

CS 167 Midterm Exam Solutions

Fall 2009

Do all of questions 1 through 4.

1. *You are to write a device driver for a data-collection device. Each time a new piece of data is gathered, the device interrupts the processor. The device driver must respond to the interrupt, and, in the interrupt handler, fetch the data from a device register and put the data into a queue from which the data will be consumed by user threads (that have entered the kernel to perform read system calls). Each user thread consumes one data item at a time. There may be multiple data-collection devices attached to the system; their interrupt handlers put their data on the same shared queue. All interrupt the processor at the same priority level (thus one interrupt is processed at a time). Assume that there are no special instructions for putting items into a queue atomically or for removing items from a queue atomically. Keep in mind that when an interrupt at a particular priority takes place, the processor automatically masks interrupts at that priority and lower.*

a. *Assume we have a uniprocessor system and the operating-system kernel is non-preemptible. Describe how you would synchronize access to the queue.*

Threads should mask interrupts while accessing the queue.

b. *Suppose now the kernel is preemptible. How would you change your answer to part a?*

There is now the additional concern that multiple threads might access the queue concurrently. To prevent this, a mutex should be associated with the queue and threads must lock the mutex before accessing the queue and unlock it afterwards.

c. *Suppose further that we have a multiprocessor system and that each device interrupt is delivered to a processor chosen at random (thus two devices can interrupt different processors simultaneously). How would you change your answer to part b? Note that a thread can mask interrupts only on the processor on which it is running.*

It's no longer the case that masking interrupts is sufficient to keep an interrupt handler from accessing the queue. Since interrupt handlers cannot sleep waiting for a mutex, we use a spin lock. Thus all entities, both threads and interrupt handlers, must lock the spin lock before accessing the queue and unlock it afterwards. It's now a problem if a thread, holding the spin lock, loses its time slice. So, to prevent this from happening, threads disable preemption (perhaps by masking clock interrupts) while accessing the queue.

d. *Finally, suppose that each piece of data requires significant processing that must be completed before the data is made available to user threads. What additional concerns do we have? How should they be coped with?*

Our concern is that we don't want the processing of data to interfere with other important interrupt-handling duties of the processor. One way of avoiding the problem is to defer the processing by use of a deferred procedure call (DPC) so that the processing is done at the lowest interrupt priority level, but before the system returns to the context of the interrupted thread.

2. *In systems using x86 architectures, devices are controlled via registers that are mapped into the address space. So, in particular, a disk device might have a memory-address register into which the operating system places the address of a buffer, a device-address register into which the operating system places the disk address of an operation, and a control register into which the*

operating system places the device command to be performed. All three registers are accessed by loading from and storing to what appear to be memory locations.

Explain how a virtual machine monitor (VMM) would virtualize disk I/O in such an architecture. Assume that pure virtualization (and not paravirtualization) is being used, and thus unmodified operating systems that normally run on bare hardware are to run in virtual machines. (Hint: the address space is divided into fixed-size pieces called pages, and each page can be separately protected.)

The VMM would make sure that any form of access to pages containing the disk device's registers would cause faults. Thus any attempt by a virtual machine to access a device register would get the attention of the VMM. The VMM would examine the instruction whose execution was being attempted and check to make certain that the access is proper. In the case of a disk device, it would make certain that a value placed in the device-address register is within the bounds of the virtual device. It would then map the request to the area in which the virtual disk has been allocated on the real disk.

3. We have a priority-based scheduling algorithm that assigns threads priorities in the range of 1 to 10. When the scheduler chooses the next thread to run, it chooses the runnable thread with the highest priority. If there is more than one highest-priority thread, it chooses the one that has been runnable at that priority the longest.

We'd like to give favored treatment to interactive threads, which, for our purposes, are threads that require infrequent short bursts of processor time. To accomplish this, the scheduler increases the priority of all threads by one every second and decreases the priority of the currently running thread by one every tenth of a second. Describe the circumstances under which this approach will not lead to favored treatment of interactive threads and explain why this is so.

If the system is very busy with many runnable threads, then, since all threads will have to wait a long time before they get to run, all will have had their priorities improved to the highest possible value. Thus all will get equal treatment, interactive or not.

4. Recall that the Rhinopias disk drive has the following characteristics:

Rotation speed	10,000 RPM
Number of surfaces	8
Sector size	512 bytes
Sectors/track	500–1000; 750 average
Tracks/surface	100,000
Storage capacity	307.2 billion bytes
Average seek time	4 milliseconds
One-track seek time	.2 milliseconds
Maximum seek time	10 milliseconds

- a. Suppose we are using S5FS on Rhinopias and we've doubled the block size from 512 bytes to 1024 bytes. We now use the file system to read a large file sequentially. Is the maximum expected transfer rate roughly doubled? Explain.

Yes. Since the blocks of a file are randomly scattered, fetching each requires an average seek time plus an average rotational delay. The time for the actual transfer of a block is negligible in comparison. By doubling the block size, after each seek and rotational delay, twice as much data is transferred as before.

- b. We've done the same with FFS. Again we use the file system to read a large file sequentially. Is the maximum expected transfer rate roughly doubled? Explain. Assume

that we're using two-way block interleaving: when we allocate successive blocks on a track, one block is skipped. Thus, in the best case, every other block of a track is allocated to a file.

In the best case, in which the blocks of a file are all within a cylinder group and arranged on tracks as efficiently as is permitted, each rotation of the disk causes half a track's worth of data to be transferred. This is unaffected by the block size.

c. *Why should we care about the block size in FFS?*

Consider the case in which FFS is being used on a busy server, with concurrent requests for blocks from many files. The situation could be as bad as things are with S5FS, in which successive requests are to random disk locations. Thus the analysis given in part a for the speedup in S5FS applies here.

If you do all of the following correctly, you'll get an A regardless of how well you do on the first four problems. If you miss any of the following, your grade will be based solely on how well you do on the first four problems.

5. *Gary Kildall is an important figure in the history of operating systems. What did he do?*

He wrote the CP/M operating system, which is what QDOS, the predecessor of MS-DOS, was based on (or, according to some, copied from).

6. *How many steps are there between the first and second floors of the CIT building? How many between the third and fourth floors?*

24 and 20.

7. *Unix System III was a commercial version of Unix released by AT&T in the 1980s. What was the next release called?*

Unix System V.

8. *What was the version of Solaris that was released after Solaris 2.6?*

Solaris 7.

9. *Who was the president of Brown when the Brown Computer Science department was founded? Who was the governor of Rhode Island? Who was the president of the United States?*

Howard Swearer, J. Joseph Garrahy, Jimmy Carter.

10. *The Brown CS department's first general-purpose computer was a Digital VAX-11/780. Eventually the department owned three VAXes, named Nancy, Sluggo, and Skyler. By the time the department moved to the CIT building, these computers were obsolete. For a short period of time, before everyone started using Sun workstations, much of the department's computing was moved to another time-shared system, a 12-processor Encore Multimax. What did we name this computer?*

Zaphod.