

Name _____

Exam One

**CS160 - Operating Systems
Drake University - Spring, 2004**

Directions: This is an open book exam. Do all of the following seven problems. Show all work. Please work first on problems with which you are most comfortable.

Problem 1 (10 points) The IBM 360 was an amazingly successful line of computers. What were two very different innovations that it introduced?

Problem 2 (15 points) Using pseudocode (fake code) instead of C code, if you prefer, rewrite the shell code in Figure 1-10 (page 24) so that it checks the parameters (an array of strings = array of character arrays) to see if one of the strings is "&". If so then the shell should not wait for a command to complete before looping back to get another command.

Problem 3 (15 points) As clearly and completely as you can, explain what will happen when the following C code is executed. What can you say about the order in which messages are displayed?

```
int i, status;

for (i = 0; i < 4; i++)
{
    if (fork() != 0)
    {
        printf("hi\n");
        waitpid(-1, &status, 0);
        exit(0);
    } else {
        printf("hello\n");
    }
}
```

Problem 4 (15 points) Assuming that the following code is executed with file descriptor zero referencing the keyboard, and with file descriptor one referencing the screen (monitor), and assuming that no errors occur, describe in detail what happens when the following C code is executed.

```
int m, n, p, r, x;

m = dup(0);
n = dup(1);
close(0);
close(1);
p = open("1.dat", ...);
q = open("2.dat", ...);
scanf("%d", &x); /* read integer from standard input */
printf("%d", x); /* write integer to standard output */
close(p);
close(q);
open(m);
open(n);
```

Problem 5 (15 points) If a user process is interrupted by a clock pulse, and assuming that all tasks are blocked, describe the precise mechanism that results in the clock task running instead of the interrupted user task as soon as the (low level) interrupt handling is completed. Indicate significant locations in the Minix code listing by means of the line numbers in the listing.

Problem 6 (15 points) Describe how the `mini_send` and `mini_rec` functions work in tandem to implement the rendezvous method for message passing. Why don't messages get lost? What happens when several processes try to send a message to the same process at (essentially) the same time, too fast for the receiving process to immediately handle. Indicate the regions of the Minix source code (by line numbers) that deal with the various aspects you discuss.

Problem 7 (15 points) Rewrite the task queue portion of the `ready` function so that it first rotates the task at the head of the queue to the tail of the queue, and then inserts the task to be added to the queue, but inserts it at the head of the queue.