Computer Science 161

Midterm

150 minutes/150 points

Fill in your name, logname, and TF's name below.

Name ______________________________

Logname ______________________________

The points allocated to each problem correspond to how much time we think the problem should take. We ask that you not spend more than 150 minutes on this exam. You need not complete the exam in one sitting, so you can think about problems while you are not actively working, but we ask that you limit actual work to not more than 150 minutes total.

Please place the answers to the questions in a PLAIN TEXT file. Mark the answers clearly so it is obvious what answers correspond to what questions. Please place your answers in order in the file. Send your completed exam to margo@eecs.harvard.edu. Please enclose your answers directly in your mail message - do not make it a separate attachment.

This exam is open text book, open source code, and open notes. **You may not use computers[1] during the exam** (other than to record your answers or view OS/161 code). You must complete the exam on your own.

I have neither given nor received help on this exam.

____

Please sign your name here indicating that you have neither given nor received help on this exam.

Please note in this space the time period(s) during which you worked on this exam.

1. Computers include cell phones, pdas, smart phones or any device with which you can talk to a network.

True/False (20 points)

For each statement, answer T if the answer is always true, else answer F. (2 points each)

1. All operating system code run in the same address space.	T	F	F
2. Once an operating system is running, you cannot extend its functionality.	T	F	F
3. If memory were infinite, there would be no need for virtual memory.	T	F	F
4. The scheduler runs on every context switch.	T	F	F
5. A context switch can occur in the middle of an instruction.	T	F	T
6. Shortest-time-to-completion-first is the fairest scheduling algorithm.	T	F	F
7. Multiprogramming is unnecessary on single-user systems.	T	F	F
8. Locks and binary semaphores are equivalent.	T	F	F
9. Virtual memory requires hardware support.	T	F	T
10. Unless you put bugs in your code, you cannot have deadlock on a preemptible resource.	T	F	T

Multiple Choice (9 points) (3 points each)

11. Assume that you have a semaphore associated with each item (and on the head structure) on a doubly linked list. Using no other synchronization primitives, what is the fewest number of semphores that you must acquire for any operation (lookup, insert, delete)? A - always lock the head of the list

A. 1

B. 3

C. N, where N is the number of items on the list.

D. None of the above

12. Which of the following operations can be done on a typical processor without trapping into the kernel (that is, which of these, could be implemented in user space without wreaking havoc)? You may list as many answers as you think are correct. B, C, F

A. process fork

B. thread fork

C. getpid()

D. write data to a disk drive

E. increase the amount of memory allocated to a process

F. increase the amount of memory allocated to a thread's stack

13. Which of the following scheduing algorithms will tend to schedule I/O bound jobs before CPU bound jobs? B, C

A. Shortest-time-to-completion-first

B. Multilevel Feedback Queues

C. Lottery Scheduling

D. Round Robin

E. First Come First Serve

Short Answer (43 points)

14. List two common hardware primitives with which you can implement synchronization primitives on a multiprocessor. Explain how each can be used. (4 points)

TAS, CAS, LL/SC

15. What other hardware mechanism can be used on a uniprocessor to achieve mutual exclusion? (2 points)

Turning interrupts off.

16. List two reasons why an operating system designer might choose not to have user threads map 1:1 with kernel threads. (4 points)

user-level thread scheduling is faster because you don't have to context switch into the kernel.

the application can implement whatever scheduling policy it wants -- not left to the discretion of the OS.

17. For each of the following traps, indicate whether the trap is synchronous or asynchronous with respect to a user-level program. (3 points)

System call (S)

Exception (S)

Interrupt (A)

18. Why might it be advantageous to keep a process running on the same processor on which it last ran (this is called processor affinity)? (2 points)

You already have the process's memory in the processor's cache.

19. Imagine that you've been asked to build an operating system for a simple embedded processor. The processor has no virtual memory and no memory translation unit. What functionality must you build into your process loader if you wish to have multiple programs resident in memory? (4 points)

You have to do some kind of address relocation to place the process onto a specific chunk of physical memory.

20. In a system with virtual memory, how can you share memory between two processes? (4 points)

Processes can share segments or page table entries.

21. You have been asked to design an adaptive VM system. On a per-process basis, you must decide whether to swap the entire process in and out of memory or whether to simply demand-page it in. (6 points; 2 and 4)

What statistics do you want to gather to help you make this decision?

process image size, working set size, rate of change of working set

Describe how you'll use the information that you collect.

The key question I'm trying to figure out is how expensive the entire swap in/out is and how many page-faults I'd have to handle. So, the working set size will give me an indication of how long it takes to swap in the process. The rate of change in that working set tells me what kind of page fault rate to expect (and also the granularity at which I want to do working set swaps). The process size gives me some upper bounds on how bad things can get. So, I'll compute the time for page faults at a given rate and compare that to the process working set swap time.

