system. This usually required changing some code within the operating system. Describe one such situation where this occurs, and how it improves performance.

16. In the incremental specialization paper, "guards" are *not* used directly or indirectly to:

    (a) Ensure the absence of user-level locks
    (b) Optimize small, private reads to files
    (c) Detect shared access to non-specialized files
    (d) Ensure that the proper block size was used

17. Which of the following were *not* added to the file system by the designers of FFS?

    (a) Hard links to other files
    (b) Bigger block size
    (c) Redundant staggered-offset superblocks
    (d) Bit maps

18. An object in Hydra consists of a unique name, type, and representation of the object. The representation, in turn, consists of capabilities and data. In general, which are user programs are able to directly manipulate? (capabilities, data, both, neither?) In general, which are the kernel able to directly manipulate? (capabilities, data, both, neither). Explain.

Part II: Longer Questions

1. **Nucleus meets the Lottery:** In this problem, you will design a simple lottery scheduler using the primitives provided by the Nucleus system. In other words, your personalized OS will be running some number of processes as its children, and your job is to schedule them via a lottery. The basic Nucleus primitives are send_message, wait_message, send_answer, wait_answer for communication, and start_process and stop_process for job control.

   Two important assumptions:

   (a) Assume that the scheduler can communicate with an external alarm clock process.

   (b) Assume that you also have already implemented (luckily) the do_lottery_scheduling() function, which returns a job to run based on a lottery that it holds.

   a): First, show the pseudo-code of main body of the scheduler. Don't worry about initialization or details like that; just demonstrate how system basically operates. Also, assume all processes fit into memory.

   b): Now, use the primitives load_process and save_process to add swapping to your Nucleus scheduler.

   c): Assume that loading a process takes 50ms, saving it about the same, and stopping and starting it each about 5ms. How big should your lottery scheduling time quantum be?

2. **Pilot reads and writes:** Pilot provides users with memory-mapped access to storage. Unfortunately, you are contracted to provide a open()/read()/write()/lseek()/close() interface for users. Don't worry, you are getting paid quite well for your services. The Pilot Space primitives you can use Space.Map(file reference, file offset, length of mapping), which returns a virtual address where that part of the file was mapped, Space.Activate(offset into space), Space.Deactivate(offset into space), and Space.Kill(). The File primitives should just be restricted to File.Open(filename, mode), which returns a reference to the file and which can be used by the Space primitive to map parts of the file, and File.GetLength(), which returns the length of the file.

   Remember, Activate() and Deactivate() are just hints!

   a): Show how open(filename) would be implemented for read access. Assume that the entire file *cannot* fit into physical memory. Make sure to return a file descriptor to the caller! Assume that all files are a multiple of 1MB in size.

   b): Now implement read() on top of a mapped file. The read(fd, buffer, size) call takes three arguments: a file descriptor, a buffer to read the data into, and a size, and in general should behave like a Unix read() call. For simplicity, assume that the read is small (page-sized) and page-aligned.

   c): Now, describe how simple prefetching could be added to your read() code. Assume access to the file is sequential for simplicity (so the code doesn't have to be overly general).

3. **Multics on a NUMA:** Multics lives! And it is your job to port Multics to a NUMA machine. Luckily, you are just the conceptual designer, and can leave all of the low-level hacking to your underlings. Your basic goal will be to design a page/segment migration functionality into the basic Multics design.

   a): First, describe how a process in Multics logically accesses an external segment. For this part of the problem, assume that this is a purely segmented system (that is, no paging underneath of all of that segmentation).

   Use a picture instead of lots of words.

   b): Still assuming that there is no paging, show how a segment could be migrated to a different location, perhaps to improve memory locality. What are the steps in the process? What data structures would you need to add to allow for this book-keeping?

   c): Finally, assume that you have paging underneath of all of that segmentation, and describe how page (not segment) migration would occur. Given your answer, which would you implement: full segment migration, or page migration? (Why?)