Stanford University
Computer Science Department

Fall 2005 Comprehensive Exam in Software Systems

1. **Open Book, - NO laptop. Write only in the Blue Book provided.**
2. **The exam is timed for one hour.**
3. **Write your Magic Number on this sheet and the Blue Book; DO NOT WRITE YOUR NAME.**

The following is a statement of the Stanford University Honor Code:

A. The Honor Code is an undertaking of the students, individually and collectively:

1. that they will not give or receive aid in examinations; that they will not give or receive un-permitted aid in class work, in the preparation of reports, or in any other work that is to be used by the instructor as the basis of grading;

2. that they will do their share and take an active part in seeing to it that others as well as themselves uphold the spirit and letter of the Honor Code.

B. The faculty on its part manifests its confidence in the honor of its students by refraining from proctoring examinations and from taking unusual and unreasonable precautions to prevent the forms of dishonesty mentioned above. The faculty will also avoid, as far as practicable, academic procedures that create temptations to violate the Honor Code.

C. While the faculty alone has the right and obligation to set academic requirements, the students and faculty will work together to establish optimal conditions for honorable academic work.

By writing my Magic Number below, I certify that I acknowledge and accept the Honor Code.

Magic Number----------

Comprehensive Exam — Systems software

Fall 2005

November 9, 2005

Answer each of the following questions, and give a sentence or two justification for your answer (4 points each).

1. Write the shortest, non-reentrant legal C function you can and why you think it is non-reentrant.

2. Your machine can only do 32-bit loads and stores. What is the locking-related problem in the following code? How would you fix it?

```
struct foo {
        lock_t a_lock; // always held before touching a
        lock_t b_lock; // always held before touching b

        char a;
        char b;
};
...

void update_ab(struct foo *f) {
        lock(f->a_lock);
        f->a++;
        unlock(f->a_lock);

        lock(f->b_lock);
        f->b++;
        unlock(f->b_lock);
}
```

3. Your threaded code has no race conditions. It does have this routine:

```
foo()
        lock(a);
        lock(b);
        ...
        unlock(a);
        ...
```

```
lock(a);
...
unlock(b);
unlock(b);
}
```

Ignore performance: what is wrong here?

4. In what way is a good proportional-share process scheduling algorithm essentially equivalent to a good graphics line-drawing algorithm? (Use a picture in your answer and label it.)

5. You run program A using kernel threads, and then re-run it using user-level threads. How could these two runs behave differently with respect to load and store instructions?

6. Draw the structure of a standard 32-bit virtual address with 4K pages. What bad things happen if you switch the order of the two components?

7. Your (ancient) machine has an 8K, direct mapped, physical cache with 4K pages. Program A is using an 8K, page-aligned array, whose first page maps to physical page 31. Give the set of physical pages that the second page of the array should be mapped to and why.

8. What address space layout will be the best for a linear page table as compared a hashed page table and vice versa? (Make sure to say why.)

9. Give the simplest example of a chunk of data and a predictable access pattern that LRU will perform optimally bad on. In the absence of prefetching, what is the best *realistic* algorithm to use?

10. Joebob, your mortal enemy, gives you a USB stick that you want to mount as a file system on your computer. Give the type of checks that the file system should do before treating the USB data as a valid file system.

11. You create a file on a Unix file system (such as FFS). Roughly: what meta data do you modify, what errors could you get, what order do you write the metadata out in?

12. Among NFS's operations are:

```
// write nbytes from buf into offset of the file named by fh
write(fh, offset, buf, nbytes)

// make directory "name" in directory named by dh
mkdir(filehandle, name)
```

It sends these requests across a lossy network, so may obviously have to retransmit them. Discuss: what problems could occur because of retransmission for these two operations and a (partial) fix.

13. You have a distributed file system. What is the perfect guarantee it could give for cache consistency? What would you have to do to implement this? Is there any way for an application running on top of this perfect consistency could see stale data?

14. Many distributed file systems implement close-to-open consistency. Give an intuitive statement of what this is, and two reasons two reasons you might prefer it over perfect consistency (including one non-performance reason)

15. Let' say you have a network interrupt handler that looks something like:

```
interrupt_handler() {

        while(there are packets to receive)
                pull pkt off network interface
                enqueue(pkt);

        while(there are packets to transmit)
                dequeue from transmit queue
                give to network interface
}
```

You notice that sometimes no packets come out of the system for awhile, in situations where they should. Similarly, you notice that sometimes applications do not run, but should. What problems in this code could cause this behavior? Give a sketch of how to fix it.