22. Which scheduling algorithms discussed in class, listed in the notes, and described in the book guarantee no starvation? (4 points)

round robin

fair share

23. In a real time system, processes indicate hard constraints about when they need to run. If you were designing a scheduler for such a system, speculate about how would you use that information and what scheduling algorithm you would implement? (6 points).

The deadlines that the real time apps give you essentially gives you a way to actually do Shortest-time-to-completion-first (although they call it earliest deadline first). So, you use the information from the apps to define your "completion" times and then implement Shortest-time-to-completion-first.

24. Although First-come-first-serve is less fair than multi-level feedback queues, what reason might you have for deciding to use it anyway? (4 points)

It's more efficient in that you (probably) use fewer context switches.

Fun with MMUs (23 points)

BNE Corporation has just announced a new CPU/MMU combination.

It has a 9-bit virtual address space with 3 bits of segment number and 6 bits of offset. The 3 bit segment number indexes into a segment table. The machine supports a maximum of 4 KB of physical memory. They claim that the system can support demand-paging of 16-byte pages.

25. How large are segments on this architecture? (2 points)

64 bytes

26. How many pages are there per segment? (1 point)

4

It turns out that BNE printed their architecture manual single-sided, omitting every other page. As a result, you know some things about the archtiecture, but you're going to have to reverse engineer the rest. Here is what you know.

1. The MMU contains a segment table containing 8 entries.
2. Each entry contains a base (8 bits) and a bounds (6 bits) and 1 protection bit indicating if the segment is executable.
3. There is mention of page tables in the MMU, but most of the discussion about them is missing.

For the next set of questions, make an informed guess about how the machine works. Justify your guess.

27. During execution, when an address goes to the MMU, how does it determine which segment table entry to consult? (2 points)

Look at the top 3 bits.

28. How many PTEs are in a page table? (2 points)

Well, there are two bits of page number, so I'd say 4.

29. List two different conditions that might cause the MMU to generate an exception. Give concrete examples (explain the contents of an entry in the segment table, what it would contain and what address and/or instruction would cause the exception). Try to make your exceptions as different as possible. (6 points)

1. The segment table does not show proper access (read/write/execute) for the way that the address is being used.

execute instruction at X; X maps to a segment without execute permission

2. The address exceeds the bounds of the segment. Let's say that the bounds is 0xf and we generate an address whose offset is 0x1f., that should generate a segmentation violation.

Recall that the architecture supports demand paging. With that in mind, answer the following questions.

30. What information must appear in a page table entry? (4 points)

A physical page number or a disk address.

(Don't need to have permission bits because those are on the segments.)

31. Is it possible for an address to be generate neither of the exceptions you listed in question 29, but still generate a page fault? If so, how? If not, why not? (6 points)

Well, if we turn paging on, then the base and bounds must be pointing to page tables (we can still manage those in software if we want). So, we would properly go through the segment table, look up an entry in a page table and find that the entry isn't in memory. In that case, we'd generate a page fault.

Synchronization (15 points) (3 points each)

For each of the problems described below, indicate which of the following synchronization primitives is BEST suited for it. Explain briefly why you chose the primitive you did.

Counting semphore
Lock w/condition variable
Locks
R/W Locks
Monitors

32. You have a bushel of apples. There are a pile of worms nearby. The bad news is that both you and the worms want to eat the apples. The good news is that the worms are willing to use whatever synchronization primitive you want. The worms don't mind if there are other worms or people eating the apple while they are munching away, but you are a bit squeamish and refuse to eat an apple while any worms are eating it (you are however, not so squeamish as to pass up an apple that currently has no worms, even if if had worms in the past). RW LOCK

33. You notice that the person you'd like to ask out on a date is wearing a mood ring. You decide that it's best to ask the person out when the mood ring is either blue or purple. Once again you find yourself in the fortunate situation that the mood ring contains an embedded controller capable of issuing appropriate synchronization operations before it changes color (and of course, you know how to use synchronization primitives as well). What synchronization primitive do you want to share with the mood ring to help you get a date? CVs

34. Professor Seltzer has four cats and two hamsters. Her friend Mary has eleven cats. Margo invites Mary's eleven cats over for a playdate. Each cat is allowed to play with either hamster, but only one cat can play with a particular hamster at any one time. The cats are clever beasts and know how to use synchronization primitives. What primitive to you want to use to grant the cats access to the hamsters? (Don't worry, the hamsters are studly and will take out any cat that tries to eat them.) Counting sempahore

35. You've been giving a set of self-synchronizing chairs. The set supports the operations: `allocate_chair, free_chair, spin_all_chairs, throw_chair_down_stairs`. The set was manufactured by a company all of whose employees aced CS161. What primitive do you suppose they used to implement the chair interfaces? monitor

36. Every CS161 student was so excited about the midterm that they wanted to schedule a meeting with Professor Seltzer. Naturally, she required proper synchronization to do so. What primitive did she pick? locks