Name: _____ **Student ID #:** _____

Lab #4: COMP 3000B (Operating Systems)
November 6, 2007

Please answer all questions below, there are 70 marks total. Part A of this lab is intended to be completed within the lab during Lab hours. Part B can be completed on your own time, either in the lab or on your own computer.

You may find the Linux Cross Reference website, `http://lxr.linux.no/source/`, useful when completing this assignment.

1 Part A

This section is designed to be completed in the lab. You get 10% of the total marks for attempting to do part A during assigned lab hours. Please ensure that one of the Lab instructors takes your attendance.

1. **[7]** Have the instructor mark down that you were present and attempted part A during lab hours.

2. **[3]** In the Linux kernel, what file contains the kernel entry point code for system calls that execute on the machines in the lab? What language is it written in? Why? (Hint: searching for sysenter may be helpful)

3. **[2]** What is the highest numbered system call currently implemented? What is the name of this system call?

4. **[2]** Look in the file linux-2.6.18/fs/ext3/file.c. What function implements the read operation for ext3 filesystems? The write function?

5. **[2]** In what file are the directory inode operations of ext3 defined? In what structure?

6. **[3]** What function (from what file) calls do_ execve() on the kernels running in the lab? Why can't the system call dispatcher directly call do_execve()?

7. **[2]** Briefly, what is the purpose of the `bprm` structure as used in the function do_execve()?

8. **[4]** In order to complete the rest of the lab, you must patch the stock kernel with some code specific to comp3000. The patch is `comp3000-lab4.patch`. Unpack the kernel and apply the patch. What command line did you use to apply the patch and what directory did your run the command from?

2 Part B

The kernel source on the lab computers contains a skeleton file which will be used in completing this lab. This file is located at `kernel/comp3000.c`. Currently, the code in the file creates a proc filesystem entry at `/proc/comp3000` that outputs *Hello World.* In this lab, you will be modifying the file comp3000 to do more.

1. **[10]** Modify `kernel/comp3000.c` to output the process number and name of the process which has a PID closest to 1000 (without going over).

2. **[20]** Modify `kernel/comp3000.c` to output a complete list of process ID and command lines. Your output should be similar to that seen when running the `ps -e --format "pid args"`.

3. **[15]** Modify `kernel/comp3000.c` so that you can write a number to the proc filesystem and subsequent reads will return all processes with ID's above the number written. Writing an ID number of 0 should result in all processes being shown when reading from the proc file.