CS 305: Operating Systems
Department of Computer Science
The University of Western Ontario
Programming Assignment 2
Winter, 2010

Purpose:

The goals of this assignment are the following:

- To gain familiarity with operating system organization.
- To gain familiarity with system calls.
- To gain familiarity with CPU scheduling algorithms.

Part 1: Your first Minix Hacks

In this part of the assignment you are asked to make specific modifications to Minix and answer one or more questions for each modification.

1. In the server source file that executes the *exec system* call, insert a *printf* statement that outputs the string "executing" followed by the name of the file. You should recompile Minix and reboot. Provide a list of what was printed between the login prompt and the command prompt. In the submission you should specify the Minix source file that you modified and the line where you placed the print statement after.
2. Search for the *announce()* function (Hint: This is in the *kernel* directory). This function is responsible for printing a message to the screen stating the version of the operating system about to boot. Use *kprintf* to print "Modified by .." and then the names of the members of your group.

Part 2: A Simple System Call

You are to write a system call, *numberprocs()*, that returns the number of processes currently running in the system.

Part 3: Comparing Scheduling Algorithms

Currently the Minix scheduler maintains 16 queues of processes. System processes use queues 0 to 6. Queue 15 is for the idle process which only runs if there are no other processes to run. The scheduling algorithm finds the highest priority queue that is not empty and picks the process at the head of the queue. Scheduling is round robin for each queue.

In this assignment you will implement other scheduling algorithms and compare them with the current Minix scheduling algorithm. These algorithms are described below.

1. Change the Minix scheduler so that you use only one queue for user processes. This queue should be the one denoted by USER_Q (see *proc.h*). Do not change the way that a process is added to the queue.
2. Change the Minix scheduler so that you use only one queue for user processes. This queue should be the one denoted by USER_Q (see *proc.h*). For any of the queue, when a process is added to the queue it should always be added to the end of the queue.
3. Change the Minix scheduler such that the process selected from a queue is the process with the least amount of CPU scheduling time.

You are to compare these schedulers with the original scheduler. This will be done by running a workload generated by two programs: *scanfiles.c* and *timewaste*.c. The program in *scanfiles.c* is an I/O bound program while *timewaste.c* is a CPU bound program. Both of these are found on the assignments web page. Your comparison should be based on the time it takes to complete *scanfiles.c*. You can use the *time* command which will give you the real time used by a process. You should average times observed over a series of runs since times will fluctuate (we suggest 3 to 5). There is no need to reboot Minix or log in again. You should run the trails in quick succession. You are to provide a report with your results and explain why you think you got the results you have.

Hints

1. Use F1, F9.
2. Use *kprintf*, *printf* as needed.
3. Reading Minix source code can be tedious. A website that may help is the following:

   http://www.raspberryginger.com/jbailey/minix/html/dir_83302e5ef083f5b0e601e389c86705ca.html

   Another useful website is the following: http://www.minix3.org/doc/AppendixB.html

What to hand in

1. Your virtual machine; In the directory */usr/src/kernel* you should have *proc1.c*, *proc2.c*, *proc3.c* for each scheduler in Part 3. The original should be in *proc.c*.
2. Report
   a. Part 1, answer to 1
   b. Part 3's report