CS 167 Midterm Exam

Spring 2016
Two Hours, Closed Book

Please answer each question in the space provided. Use the back of the page if necessary.

Do all of questions 1 through 4.

1. In the *mthreads* assignment you implemented user threads on top of POSIX threads (referred to as LWPs — lightweight processes). This is known as a two-level threads implementation. One detail is the number of LWPs that should be created. POSIX threads allows for two-level implementations (though the Linux implementation is not done this way). It provides the routine *pthread_setconcurrency* to provide “advice” to the pthreads library on how many LWPs should be created. How this advice is used is not specified, but it has been used as a specification of a lower bound on the number of LWPs that should be created. (Note that “pthread_setconcurrency” is mentioned only as background information. You don’t need to refer to it in your answers.)

    a. Why is setting such a lower bound important on multicore systems?

b. Could there be problems, even on single-core systems, if there are not enough LWPs? Either give a specific example of such a problem or provide a careful argument as to why there wouldn’t be problems. What we are looking for is a major problem that would go away with the addition of an LWP. Assume the system has at least one LWP.

2. 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.

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

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.

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?

3. In systems using x86 architectures, devices are controlled via registers that are mapped into the address space. So, for example, 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.

   How would a virtual machine monitor (VMM) virtualize device I/O in such architectures? 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.)

4. Assume we’re using the Rhinopias disk drive.

   a. Suppose we are using S5FS and we’ve doubled the block size from 512 bytes to 1024 bytes. Is the maximum expected transfer rate roughly doubled? Explain.

   b. We’ve done the same with FFS. 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.

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

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. Roy Tomlinson, who is credited with the invention of email and, in so doing, with the use of the @ sign for email addresses, recently passed away. He was a contractor working on the development of a computer network. Which network was this?

6. Early Unix systems, which were interconnected via telephone lines, used a different syntax to specify an email address. What was this syntax?

7. Dynamic linking was introduced into both Unix and Windows in the early 1990s. On which operating system, and roughly when, was the concept first developed?

8. Gene Amdahl, who passed away recently, was the founder of the Amdahl Corporation, a maker of mainframe computers that competed with IBM. He was prominent in the computer world before forming his company. What was he most famous for before then?

9. Douglas Engelbart passed away in 2013. He was a computer visionary who is most famous for what invention?

10. What female computer scientist (still living) is credited with the development of the first operating system for a personal computer?