CS240
Advanced Topics in Operating Systems
Quiz 2 – Spring 2008

OPEN BOOK, OPEN NOTES

!! You should skip 10 points worth of questions !!

Your name: ______________________________

Stanford ID: ______________________________

In accordance with both the letter and the spirit of the Stanford Honor Code, I did not cheat on this exam. Furthermore, I did not and will not assist anyone else in cheating on this exam.

Signature: ______________________________

The quiz has 12 questions. You have 50 minutes to complete 10 of them. Some questions may be much harder than others.

I Short answer

1. [5 points]: Your C runtime provides a function:

```
// Return object (if any) that addr is contained within: base holds
// start, size the object size in bytes.  Returns 0 if addr is not
// within any object.
int getobject(void *addr, void **base, unsigned *size);
```

You replace the object tracking code in the failure oblivious system and rewrite it to use `getobject` on every load and store to determine what object, if any, a pointer is contained within. Will the failure oblivious system likely work better, worse, or no different? (Make sure you state why!)

2. [5 points]: Nooks: You see the following code in a driver:

```
lock();
p->y++;
unlock();
```

If `p` came from the kernel, what is broken about this code? How would you have to fix it?

3. [5 points]: Nooks: Violating linux religion, you write a bunch of device drivers that use a lot of (well-crafted and well-placed) `assert` statements and rerun the experiments in Figure 6 and 7: what changes might you plausibly see and why?

4. [5 points]: LFS: What ordering rules do you have between writing out segments and checkpoints? Assume you crash and read in a valid checkpoint. What else do you have to do before you can start using the file system?

5. [5 points]: LFS: The segment usage table records the number of live bytes in a segment. You write a file. Explain how to update the segment table — which entries you would have to update, and how you would determine them.

6. [5 points]: Sketch the eXplode checker (the mutate and check method) for the CVS check that the paper describes. Make sure you provide the intuition for what you are doing!

7. [5 points]: Assume it's the 1980s, so computers are too slow to efficiently do SHA1 or compression tricks. Sketch how you would retrofit two ideas from LBFS onto NFS (not compression and not hashing!) in a way that would clearly improve performance or functionality.

8. [5 points]: You run NFS on top of xsynfs instead of ext3. What kind of performance improvement do you expect to see compared to the experiments in the paper? (Justify your answer.)

II “I am only sorry when I am caught.”

xsynfs shows the power derived from doing anything you want as long as it can’t be observed. Assume we change failure oblivious to exploit a similar dynamic. We mark registers and memory locations that contain fabricated values as “tainted”.

9. [5 points]: How might you change the OS to use these tainted annotations so that it does not make “made up” values visible to the user? What do you have to do on memory deallocation, both from `malloc` and function call return? (Hint: think about eraser and memory allocation.)

10. [5 points]: Give two scenarios: one where you would work better than the original failure oblivious system, the other where you would work worse.

III Receive livelock

Mogul and Ramakrishnan re-architected the Digital Unix kernel so as to eliminate the livelock problem. The following pseudo code corresponds to the polling thread they describe:

```
poll ()
{
  int io;

  for (;;) {
   foreach (device) {
     if (clock_interrupt) {
        clock++;
        io = 1;
        for (i = 0; i < nthreads; i++) {
          if (thread[i].queue is full) {  // should thread i run?
            io = 0;
            break;  // let the scheduler schedule a user-level process
          }
        }
     } else {  // network device
       for (maximum of n pkts on device) {
         switch (type) {
         case RECEIVE:
           do device-specific stuff (e.g., copy packet from card);
           process_receiveintr(pkt);
           break;
         case SENDCOMPLETE:
           do device-specific stuff;
           process_sendcomplete(pkt);
           break;
         }
       }
     }
   }
   if (io) enable_interrupts(); // enable network interrupts
   wait (polling);
  }
}
```

11. [**5 points**]**:** What problem does the statement

```
for (maximum of n pkts on device)
```

in the function poll() address? Be concise and brief.

12. [**5 points**]**:** What problem do the following statements

```
    if (thread[i].queue is full)  // should thread i run?
      io = 0;
      break;   // let the scheduler schedule a user-level process
    }
```

in the function poll() address? If you omit the statement, what problems can occur? Be concise.