CS 240 Quiz Questions 2002 - 2005

Stanford University
Computer Science Department

April 26, 2005

These quizes were all open-book exams. In general you were asked to answer somewhere between 8 out of 10 to 10 out of 12 given questions. The questions below are actual questions that were given.

Write all of your answers directly on the paper. Make your answers as concise as possible. Sentence fragments ok.

NOTE: We will take off points if a correct answer also includes incorrect or irrelevant information. (I.e., don't put in everything you know in hopes of saying the correct buzzword.)

Stanford University Honor Code

In accordance with both the letter and the spirit of the Honor Code, I did not cheat on this exam nor will I assist someone else cheating.

Name and Stanford ID:

Signature:

Answer the following questions and, in a sentence or two, say *why* your answer holds.

1. Gabriel uses "pc-losering" as an example of how "worse is better" can triumph even when it makes an interface more complex. Taking his explanation of how Unix works at face value: explain how a luser-level library could make the Unix interface into the Right Thing. Does this invalidate his argument?

2. The Therac paper is a study in severed feedback loops, ranging from AECL not informing customers of problems in the Therac-25 to the video camera being off when a patient was zapped. From the paper, give three (other) examples of such missing feedback loops.

3. What would be a difficulty in detecting the Therac races using an Eraser-type approach?

4. The Therac code in Figure 4 on page 34 shows an overflow error involving the "Class3" variable. If we assume the code in Figure 4 accurately represents the code in the Therac, describe the race that exists between Set-up Test and Lmtchk.

5. The Eraser paper claims that they have never observed more than 10,000 distinct sets of locks for a given program. Give a data structure and locking policy that would pass this limit.

6. Eraser only ensures that data is protected by a consistent set of locks. Give an intuitive sketch of a class of race conditions it will miss, an example, and explain how Eraser could be extended to handle them.

7. Assume we have two threads, T1 and T2, that execute the following code in the following order:

```
  int *q = NULL; /* shared global variable */

          T1                              T2
1:   int *p = malloc(sizeof *p);
2:   *p = 0;
3:   q = p;
4:                                        while(q == NULL)
```

```
 5:                                           ;
 6:                                      *q = 1;
 7:                                      q = 0;
 8:  while(q != NULL)
 9:     ;
10:  printf("p = %d\n", *p);
11:  exit();
12:                                       exit();
```

Assuming naive memory semantics (if you don't know what this means, you are just fine): What will Eraser do for this code? Is its behavior correct?

8. The set of locks that are acquired at any given time is a global property (e.g., switching threads has no effect on it). Given this, why does Eraser use per-thread locksets rather than a single global lockset? Give an intuition and a simple example of the problem a global lockset would cause.

9. The Eraser paper claims "A write access from a new thread changes the state from Exclusive or Shared to the Shared-Modified state..." But Figure 4 says that a write by any thread in the Shared state takes it to the Shared-Modified state. Which is right?

10. Eraser, like most tool papers, defines a race condition as a reference to shared memory without holding a (consistent) lock. Give two intuitive, short examples showing that this definition is both too strong and too weak.

11. Eraser uses local locksets — i.e., the locks in a threads lockset are only those that it acquired. Of course, the set of locks that are currently locked is a global property that does not depend on which thread is running. What would happen if Eraser used this global rather than local lockset when checking for races?

12. In Section 4.3 in the Eraser paper, they state that the routine `GmapCh_Write2` passes the address of a stack variable to its children threads. If this variable is on the stack, why does Eraser flag an error? Why can't they annotate this false positive away?

13. Assume only one thread has a pointer to a data location `p`. How would you use this ability to improve Eraser? (Make sure you do not miss races that Eraser currently detects.)

14. For the following snippet of Mesa monitor code:

    ```
    void foo() {
        signal(c);
    }
    ```

    If the code runs on a system with different priority processes, what is a potential way that it will generate useless overhead? Is there an automatic way to fix this problem?

