CSCI 473: Network Programming
Midterm #1 -- February 20

Write your answers on your own paper. This exam is to be turned in by 12:10PM. For each question, I have given, in addition to the number of points, a suggested number of minutes that you should allocate to the question. This should be a challenging exam!

Problem 1. (25 points) -- [15 minutes, 11:00AM to 11:15AM]

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.

Problem 2. (20 points) -- [10 minutes, 11:15AM to 11:25AM]

Describe how a text editor can use signals to do the following useful tasks:

(a) Save a backup copy of a file if the user hasn't typed any characters for the last 30 seconds.
(b) Save a backup copy of a file when a user's terminal becomes disconnected.
(c) Completely refresh the screen when brought from background to foreground execution.

Please be brief. No more than four sentences each is required.

Problem 3. (10 points) -- [5 minutes, 11:25AM to 11:30AM]

Why is a signal generated for the writer of a pipe when the other end disappears, and not for the reader of a pipe when its writer disappears? [Exercise 3.3, p. 170, Stevens.]

Problem 4. (15 points) -- [15 minutes, 11:30AM to 11:45AM]

Almost all Unix system calls simply return an error value when they fail. Writing to a pipe with no readers is a most unusual exception. Instead of having the **write** system call return a error indication which could then be checked by the programmer, the designers of Unix decided that the **SIGPIPE** signal should be delivered to the process. This action terminates the writing process unless it has installed a signal handler for **SIGPIPE**.

For a minute, imagine yourself as one of the designers of Unix. Justify your extraordinary decision to have writing to a pipe without readers result in a delivery of a signal rather than return of an error value *like every other system call error*.

Problem 5: (30 points) -- [25 minutes, 11:45AM to 12:10PM]

The following program has been successfully compiled and run 100 times on a Unix system. It wrote ``**AA**" to standard output 47 times, it wrote ``**AB**" once, it wrote ``**ABA**" 39 times, it wrote ``**ABB**" 12 times, and it wrote ``**BAA**" once.

First, draw a picture showing the pipes created by the processes.

Why is two the least number of characters that can be written to standard output?

Why is three the greatest number of characters that can be written?

Describe how the program can produce ``**AA**" as its output.

Describe how the program can produce ``**AB**" as its output.

Describe how the program can produce the character `**B**' as the first letter of its output.

```
#include <signal.h>
main(argc, argv)
  int argc;
  char *argv[];
{
  int p[2], duck;
  char b = 'X';
  pipe(p);                    /* 1 */
  write(p[1], "A", 1);        /* 2 */
  duck = fork();              /* 3 */
  if (duck > 0) {             /* 4 */
    close(p[1]);              /* 5 */
    read(p[0], &b, 1);        /* 6 -- read a single character from p[0] into b */
    write(1, &b, 1);          /* 7 -- write the character b to standard output */
    kill(duck, SIGKILL);      /* 8 -- this kills the process with id duck */
    read(p[0], &b, 1);        /* 9 */
    write(1, &b, 1);          /* 10 */
    wait(0); }                /* 11 */
  else {                      /* 12 */
    write(p[1], "B", 1);      /* 13 */
    read(p[0], &b, 1);        /* 14 */
    write(1, &b, 1);          /* 15 */
    exit(0); } }              /* 16 */
```

Possibly interesting facts

If two processes are reading from the same pipe, they cannot read the same characters. Each character when read is removed from the pipe.

If a process tries to read into a buffer by executing:

```
read(p[0], &b, 1);
```

and there is nothing to read, the value of b will not be changed.

And finally, for convenience use the numbers in the comments to refer to individual lines of the program. By the way, some of the answers to the subquestions are as obvious as they seem.