CS240
Advanced Topics in Operating Systems
Final – June 11, 2007

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 14 questions. You have 75 minutes to complete them. Some questions may be much harder than others.

I Short answer

1. **[5 points]:** nesC's heavy inlining reduces code size and typically decreases the running time of any function. Occasionally, however, it greatly increases interrupt handler overhead. What might cause this to happen, and sketch a plausible example.

2. **[5 points]:** LFS: Give the sequence of directory and file i-node accesses involved in opening "/etc/passwd" and reading its contents into memory. Be clear about any information you have to "just know."

3. [5 points]: The LFS paper claims that systems such as NFS "introduce synchronous writes when none previously existed." What is a concrete example of what they mean? If you eliminated this NFS behavior what is a concrete example of a problem that could result?

4. [5 points]: Suppose you change the CONDWRITE operation in LBFS to take a normal NFS file handle and directly overwrite the target file in place. Give a scenario in which this modified LBFS would consume more bandwidth. (Assume the same implementation restrictions as the paper—namely no modification to the server's kernel or underlying local file system implementation.)

5. [5 points]: The NFS paper claims that adding locks is easy. You believe them, and "add" locking by just calling Chubby (of BigTable fame) to acquire and release locks but make no other changes. Assume your NFS implementation provides close-to-open consistency and no failures happen. You run a set of programs that (in theory) correctly use locks to read and write the same set of files. However, things don't work as you expect. Explain what the problem is. How would you change the cache consistency protocol to get more reasonable behavior?

6. [5 points]: In what way are large NFS writes much slower than necessary? Describe how to fix it being very clear what gets sent to the server. (Hint: think about how LBFS does writes, but don't use hashing or compression.) Make sure your solution works if the server crashes in the middle and comes back up.

7. [5 points]: How does a 0-term lease for the system in the original leases paper actually work? I.e., if a 0-term expires immediately, how could the client ever operate on data? (You may have to read between the lines in the paper.) Why is a 0-term lease better than a very short one?

8. [5 points]: The BigTable paper states that clients "can reason about the locality properties of the data represented in the underlying storage" and "can control locality of their data through careful choices in their schemas." Explain how BigTable is designed to make these assertions true.

9. [5 points]: Give two examples (from any of the papers this quarter) where a deallocation of "A" followed by an allocation of "B" could cause the same underlying physical resource to get reused, causing problems and what the solution was. The example should not be contrived.

II HiStar

10. [10 points]:

Suppose four distinct categories have been allocated in the system, r, w, s, and n. A thread T exists with the following label and clearance:

$$\begin{aligned} L_T &= \{r\star,\ w\star,\ \mathbf{1}\} \\ C_T &= \{r\,\mathbf{3},\ w\,\mathbf{3},\ \mathbf{2}\} \end{aligned}$$

A container D, a segment S in D, and a network device N also exist, and have labels, respectively, of:

$$\begin{aligned} L_D &= \{r\,\mathbf{3},\ w\,\mathbf{0},\ \mathbf{1}\} \\ L_S &= \{s\,\mathbf{3},\ \mathbf{1}\} \\ L_N &= \{n\,\mathbf{2},\ \mathbf{1}\} \end{aligned}$$

Assume that no gates or other threads exist on the system, and that S does not exist in any other containers besides D. Which of the following statements is true about what T is permitted to do?

Circle all that apply; there may be more than one answer. (2 points each)

A. Thread T can read segment S without adjusting its own label L_T.

B. Thread T can read S, but only after adjusting L_T. Once it examines S, T will be unable to send messages over the network.

C. Thread T can delete segment S.

D. Thread T can receive data from the network, possibly after adjusting its label L_T.

E. Thread T can create a new segment S' in D with the same label as the network device (i.e., $L_{S'} = \{n\,\mathbf{2}, 1\}$), then create another thread T' that receives data from the network and stores it into S'.

III Exokernel

For this question you need to design an interface for packet reception on an exokernel system. **Your solution should be fast and adhere to the exokernel religion.**

The network hardware on your system uses DMA to copy incoming packets into a pre-specified range of physical memory (whose extent must be large enough to contain the largest possible packet). Assume the exokernel uses the packet filter system discussed in class to decide which process a packet belongs to.

11. [5 points]: What abilities should the libOS have? Give a high-level design of a simple system to provide these abilities. You are allowed to copy network data from exokernel to libOS buffers. Make sure you think about any starvation issues.

12. [10 points]: Design a "zero copy" system that does not copy data between exokernel and libOS buffers. Note, packets for multiple libOSes will arrive on the same card and libOSes should only see their own packets. Describe what happens to a packet from the time it is received through being "delivered" to a libOS. When and by whom (exokernel or libOS) are buffers allocated, freed, and mapped into libOS' address spaces?

IV 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);
  }
}
```

13. [5 points]: What problem does the statement

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

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

14. [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 block, what problems can occur?) Be concise and brief.