**Computer Science Department**
**Stanford University**
**Comprehensive Examination in Software Systems**
**Autumn 1999**

**November 2, 1999**

*READ THIS FIRST!*

1. You should write your answers for this part of the Comprehensive Examination in a BLUE BOOK. Be sure to write your MAGIC NUMBER on the cover of every blue book that you use.

2. Be sure you have all the pages of this exam. There are 2 pages.

5. This exam is OPEN BOOK.

# Comprehensive Exam — Systems software

Fall 1999

**Short-answer (32 points)**

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

1. Without "free" (deallocate) it is easy to write a "malloc" implementation that never fragments memory. (True/false)

2. You profile a web server and notice that over the course of a day the virtual memory it allocates and uses is 1000 larger (measured in bytes) than the system's page cache. The cache cannot provide much benefit for this type of workload. (True/false)

3. Even without locks, a system can deadlock. (True/false)

4. Why do page-based systems use the high bits in a virtual address to specify the virtual page number?

5. Give two examples of when an OS intentionally causes internal fragmentation and say why.

6. You implement a distributed file system that allows clients to cache file blocks. You notice that decreasing the block size results in less network traffic when you revoke a cached block. What's a likely reason for this?

7. Ignoring the overhead of context switching, will I/O utilization on a round-robin scheduling system increase or decrease as the time-slice length is increased?

8. Ignoring context switching overhead, can increasing the time-slice length on a round-robin scheduling system dramatically improve a process's execution time?

**Problem 2 — File system atomicity (18 points)**

The Happy-go-lucky file system (HFS) moves a file from one directory to another by:

1. finding a free slot in the destination directory and inserting the file-name-to-inode mapping in this slot;

2. deleting the file-name-to-inode mapping in the source directory;

3. returning to the user;

4. at some later point HFS writes the cached copies of the modified directory blocks and the inode to disk (in no particular order).

Recall that disks can only atomically write a sector at a time.

1. (9 points) The system crashes. Give two possible file system inconsistencies that you might see.

2. (9 points) In a few sentences, describe how to use synchronous disk writes and file system reconstruction to eliminate these two inconsistencies. (Recall, "synchronous disk write" means when the OS issues a disk write, it waits until the write completes before performing another disk operation.)

## Problem 3 — Synchronization (10 points)

The following sequential code adds and removes characters from an infinite buffer ("buf"). Rewrite the code *without using locks* to work correctly when there are exactly two threads: one producer thread that adds characters, and one consumer thread that removes them. State what assumptions your code makes and provide a short, intuitive correctness argument for your modifications.

```
char buf[]; /* infinite buffer */
int head = 0,   /* producer's position in buf */
    tail = 0,  /* consumer's position in buf */
    n = 0; /* number of characters in buf */

char get(void) {
        /* no characters = infinite loop */
        assert(n > 0);
        /* take character */
        c = buf[tail];
        tail++;
        n--;
        return c;
}

void put(char c) {
        buf[head] = c;
        head++;
        n++;
}
```