Things to know by April 10

Warning. This is not guaranteed to be an inclusive list of things to know.

Know something about the following terms:

clock tick	context (of process)	context switch
datagram	environment	event
interrupt	kludge	page
ports (with sockets)	process table	priority
region	round robin	setuid
shell	signal	sleep
stream (virtual circuit)	switch table	time sharing
time slice	trap	u area

The midterm is based on Chapters 6 and 7 and Subsections 8.1.1 through 8.1.4 (pp. 146-255) and Sections 11.3 and 11.4 (pp. 382-388).

Understand the region table and the algorithms that manipulate it. Understand how several processes may share the same text.

Understand *well* the system calls which create processes and connect them with pipes. Be sure you can hand execute C programs that use them.

Understand the systems calls for sending and handling signals.

Understand how sockets are used to communicate between different machines.

You might look at the following homework problems: Chapter 6, exercises 6, 8, 17; Chapter 7, exercises 4, 9, 12, 22, 25, 30, 33, 41, 43; and Chapter 8, exercises 1, 3, and 8. Note: Some of these problems are difficult. Don't expect to come up with great solutions to them but at least understand the problem.

The last four pages of this handout it a *copy* of *last* year's second midterm in Comp 190. The topics covered on last year's midterm are slightly different than those to be covered on this year's. Consequently, some of the questions may make little sense. (In particular, shared memory and devices drivers were on last year's second midterm.) Also, the closed book part of last year's midterm was too long.

Midterm 2–April 6 **1988**

Closed book section (64 points)

The exam is to be turned in at 1:50 pm. 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. (25 points–5 points each)

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

context (of a process)

device driver

memory-mapped I/O

per process region table

shell

Problem 2. (20 points–5 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.*

`execve`(path, argv, envp)

`fork`()

`signal`(signum, function)

`shmget`(key, size, flag)

Problem 3. (6 points)

Differentiate between the three kinds of user id's: real, effective, and saved.

Problem 4. (6 points)

Differentiate between the two kinds of device numbers: major and minor.

Problem 5. (1 points)

Name a patented feature of the Unix operating system.

Problem 6. (6 points)

Assuming pages are 1024 bytes long, make up a page table for accessing the logical addresses from 0 to 5764 and state the real address corresponding to logical address 2548.

Midterm 2–April 6 **1988**

Open book section (36 points)

The exam is to be turned in at 1:50 pm. 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. (15 points)

Suppose the following C program:

```
main(argc, argv)
     int argc;
     char *argv[];
  {
     int p[2], pid;
     char c = 'x';
     pipe(p);
     ppid=fork();
     if (pid != 0)
      {
     c := 'y';
     write(p[1], &c, 1);
     close(p[1]);
     read(p[0], &c, 1);
     write(1, &c, 1);
     kill(pid, SIGBUS);
     wait(0);
      }
     else
      {
     read(p[0], &c, 1);
     fork();
     if (getpid()==pid)
         c := 'z';
     close(p[1]);
     write(1, &c, 1);
      }
  }
```

is compiled and the compiled code is stored in the file `c190`. Describe what can happen if the command

```
% c190 hello world
```

is executed and all of the `fork` system calls work properly. (`getpid()` returns a processes own process id.) Merely "describing" the possible character(s) output on standard output does not constitute an adequate answer to this question.

Problem 2. (15 points)

We want to implement an anonymous suggestion box in Unix.

Every user will have file called `suggestions` stored in his or her home directory We want to create a program `suggest` such that whenever any user types the command:

```
% suggest smith daily.bathing
```

the file `daily.bathing` is appended to `smith`'s file `suggestions`. We want the system to work even if the `suggestions` file is read and write protected by its owner.

Describe how the features of the Unix operating system can be used to implement a single program `suggest` which will accomplish our goals. Be sure that your implementation of `suggest` does not introduce any security risks into the system.

Problem 3. (6 points)

A process checks for signals when it enters or leaves the sleep state (if it sleeps at an interruptible priority) and when it returns to user mode from the kernel after completion of a system call or after handling an interrupt. Why does the process not have to check for signals when entering the system for execution of a system call?

This problem is exercise 12 on page 241 of Bach's book.