15. Would a Mesa programmer have any use for a Mesa version of Eraser?

16. Give two examples where Mesa makes a "New Jersey" style decision.

17. Explain from the Mesa paper: "...[while] any procedure suitable for forking can be called sequentially, the converse is not true."

18. Your ex-140 partner loudly declares that if the semantics of the "wakeup-waiting switch" is good for naked notifies, it must be good for normal notifies, which should be replaced with them. Can you do this substitution and preserve correctness? (List any assumptions you must make.)

19. The mesa paper claims that because of their semantics on waits, "verification is actually made simpler and more localized" (page 11). Give an intuition for why this is true or false.

20. Linux has a "big kernel lock" (the BKL) originally used to more-or-less turn the OS into one large monitor. Using your knowledge of Mesa: what must Linux do (and why) if a kernel thread holds the BKL and sleeps (i.e., puts itself on a block queue and allows another thread to run)? Give an example of a problem this can cause. (Note: you only need to understand Mesa to answer this question, the details of Linux are completely irrelevant.)

21. Thread 1 consists of a single call to `foo`, which is an entry routine in the following monitor M:

```
---------------------- Thread 1 -------------------------
void thread_1(void) {
      foo();
      bar();
}
-----------------------------------------------------------

---------------------- Monitor M -------------------------
condition c;

entry foo() {
        wait(c);
        return;

}

entry bar() {
        signal(c);
        return;
}
-----------------------------------------------------------
```

Assume you are running on a MESA system and there are other threads on the system. Point out four locations in the code (in either the caller or callee) where thread 1 could get switched out. NOTE: the switches cannot occur for identical reasons.

22. Capriccio, Figure 1: what is a plausible reason that performance for the non-Capriccio systems goes down so quickly?

23. Give two examples of where a static stack allocation scheme would perform better than Capriccio's dynamic approach.

24. The Capriccio paper states that "larger path lengths require fewer checkpoints but more stack linking." What is the intuition behind this statement?

25. The capriccio paper (weirdly!) limits the buffer cache in its experiments to 200MB. Assume you rerun the experiment without this limit and notice that LinuxThreads improves somewhat and capriccio performs much worse. What is a plausible explanation?

26. Assume we only have two jobs, a and b. The VTRR paper would claim that if $W_a(t)/S_a = W_b(t)/S_b$ then we have "perfect fairness." Prove this is true.

27. Why does WFQ run the job with the minimum VFT rather than the job with the minimum VT?

28. What in the WFQ scheduling pattern in Figure 5 is it impossible for VTRR to mirror?

29. Identify several potential weaknesses in the VMS memory system and describe experiments to measure them.

30. You increase the size of the free list to 80% of memory. Describe a workload that will run better, and one that will run worse.

31. VMS has several different parameters. If you were to try to hurt system performance the most by setting a single parameter to its worst value, which one and value would it be? Explain your reasoning in terms of numbers given in the paper. You will need to make assumptions: make sure they are reasonable, and make sure to justify them.

32. Describe the worst-case address space layout that maximizes VMS page table overhead (i.e., the amount of memory consumed by page tables given the amount of memory mapped). Be very specific.

33. The experimental hypothesis of Figure 3 in the VMS paper is not well evaluated. Give three *important*, concrete limitations and how to fix them.

34. When the reference bit of a base page within a superpage is reset, the superpage management system demote the superpage speculatively, recursively performing the demotions with a probability $p = 1$. Suppose we set $p = 0.5$ instead. Give one reason why the system might perform

better because of that, and one reason why the system might perform worse because of that.

35. When does the coalesce daemon run in the superpage system? What would the effect of running it more often be?

36. Assume you run a process that has a 513K text segment on the Alpha system described in the superpage paper. What are all the different actions in the superpage system that can be triggered when you jump to the first instruction in this process? (Hint: make sure you consider more than just reservations.)

37. Assume superpage management (promotion, demotion, etc) is free. Describe a workload that will perform much worse on the Navarro superpage system as compared to FreeBSD.

38. Superpages are demoted both under memory pressure and when the first write occurs. How does this demotion differ, why does it differ, and when does re-promotion occur?

