CSE121 — Midterm

Yee **Fall '00**

Name and Class Account Login: Answer Key

There are a total of 10 questions on 6 pages. There are 101 points possible. You may not finish the entire exam — don't be discouraged. Wait until the instructor has passed out exams to everybody before you start. Advice: skim through the entire test to determine which of the problems you can solve quickly and work on those first, rather than getting stuck on a hard problem early and wasting too much of your time on it.

When you can start, you should first make sure that you have all the pages, and write your name and your login name on the first page, and your login name on the top of *all subsequent pages*. Pages of this exam will be separated and graded separately — if you fail to write your name at the top of a page, you will not receive credit for answers on that page. **Write clearly**: if we cannot read your handwriting or your pencil smudges, you will not properly get credit for your answers.

This exam is closed book. You are allowed a single sheet of notes You may look at your *own* notes all you want. You may **not** look at anybody else's books, notes, exam, or otherwise obtain help from another human being, artificial intelligence, metaphysical entity, or space alien. If I see your eyeballs wandering, you will get a zero for the exam. If you must look away from your exam/notes to think, look up at the ceiling / into space or close your eyes.

No electronic computation aids are allowed.

Problem	1	2	3	4	5	6	7	8	9	10	**Total**
Score											
Possible	2	10	6	6	10	15	10	14	14	14	101

1 (Class basics) What is the URL for the class web page?

(2pts)

`http://www-cse.ucsd.edu/classes/fa00/cse121/`

2 (OS Architecture) Describe

A. what kind of kernel code are placed in the "upper half" of the BSD kernel, and

B. what kind of kernel code are in the "lower half".

How are the halves invoked?

(10pts)

The upper half of the kernel contains the kernel code that handles requests due to system calls from user-level program.

The lower half of the kernel contains code that are invoked as part of interrupt/exception processing.

3 (OS Architecture) How do the upper and lower halves of the kernel communicate in the BSD kernel?

(6pts)

The upper and lower halves communicate via shared data structures. Data structure corruption is prevented via the `spl` routines, which provide mutual exclusion between the upper and lower halves.

4 (Virtual Memory) Is the virtual memory system code in the upper half of the kernel, the lower half, or both? Explain.

(6pts)

The virtual memory system code is in both the upper and lower half. The main body of the machine independent (vm) and the machine dependent (pmap) modules are in the upper half and handles virtual memory requests that arise from `fork`, `mmap`, etc; the lower half code are the code that handles page faults, which are exceptions and so must be in the lower half.

5 (Virtual Memory) What are shadow objects? How does the kernel use them, and for what purpose?

(10pts)

Shadow objects are used to hold private copies of memory pages that were created due to copy-on-write operations.

6 (Pagers) Two parts:

A. What are the different kinds of pagers in BSD? (enumerate them; 3pts)

B. Describe what kinds of memory are backed by each of these pagers (detailed description; 12pts)

(15pts)

There are 3 pagers:

A. **vnode pager**. The vnode pager handles pages mmapped from a file. This could be text pages mmapped in read-only mode from executable files, initialized data pages mmapped in copy-on-write mode from executables, or general file data that's mmapped into memory.

B. **swap pager**. The swap pager handles anonymous memory. This would usually be uninitialized data and stack pages, though memory that's mmapped from `/dev/zero` or using a file descriptor of `-1` would also be allocated from the swap pager. Similarly pages created by a copy-on-write copy would be backed by the swap pager.

C. **device pager**. The device pager provide mapped memory access to device memory. An example of this is frame buffer memory from a video card.

7 (Kernel Internals) What is `hardclock`? What is `softclock`? What are the differences?

(10pts)

`hardclock` is a low-level kernel routine that is called on every hardware clock interrupt (HZ times per second).

`softclock` is a kernel routine that is only invoked by `hardclock` periodically as needed. `softclock` is responsible for timeouts and callouts.

8 (Drivers) What are "collisions" in device drivers? When do they occur and how are they handled? What happens when there are no collisions?

(14pts)

Device drivers has a slot for recording a single process that is currently invoking the `select` system call with the device in one (or more) of the input, output, or exception descriptor sets. Normally, when the lower half of the driver gets an interrupt showing that I/O is now possible, the `selwakeup` routine (used by all drivers that implement `select` functionality) can `signal` and wakeup the sleeping process via `setrunnable` or `unsleep`.

If more than one process is `select`ing on a device, however, there isn't space to record all of them. This is when a "collision" occurs. When a collision occurs, that it occurred is recorded, and the driver will instead awaken (via `wakeup` on the kernel-global variable `selwait`) all processes that are sleeping due to the `select` system call (sleeping on `selwait`).

9 (Buffer Cache) What are the LOCKED, LRU, AGE, and EMPTY lists? What kinds of buffers are placed in which list, and in what way are they treated differently?

(14pts) LOCKED buffers are never removed; they are used for superblock data and LFS data.

LRU buffers are in least-recently-used order. Whenever a buffer cache hit occurs, the buffer is moved to the end of the ALU list.

The AGE list are buffers that have not shown their usefulness, but might be useful. These include readahead buffers, which would be pushed onto the end of the AGE list. When a buffer is deallocated, e.g., due to a file being unlinked, the associated buffers are placed at the head of the AGE list, since they can be reused as soon as possible.

The FREE list contains buffer headers with which there are no associated physical memory; these buffer headers become useful when, for example, physical memory is released due to buffer allocation (for a smaller buffer).

On a buffer cache miss, buffers are taken from the head of the AGE list, and if the AGE list is empty, the head of the LRU list.

10 (Buffer Cache) What are the routines that comprise the upper interface to the buffer pool? (14pts)

A. `bread` returns a locked buffer, allocating a new buffer and reading the data from the underlying block device if necessary.

B. `brelse` release the lock on the buffer.

C. `bwrite` writes the buffer synchronously.

D. `bawrite` asychnrously writes the buffer; the buffer is marked dirty and a write is scheduled, but `bawrite` returns immediately.

E. `bdwrite` delay-writes the buffer. The buffer is merely marked as dirty, and no writes are scheduled.