Midterm 2–April 6
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 either on the exam in the space provided (if you can write very small) or on separate pieces of paper.

Problem 1. (8 points)

Why do processes waiting for terminal input sleep at interruptible priority levels?

Why do processes waiting for terminal input sleep at higher priority levels than processes waiting for terminal output?

[This is similar to problem 1 in Section 8.5 of Bach's book.]

Problem 2. (4 points)

When executing untested programs from the shell, a common error message printed by the shell is "Bus error – core dumped." The program apparently did something illegal; how does the shell know that it should print an error message? [This is problem 41 in Section 7.11 of Bach's book.]

Problem 3. (4 points)

The notes to an operating systems course offered by a well-known computer company contain the following discussion of a well-known feature of Unix.:

- Domain bit
  - with each executable file a domain-bit is associated
  - when a file is executed by a user, say Jim, and the associated domain-bit is on, the user-id is temporarily switched to the id of the owner of the file
  - when the execution of the file terminates, user-id is switched back to Jim

  [Stolen from the notes of the **XXX** Corporation.]

What feature of Unix is really being discussed in these notes? How accurate is this description?

Problem 4. (20 points)

When a process attempts to `write` to a pipe which is no longer opened for reading by any process, a `SIGPIPE` signal is sent to the process attempting the write. Unless the process is handling `SIGPIPE` signals, it will be aborted.

Consider the following rather odd program:

```
main() {    /* this program ignores any arguments */
    int foo[2];
    pipe(foo);                  /* 1 */
    fork();                     /* 2 */
    close(foo[0]);              /* 3 */
    write(1, "X", 1);           /* 4 */
    write(foo[1], "X", 1);      /* 5 */
    write(1, "Y", 1);           /* 6 */
    if (wait(0) > 0)            /* 7 */
        write(1, "Z", 1);       /* 8 */ }
```

Be sure to note that: (1) any process executing line 5 is terminated if there are no readers to the pipe; and (2) the `wait` system call in line 7 will return a negative value if the process executing it has no children.

On account of races, the program has several possible sequences of output written to standard output (file descriptor 1). When the program was run 1000 times with its output directed to a disk file, it wrote "`XX`" 8 times, "`XYX`" 188 times, "`XXYZ`" 4 times, and "`XYXZ`" 800 times.

Explain how the sequence "`XX`" can be output?

Explain how the sequence "`XYXZ`" can be output?

Can the sequence "`XYZX`" be output? If so, how? If not, why not?

When the program was ran 100 times with it's output directed to the terminal, it wrote "`XX`" 98 times and "`XYXZ`" 2 times. Explain why "`XX`" is written so frequently when output is directed to the terminal and so infrequently when output is directed to a file.