*Department of Electrical Engineering and Computer Science*

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

## 6.033 Computer Systems Engineering: Spring 2011

# Quiz I

There are 10 questions and 7 pages in this quiz booklet. Answer each question according to the instructions given. You have **50 minutes** to answer the questions.

Some questions are harder than others and some questions earn more points than others—you may want to skim all questions before starting.

For true/false questions, you will receive 0 points for no answer, and negative points for an incorrect answer.

If you find a question ambiguous, be sure to write down any assumptions you make. **Be neat and legible.** If we can't understand your answer, we can't give you credit!

**Write your name in the space below.** Write your initials at the bottom of each page.

**THIS IS AN OPEN BOOK, OPEN NOTES, OPEN LAPTOP QUIZ, BUT DON'T USE YOUR LAPTOP FOR COMMUNICATION WITH OTHERS.**

**CIRCLE your recitation section number:**

| | | |
|---|---|---|
| **10:00** | 1. Lampson/Pesterev | |
| **11:00** | 2. Lampson/Pesterev | 3. Ports/Mutiso |
| **12:00** | | 4. Ports/Mutiso |
| **1:00** | 5. Katabi/Raza | 6. Strauss/Narula |
| **2:00** | 7. Katabi/Raza | 8. Strauss/Narula |

*Do not write in the boxes below*

| 1-4 (xx/40) | 5-7 (xx/30) | 8-9 (xx/20) | 10 (xx/10) | Total (xx/100) |
|---|---|---|---|---|
| | | | | |

**Name:**

# I Reading Questions

**1. [10 points]:** Which of the following are true statements about the Eraser system, according to the paper by Savage *et al*?

**(Circle True or False for each choice.)**

**A. True / False** Eraser always detects and reports bugs that would cause a program to deadlock (where all of the programs threads wait concurrently trying to acquire locks).

**B. True / False** If a program contains a race condition bug, Eraser needs to observe an interleaving of program threads where that bug causes a race before Eraser can report that the bug exists.

**C. True / False** Because Eraser modifies a binary program with its own instrumentation routines, Eraser would be an ideal tool to identify locking errors in a program which uses custom memory allocation routines, has its own special-purpose synchronization plan that does not use standard acquire and release functions, and for which no source code is available.

**D. True / False** For each memory location in the *Exclusive* state, Eraser sets the candidate set for that location to all the locks held by the single thread that wrote to that memory location.

**2. [10 points]:** Which of the following statements about VMware's x86 virtual machine monitor are true, according to the paper by Agesen *et al*?

**(Circle True or False for each choice.)**

**A. True / False** The x86 architecture originally contained instructions that were non-virtualizable using trap-and-emulate virtualization.

**B. True / False** Shadow page tables store the mappings from guest physical to host physical memory addresses.

**C. True / False** VMware's VMM prevents the guest from accessing the memory used for the VMM's internal state by unmapping the corresponding entries in the guest's page table.

**D. True / False** VMware's binary translation-based VMM translates guest kernel-mode instructions but not guest user-mode instructions.

**E. True / False** A monitor using hardware-assisted virtualization (Intel's VT-x or AMD-V) is always faster than one using binary translation.

**Initials:**

**3. [10 points]:** For v7 Unix processes as described in the paper by Ritchie and Thompson, which of the following statements are true?

**(Circle True or False for each choice.)**

**A. True / False** In Unix v7's time sharing, all processes receive an equal fraction of CPU time.

**B. True / False** In Unix v7, a parent process and its child process can have exactly the same memory contents but execute different instructions.

**C. True / False** In Unix v7, two child processes can communicate via a pipe only if the pipe is created by a common ancestor (where a process can be considered to be its own ancestor).

**D. True / False** Two processes in Unix v7 can communicate via shared memory, regardless of whether they have a common ancestor or not.

# II Client/server

**4. [10 points]:**

Which of these statements about client/server organization are true or false?

**(Circle True or False for each choice.)**

**A. True / False** In a client-server system, the client and the server always run on different machines.

**B. True / False** A client-server system can be faster than a system using the same code to do the actual work, but in which everything runs in the same process, in spite of the added cost of communication between the client and the server.

**C. True / False** In a client-server system that uses remote procedure call, the only important difference from a similar system that uses local procedure call is that the remote calls may be substantially slower.

**D. True / False** A server that wants to provide at-least-once semantics to a client must store a table of previous responses.

**E. True / False** X Windows is not an example of a client-server system because the so-called X server runs on the client's machine.

**Initials:**

# III Concurrency

Ben Bitdiddle wants to implement a concurrent application. Instead of doing the sensible thing and holding a lock while manipulating shared data, Ben decides to be clever. Ben's code, shown below, runs in two separate threads on a computer with two CPUs; no other threads are running on the system. There are three shared global variables, initialized before the threads start: `i` is initialized to zero, `x` is initialized to a three element array of zeros, and `x_lock` is in the unlocked state.

```
Thread 1:
   while i < 3 do:
      while i % 2 == 0 do:
         # (while i is even)
         yield()
      acquire(x_lock)
      x[i] = 1
      i = i + 1
      release(x_lock)
```

```
Thread 2:
   while i < 3 do:
      while i % 2 == 1 do:
         # (while i is odd)
         yield()
      acquire(x_lock)
      x[i] = 2
      i = i + 1
      release(x_lock)
```

**5. [10 points]:** After both threads reach the end of their code segments, which of the following are possible values for the contents of the x[] array?

