CS240 Quiz 3 – 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 eight 5-point questions and two 10-point questions. You have 50 minutes to complete them; you may skip 10 points of questions. Some questions may be much harder than others. Be concise.

I Short answer

1. [5 points]: Consider the executable code snippets (both transformed and not transformed) in Boehm's section 4.1 and 4.3: which of these (if any) will Eraser *not* give a warning about and why?

2. [5 points]: LFS: Consider figure 7: at 0% utilization of the disk, where should no-variance intersect the y-axis? (Note: the figure is confusing so you can't trust it that much.) If we re-ran this experiment with real data rather than synthetic, what would you expect to happen for "LFS improved"?

3. [5 points]: You create a new file, which requires you to modify an existing directory and allocate a new inode and data block(s). When you flush these back to disk, give the correct order you should write these out in (and why), and two other orders that would lead to unrecoverable errors after a crash (i.e., that you couldn't fix by running a repair program).

4. [5 points]: If a computer were infinitely fast, could it livelock? As we make the computer slower and slower, could the Mogul system livelock? (This last part is a bit tricky: make sure you give your intuition. Note that once you get an interrupt, further interrupts are blocked until explicitly re-enabled.)

5. [5 points]: You add LBFS style leases with close-to-open semantics to the original NFS system. Consider the following code:

```
        fp = open("foo");
                     <----- Receive invalidation message.
        read(fp, buf, ...);
        close(fp);

...  some other code ...
```

Assume you receive an invalidation after the `open` but before the `read`. At what point would the client fetch the more recent copy and how would it know to do so?

6. [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 examples should not be contrived.

7. [5 points]: Give three examples from ESX that fall under categories from Lampson's "Hints" paper and specify what Hint it falls under.

8. [5 points]: Give three *uncontrived* examples from papers other than ESX that fall under the category of separating the common and worst case. Feel free to provide some intuition.

9. [10 points]:

In each of the five blanks below, write one the following words: **physical**, **machine**, or **virtual**. (2 points each)

An application is running on a guest operating system which is running on VMWare ESX. The application makes the following system call: `write (fd, buf, 8192);`. Variable `buf` contains a ____________ address.

The primary page tables, which are constructed by the guest operating system, map ____________ pages to ____________ pages. ESX, running underneath the guest OS, maintains *shadow* page tables that map ____________ pages to ____________ pages.

Consider the following Swift code (using the language described in “Secure Web Applications via Automatic Partitioning,” by Chong et al.):

```
public class StupidWebApp {
  int{*->*; *<-*} secret;
  int{*->client; *<-*} v1;
  int{*<-*} v2;
  int{*<-client} v3;

  void DumbFunction{*->client} (Integer{*<-client} num)
    where authority(*), endorse({*<-*})
  {
    int n = num.intValue ();
    int w = v1 + v2;
    int x = v2 + v3;
    int y = v3 + secret;
    int z = secret;

    if (secret > 3)
      n++;
    if (num.intValue () > 3)
      z++;

    ...

  }
}
```

Assume that * acts for `client` and that all labels must be simplified accordingly. Assume further that that the compiler infers the labels of variables `n`, `w`, `x`, `y`, and `z`, based on assignments in the function. Be concise.

10. [2 points]: What is the label of `n`, and why?

11. [2 points]: What is the label of `w`, and why?

12. [2 points]: What is the label of `x`, and why?

13. [2 points]: What is the label of `y`, and why?

14. [2 points]: What is the label of `z`, and why?