Things to know by May 2

Warning. This is not guaranteed to be everything you need to know.

The final exam is 2:00 PM on Tuesday, May 2, in Sitterson 014.

Know everything you were supposed to know for the two midterms and know something about the following terms:

base register	broadcast (message)	CSMA/CD
CSR	device driver	device registers
display	free token	Internet
limit register	local-area network	memory-mapped I/O
mouse	page fault	page table
physical memory	programmed I/O	switch table
token ring	virtual memory	virus
window	X window system	

The final is based on Chapter 1 though Subsection 8.1.4 (pp. 1-255), Chapter 10 up to Subsection 10.1.2.3 (pp. 312-322), and Sections 10.3 and 10.3 (pp. 382-388) of the textbook, the October 1987 *Scientific American* article "Networks for Advanced Computing" by Robert Kahn, the January 1989 *Byte* articles "The X Window System" by Dick Pountain and "The Token Ring" by Brett Glass, Section 5.5.3 (pp. 209-218) of *Computer Science: A Modern Introduction* by Goldschlager and Lister, and Chapter 2† (pp. 13-25) of *Internetworking with XINU* by Douglas Comer.

Understand what a device driver is, how it does its task, and how the operating system is interfaced to it.

Understand how the operating system manages page tables.

Understand how token rings and Ethernets allocate the communication channel.

Understand the roles of and interfaces between clients and servers in a windowing system.

In addition to those problems you looked at the first two midterms, you might consider exercises 1 and 11 of Chapter 10 of Bach's book and exercises 2.9 and 2.10 (p. 25) of Comer's.

The last five pages of this handout is a *copy* of *last* year's final exam in Comp 190. Last year we covered Unix shared memory and Internet name servers. Questions two, three, and four of the closed book section of last year's final relate to these topics. This year we covered process scheduling and paging. Neither was covered last year. Last year we studied token ring, but it does not seem to have appeared on the exam.

The closed book part of this year's final will probably count 36 points instead of 50.

† Yes, Chapter 3 will *not* be covered.

Final exam–May 3 **1988**, 9:00 AM
Closed book section (50 points)

The exam is to be turned in at 12:00 high noon. (Then you can turn in your card for $10, and we can all go out to get pizza.)

Work the closed book section first and turn it in before you consult your books and notes to work on the open book section. For the closed book section, write your answers on the exam itself. For the open book section, write your answers on separate pieces of paper.

University regulations require that you sign the following pledge on the first page of your turned-in exam.

I have neither received nor given any unauthorized aid on this exam. ____________________

Problem 1. (27 points–3 points each)

Give short definitions (one or two phrases or sentences) of the following terms.

broadcast (message)

collision detection

protocol

virtual circuit

"well known" port

u area

reference count

critical region

file descriptor

Problem 2. (12 points–3 points each)

Give a brief description of what the following UNIX system calls do at the user level. *You don't need to describe the implementation but say something about what they return.*

`bind`(s, name, namelen)

`chmod`(path, mode)

`dup`(oldd)

`kill`(pid, sig)

Problem 3. (2 points)

Why is `napoleon` sometimes called `napoleon.cs.unc.edu`?

Problem 4. (3 points)

Differentiate between a name server and a name resolver.

Problem 5. (3 points)

Differentiate between a shared memory "key" and a shared memory descriptor (sometimes also called an "index" or "id").

Problem 6. (3 points)

Describe caching and give at least two examples of it seen in this course.

Final exam–May 3 **1988**, 9:00 AM

Open book section (50 points)

The exam is to be turned in at 12:00 noon. The closed book section should be turned in before you open your books and notes to work the open book section. For the open book section, write your answers on separate pieces of paper.

Problem 1. (5 points)

You are the only person logged into a Unix computer. You connect to another user's directory and type

```
% ls -i foo
```

The system replies with

```
 2050 hi
```

informing you that the directory `foo` has a file `hi` with inode 2050. You then type

```
% cat foo/hi
```

and receive the error message

```
cat:  cannot open foo/hi
```

Name a way the directory `foo` could be protected to make this behavior possible.
Name a way the file `foo/hi` could be protected to make this behavior possible.
In each case, where does the operating system detect the protection violation?

Problem 2. (10 points)

Presently, Unix directories contain file names and their associated inode numbers. Suppose we wanted to "improve" the file system by abolishing inodes and moving the information presently stored in inodes into the directory.
What would be lost and what would be gained by this change?
What system calls would need to be changed?

Problem 3. (5 points)

Suppose you type the command

```
% cat .login
```

to the shell. Describe the system calls the shell uses in executing `cat`.
How would your answer change if the command was

```
% cat <.login
```

Problem 5. (5 points)

Suppose the grades for a course are stored in a read and write protected file as a series of lines each of the format:

```
("242-12-3456" "Pooh, Winnie" 10 10 13 15 30 96 71)
```

Describe how the features of the Unix operating system can be used to implement a single program `mygrades` which will allow each student to see the line containing his or her grades.

Problem 6. (10 points)

Suppose the tape drive on `napoleon` melts this morning and we wish to modify `napoleon`'s operating system kernel so that when a program attempts a tape I/O operation on `napoleon`, the actual tape operation is performed on `dopey`.

Outline (in two or three paragraphs) how the kernel must be changed to accomplish this. Be sure to point out how your solution will use the networking system calls.

Problem 7. (15 points)

Suppose the program on the following page has been successfully compiled and run on a Unix system. (Incidently, it has and executed with no errors.)

Draw a picture showing the pipes created by the processes.

Describe the possible results of executing the program. Pay particular attention to the races between the two children processes. Merely "describing" the possible character(s) output on standard output does not constitute an adequate answer to this question.

By the way, the most likely outputs of executing the program seem to be "`ACa`", "`AaC`", and "`Aae`".

Hint: Identifying the races that are more or less independent makes the problem more manageable.

```
/* The routine shuttle has three parameters:
   * fdin:      a file descriptor which can be read
   * fdout:    a file descriptor which can be written
   * incr:      a small integer
   * shuttle(fdin, fdout, incr)
   *   reads a character from fdin
   *   writes that character to standard output
   *   adds the input parameter incr to the character
   *   writes the incremented character to fdout
*/
shuttle(fdin, fdout, incr)
     int fdin, fdout, incr;
   {
     char c = 'a';
     read(fdin, &c, 1);
     write(1, &c, 1);
     c = (char) ((int) c + incr);
     write(fdout, &c, 1);
   }

main(argc, argv)
     int argc;
     char *argv[];
   {
     int p[2], q[2];
     pipe(p);
     if ( fork() != 0) {
         close(p[0]);
         write(p[1], "A", 1);
         close(p[1]);
         wait(0); }
     else {
         close(p[1]);
         pipe(q);
         if ( fork() != 0) {
             shuttle(p[0], q[1], 2);
             shuttle(q[0], q[1], 3);
             wait(0); }
          else {
             close(q[0]);
             shuttle(p[0], q[1], 4); }}}
```