Laboratory 2

Due Date: Monday, February 21, 2000
Points: 100

These problems ask you to make small modifications to the MINIX kernel. You will be modifying the kernel's source code, recompiling and assembling it, using `make(1)` to link the various executable files, and then produce a new MINIX boot "disk."

These programs require only rather minor modifications to MINIX. The key is locating the current code that should be modified; surely, there are many acceptable solutions.

To receive credit for your solutions, show the TAs the modifications you have made to the kernel source code. Also demonstrate your new kernel running programs that show that your new kernel meets the requirements of the problem. Think carefully to come up with programs that demonstrate your new kernel.

Submit a boot disk with your new Minix system to the homework box by the deadline. Put your sources for your changes to the MINIX kernel, and the program(s) you wish to run to demonstrate your new kernel, onto a second disk, and submit that disk as well.

1. **(30 points)** Modify the MINIX kernel to print the word “OUCH!” on the console whenever the user generates an interrupt by pressing the interrupt key on the keyboard. For example, if you type `^C` to terminate your program, the kernel should write “OUCH!” on the screen. After printing the “OUCH!”, the kernel should proceed to handle the interrupt in the usual way.

2. **(70 points)** Modify the MINIX process scheduler so that user processes are scheduled by a multilevel feedback queue with two levels, both scheduled using round robin. The quantum at the first level is the quantum that MINIX currently uses in its round robin scheduling algorithm for user processes. The quantum at the second level is twice that. At the first level, each process gets 2 quanta before being moved into the second level.

   Pick a keystroke (like control-T) that when struck will print the total CPU time used and the current level (first or second) of the currently-running process.

   You will be asked to demonstrate that your new MINIX kernel properly schedules processes. To do this, you might want to have a number of long-running CPU-bound processes around, and then create a few I/O-bound processes and demonstrate that these get scheduled as you would expect.