39. When measuring active memory (for use in idle memory tax calculations), ESX Server keeps three statistical averages: a "slow" average, a "fast" average, and a "very fast" average. Explain the problem that could result if ESX stopped using the "slow" average.

40. Your ex-140 partner (fresh from their triumphant foray into monitor semantics) decides to get the same effect of a balloon driver using an application running on top of the guest OS. On ESX's request the application will aggressively touch many pages of memory, causing them to be "recently used" and the guest OS to page out other, less recently used pages. Is this a bright idea?

41. A balloon driver must use fairly non-portable device driver interfaces. It would have been much more portable if Carl had used a user-level application to do ballooning instead. Would this have worked or not?

42. What bad things would happen if ESX used the average rather than the max of the three moving averages of memory usage?

43. Figure 8(c) of the ESX paper (the graph for the citrix server): explain what the synchronized spike in the "alloc," "active" and valley in "balloon" at (roughly) the 39-42 minute mark imply about what is happening in VMware and in the Citrix server. Assuming the system had 2GB rather than 1GB: what would the shape of the lines be?

44. Most OSes have some sort of `mpin(void *va, unsigned len)` system call, which will pin a range of addresses in physical memory. Explain how to use this system call to replace the need to write a balloon driver. (NOTE: make sure to describe how to communicate with ESX.) What is the advantage/disadvantage of this approach?

45. Assume you want to add ESX memory taxation to the VMS system. In a few sentences, describe why this might make sense and how you would integrate it.

46. List two realistic, specific scenarios where running VMS (or copies of VMS) on top of ESX will give better performance.

47. You use the shares-per-page ratio formula from the ESX paper to adjust the fixed partitions in VMS. However, you change the estimation of f to be based on the number of pages rescued over a 30 second period. Under what two conditions will this be identical to the original VMS scheme? In general, do you expect this estimation of f to work better or worse than the ESX estimation approach?

48. Your nooks extension makes an XPC call to a kernel function:

```
        foo(p);
```

    and the nooks system restarts your extension. Give three different errors that could have occurred in the call to cause this restart to happen.

49. You use the Safe-C compiler in the Rinard et al papers to implement memory protection in Nooks. What changes would you anticipate in Figures 6 and Figures 7 in the Nooks paper (and why!).

50. The failure oblivious guys try to put the nooks guys out of business by applying their approach to a network driver and a disk driver. In which would you expect to see better/safer results and why?

51. Livelock: in Figure 6-3 why does "Polling (no quota)" work badly?

52. The livelock paper has a hack to handle the problem of dropping packets on the "screend" queue. What limit does this solution have for multiple applications getting network data?

53. Assume we have a router that does exactly three things: (1) receives packets, (2) processes packets, and (3) transmits packets. If we use a 3-CPU multiprocessor and put each of these on a single dedicated CPU, can we still livelock? Are there any other problems with this approach?

54. Give two ways that the MTV music video award webserver could suffer from livelock despite the fact that everyone connects to it using TCP, which is flow-controlled.

55. Given the following pseudo-code:

```
// runs when there is a network interrupt.
net_interrupt() {
    while eth1 has packets on receive Q
        // process packets;
            while eth2 has packets on receive Q
        // process packets;
            while eth1 has packets to send
        // send packet
            while eth2 has packets to send
        // send packet
    }
```

    List several problems with this code from a livelock point of view and suggest how to rewrite it to eliminate these problems.

56. The livelock paper states you can use one of two approaches in solving livelock: "(1) do (almost) everything at high IPL, or do (almost) nothing at high IPL." As they describe these approaches, in what sense are they actually the same?

57. Livelock: Why are infinitely large queues not a good idea (even assuming an infinite amount of memory)?

58. Livelock: In 6-3, why does the output packet rate of “Polling (no quota)” drop to almost zero after 6K pkts/sec? What does the same thing not happen with “Polling, no feedback” in 6-4?

59. Assume you could classify packets with what application they were intended for infinitely quickly. What could you improve in the livelock paper and how?