**(Circle True or False for each choice.)**

**A. True / False** x=2,1,1

**B. True / False** x=2,2,2

**C. True / False** x=2,1,2

**D. True / False** x=2,2,1

**Initials:**

Now suppose that thread 1 instead runs the following variation below. Thread 2 runs the same code as before (shown below for convenience), and global variables are initialized in the same way as before.

```
Thread 1:
   a = 0
   while i < 3 do:
      while i % 2 == 0 do:
         # (while i is even)
         yield()
         a = i
      acquire(x_lock)
      x[a] = 1
      i = a + 1
      release(x_lock)
```

```
Thread 2:
   while i < 3 do:
      while i % 2 == 1 do:
         # (while i is odd)
         yield()
      acquire(x_lock)
      x[i] = 2
      i = i + 1
      release(x_lock)
```

**6. [10 points]:** After both threads reach the end of their code segments, which of the following are possible values for the contents of the x[] array?

**(Circle True or False for each choice.)**

**A. True / False** x=2,1,1

**B. True / False** x=2,2,2

**C. True / False** x=2,1,2

**D. True / False** x=2,2,1

**E. True / False** one or both threads might not reach the end of the code segments.

**7. [10 points]:** Having come to his senses, Ben wants to repair his code so that it doesn't take him nearly this long to understand its behavior. Which of the following options would make it easier to reason about the concurrency in Ben's code, while not introducing new errors?

**(Circle True or False for each choice.)**

**A. True / False** Remove acquire() and release() to avoid reasoning about locks.

**B. True / False** Move acquire() and release() around the entire outer `while` loop.

**C. True / False** Acquire and release the lock around every single access to `i`.

**D. True / False** Acquire the lock before the outer `while` loop and release it only around the `yield` call.

**Initials:**

# IV Operating systems

In many modern Unix systems the kernel code and a user program share a single address space on Intel x86 processors, yet the kernel is protected from malicious programs. The x86 has a bit `U/K` that specifies whether the processor is in user mode or kernel mode. Each x86 page table entry contains the physical page address, along with several bits, including:

- A present bit (P), indicating whether the virtual page should be accessible.
- A writable bit (W), indicating whether writes to the virtual page are allowed.
- A user bit (U), indicating whether this page should be accessible from both kernel and user modes (if the U bit is set), or whether it should be accessible only from kernel mode (if the U bit is not set).

**8. [10 points]:** Which of the following statements must be true to protect the kernel's data from either reads or writes by user programs, while allowing the program to read and write its own data, and allowing the kernel to operate?

**(Circle True or False for each choice.)**

**A. True / False** The kernel must set the U and P bit in the page table entries for text (i.e., code) pages of the user program.

**B. True / False** The kernel must clear the U bits and set the P bits in the page table entries for text (i.e., code) pages of the kernel program.

**C. True / False** The kernel must program the hardware to set the U/K bit to K when entering the kernel.

**D. True / False** The kernel must program the hardware so that the user program can enter only at carefully-chosen addresses.

**E. True / False** The kernel should not execute load instructions for user addresses.

**F. True / False** The kernel should not execute store instructions to user addresses.

**9. [10 points]:** Which of the following statements relating to the complexity of Linux, as compared to the v7 version of Unix system as described in the paper by Ritchie and Thompson are true?

**(Circle True or False for each choice.)**

**A. True / False** Linux has many more developers than v7 Unix.

**B. True / False** Linux has many more lines of code than v7 Unix.

**C. True / False** Only Linux uses client/server to enforce modularity within kernel (and not v7 Unix).

**D. True / False** Only Linux uses virtual memory to isolate user processes (and not v7 Unix).

**E. True / False** Only the Linux kernel supports multiple processes (and not v7 Unix).

**Initials:**

# V Virtual machines

A guest OS kernel executes a single process, using the page table shown below on the left, stored in guest physical page 4. The virtual machine monitor uses the table shown below on the right to translate guest physical pages to host physical pages.

| Virt Page | Phys Page | P | W | U |
|---|---|---|---|---|
| 0 | 5 | 1 | 1 | 1 |
| 1 | 9 | 0 | 1 | 1 |
| 2 | 11 | 1 | 0 | 1 |
| 3 | 4 | 1 | 1 | 0 |
| 4 | 7 | 1 | 1 | 0 |

| Guest phys page | Host phys page |
|---|---|
| 4 | 2 |
| 5 | 6 |
| 6 | 4 |
| 7 | 3 |
| 11 | 8 |

**10. [10 points]:** Fill in the two shadow page tables below that the VMM should use when running the VM. The VMM loads the *user* shadow PT when running VM code with the guest U/K bit set to user, and the VMM loads the *kernel* shadow PT when running VM code with the guest U/K bit set to kernel. The VMM performs no binary rewriting for any guest instructions that read or write memory. You can use a dash (—) to indicate values that don't matter. (Do not rely on demand-filling these shadow page tables whenever possible.)

**User shadow PT, stored in host phys. page 12**

| Virt Page | Phys Page | P | W | U |
|---|---|---|---|---|
| 0 | 6 | | | |
| 1 | | | | |
| 2 | 8 | | | |
| 3 | | | | |
| 4 | | | | |

**Kernel shadow PT, stored in host phys. page 13**

| Virt Page | Phys Page | P | W | U |
|---|---|---|---|---|
| 0 | 6 | | | |
| 1 | | | | |
| 2 | 8 | | | |
| 3 | | | | |
| 4 | | | | |

# End of Quiz I

Please double check that you wrote your name on the front of the quiz,
and circled your recitation section number.

**Initials:**