Things to know by April 6

**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 is a *copy* of *last* year's second midterm in Comp 190.

Midterm 2–April 10, **1989**

Closed book section (64 points)

The exam is to be turned in at 2:50 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. (24 points–4 points each)

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

context switch

page

region

round robin

time slice

u area (user area)

Problem 2. (16 points–4 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!*

`bind`(sd, address, length)

`exit`(status)

`nice`(increment)

`signal`(signum, function)

Problem 3. (1 point)

Name a patented feature of the Unix operating system.

Problem 4. (9 points)

Name six (6) *different* system calls the shell would use in executing the following command:

```
% man csh | grep cd > foo
```

Explain briefly how these systems calls are used. (By they way, there are at least eight that are needed.)

Problem 5. (4 points)

Suppose a C program starting with the following header:

```
main(argc, argv)
     int argc;
     char *argv[];
```

is compiled and the compiled code is stored in the file `surely`. If the command:

```
% surely you remember
```

is executed, what are the values of `argc` and the array elements of `argv`?

Problem 6. (6 points)

How are signals and interrupts similar? How are signals and interrupts different?

Problem 7. (4 points)

Name two bad things that can happen to datagrams?

Midterm 2–April 10, **1989**

Open book section (36 points)

The exam is to be turned in at 2:50 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. (21 points)

I log into two different terminals. On one terminal I start a program called `cpupig` in background. The shell responds with:

```
[1] 4828
```

informing me that `cpupig` is running with process ID 4828.

Immediately, from the other terminal I type:

```
% kill -BUS 4828
```

which sends the SIGBUS (10) signal to process 4828. Now the first terminal, the one with `cpupig`, prints

```
[1] Bus error    cpupig
```

(a) How did the shell know that `cpupig` was running as process 4828?
(b) How did the shell know that `cpupig` was killed by a bus error when the signal was sent from another terminal?
(c) Assuming `cpupig` does not handle bus errors, outline how the kernel sends the signal from `kill` to `cpupig`.

Now suppose `cpupig` *is* handling bus errors with the following handler:

```
sigbus_handler()
  {
     kill(getpid(), SIGBUS);
     printf("SIGBUS caught!\n");
     exit(0);
  }
```

and that `cpupig` is running under a System V operating system like `napoleon` (thus, signals work as described in Bach's book).

(d) Now, how does the kernel set up `cpupig` to receive the `SIGBUS` sent from the other terminal? Don't go into great detail. A picture and four or five sentences should do.
(e) What happens when the `kill` system call within the handler is executed?

By the way, the system call `getpid` returns the process ID of the calling process, in this case 4828.

Problem 2. (15 points)

You, as a Unix guru, have been hired by a strange company. Your first task it to write a program called `cleanup` such that the command

`% cleanup` *file*

will delete *file* if and only if the user running `cleanup` has a user ID less than 1066 *and* *file* is owned by a user whose ID is greater than 1066.

How would you write such a program and install it in your system? Assume you know the superuser password.